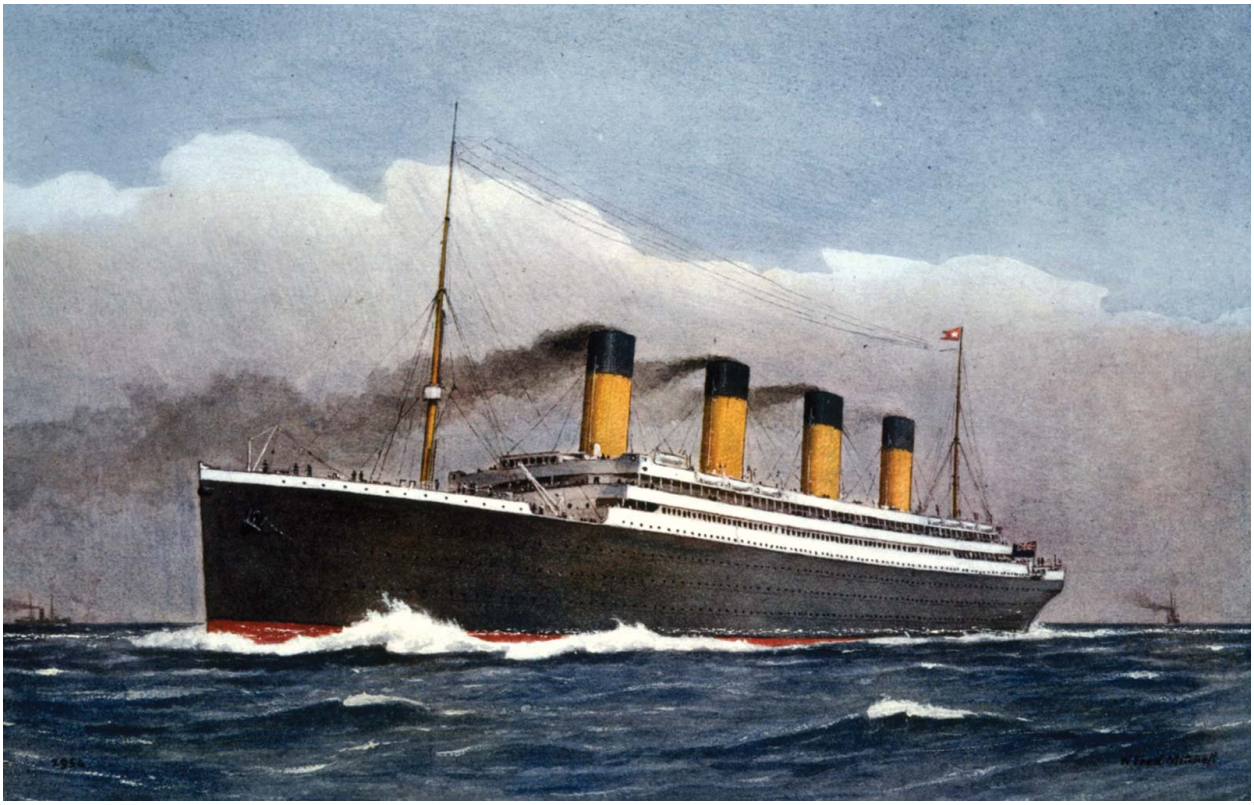
R Final Project: The Titanic

Daniel Alonso & Ander Iturburu

October 28th, 2020

Contents

The Titanic disaster dataset



Introduction

In this project we will explore the Titanic Passenger's dataset. We intend to make a surface description of the dataset through plots, analysis of such plots and detailed descriptions of our observations relating them. We will also use caret to analyze the relationship of one of our categorical variables in terms of other variables (in our case, mortality of the titanic's passengers).

The project will be divided in two parts: The descriptive analysis and the use of caret to relate the variables. Each part will explore a few key points that we consider quite interesting. We want to create a general idea of the content of the dataset in order to later on effectively know what might have affected the mortality of passengers in the titanic. The teachings from such a tragic event must've taught us something right?

A few key points to explore will be:

- How are our most relevant variables distributed?
- How do these variables respond to grouping by categorical variables in the dataset?
- Which variables are correlated to which?
- What is really relevant when it comes to determining a possible increase in risk of death of the passengers?

Among other things, which hopefully shall paint an accurate full picture when it comes to the risk of death in this tragic event.

First off let's load the libraries we'll use during this project

```
library(dplyr)
library(ggplot2)
library(fitdistrplus)
library(PerformanceAnalytics)
library(reshape2)
library(vcd)
library(EnvStats)
library(scales)
```

How does the data look like?

Importing the data

```
data = read.csv('data/titanic.csv')
```

Dataset head

```
head(data)
```

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

Ommitting nans on a separate dataset

The purpose of this is to have a dataset with all the content which we will use for predictions, and a dataset with all the data, except nan values. The latter will be used exclusively for plotting and for descriptive analysis.

```
data_n <- na.omit(data)
```

Variables

ID:

- PassengerId: A unique identifying number for every passenger listed in an SQL-like fashion

Other identifying variables:

- Name: Name of the passenger
- Ticket: Ticket number/code for the passenger
- Cabin: Cabin the passenger boarded the cruise in

Continuous:

- Fare: Amount paid by the passenger to board the cruise
- Age: Age of the passenger during the cruise boarding in years

Discrete:

- SibSp: This is the total amount of siblings/step-siblings and spouses that a passenger has
- Sex: the passenger's sex
- Survived: a boolean variable that describes whether the passenger survived or not
- Pclass: a variable that describes the class in which the passenger sailed aboard the cruise ship (1st,2nd,3rd class)
- parch: This is the total amount of parents (mother, father) or children (son, daughter, step-son/daughter) that a given passenger has

Part 1: Descriptive Analysis

Histogram for Age

```
hist(data$Age)
```

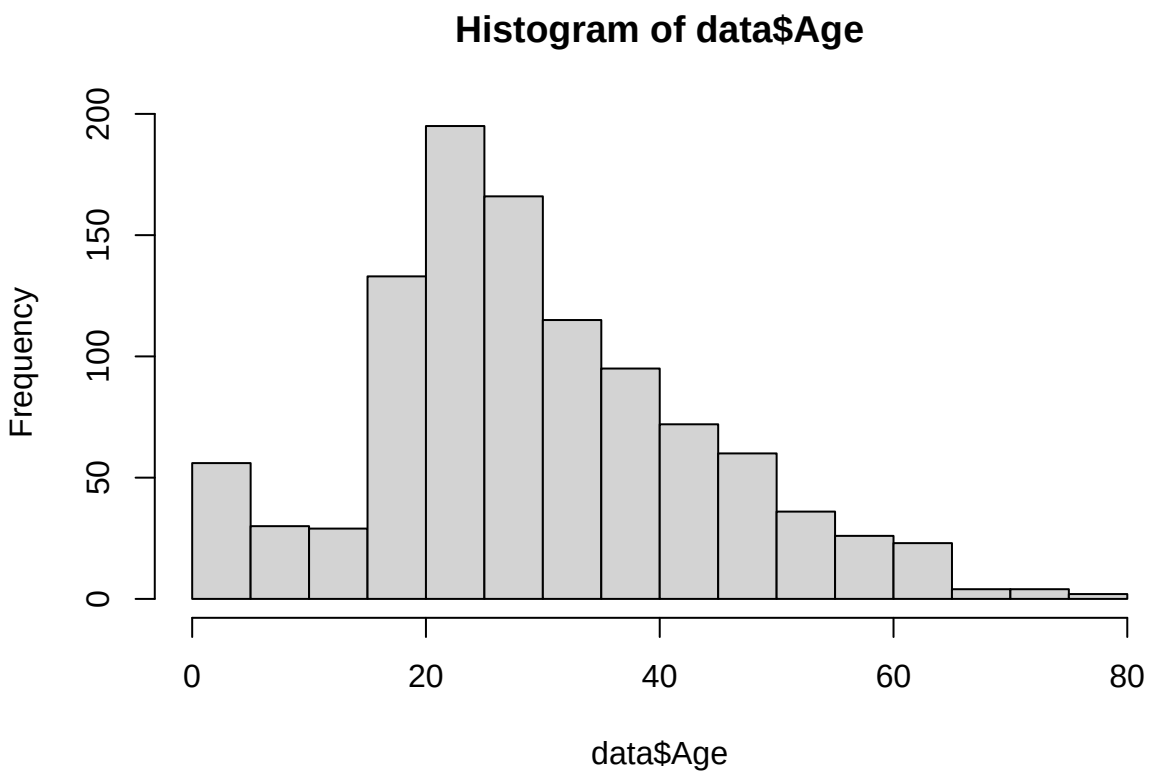


Figure 1

In Figure 1 we can see the age distribution shows the mean age between 25-35 years old. We can see there is a significantly larger right tail than left tail. With a large amount of young children, all, more than likely related to other passengers (their parents). The plot resembles a normal distribution in its concentration of individuals around the mean.

There are very few old people, which we could speculate could be due to the significantly lower life expectancy of back then (between 51 and 55 years old in 1910-1920 UK), or due to the fact that maybe a trip aboard the ship might have been aimed to families. Either way, there aren't many people above 70 (<5) and none above 80.

Histogram for Fare

```
hist(data$Fare)
```

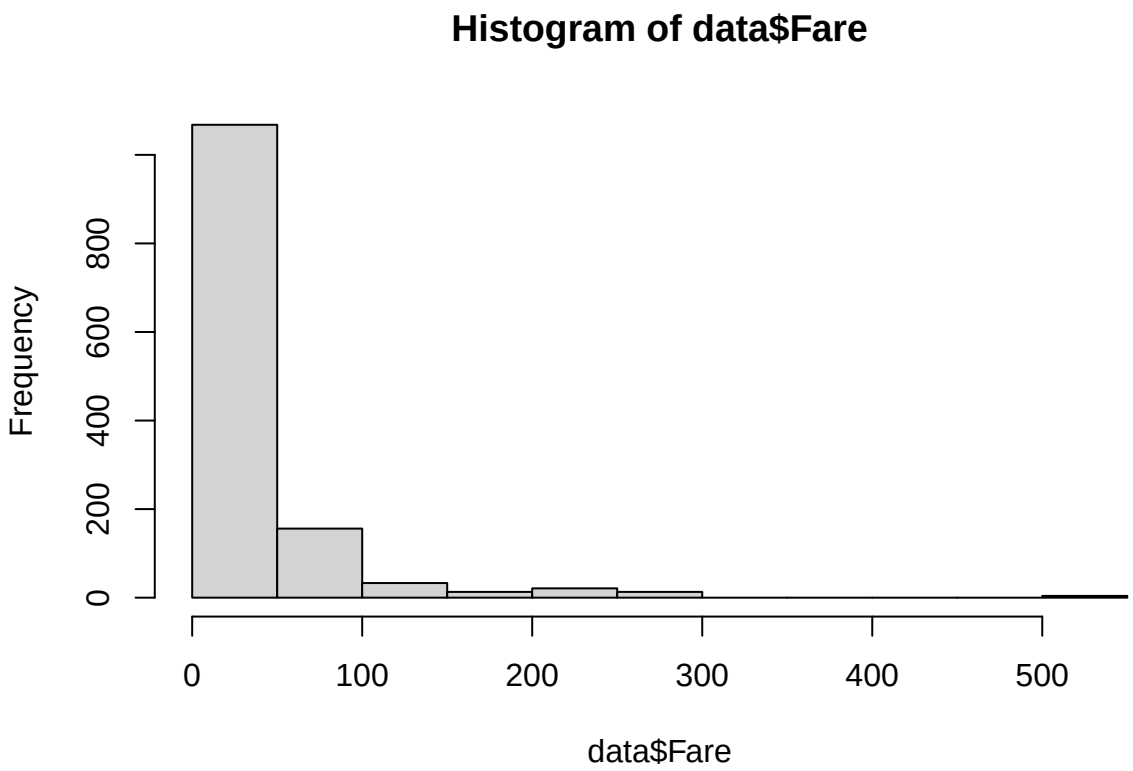


Figure 2

According to Figure 2 on our second continuous variable (Fare), we notice a significantly different situation Vs Age. We have a massively long left tail, as a result of some extremely highly priced tickets payed by a few passengers. Which shows a large and seemingly empty space between fares priced around 250-300 GBP and about 500+ GBP, with little to no tickets payed, showing evidence that some of the tickets, while expensive and very rare among all the individuals (250-300) are still far from the maximum Fare payed (500+). We can clearly see the data is extremely right-skewed.

Extremely expensive tickets, while interesting, are also quite rare. We see that almost all the data concentrates around the 0-100 GBP cost. None being actually free, but yes between 0-1 GBP. We could theorize such cheap tickets probably belonged to children, while most normal tickets were above 20 GBP.

For reference, the ticket prices to board the titanic were the following (in 1912 GBP):

- First Class (parlor suite) — £870
- First Class (berth)— £30
- Second Class — £12
- Third Class — £3 to £8

The conversion rate for £1 (in 1912) would be £115 (in 2019), which shows that even the cheapest tickets (while maybe relatively cheap to board a transatlantic cruise) were still relatively expensive by 1912 standards.

Fitting the normal distribution to Age

```
plot(summary(fitdist(data_n$Age,"norm")))
```

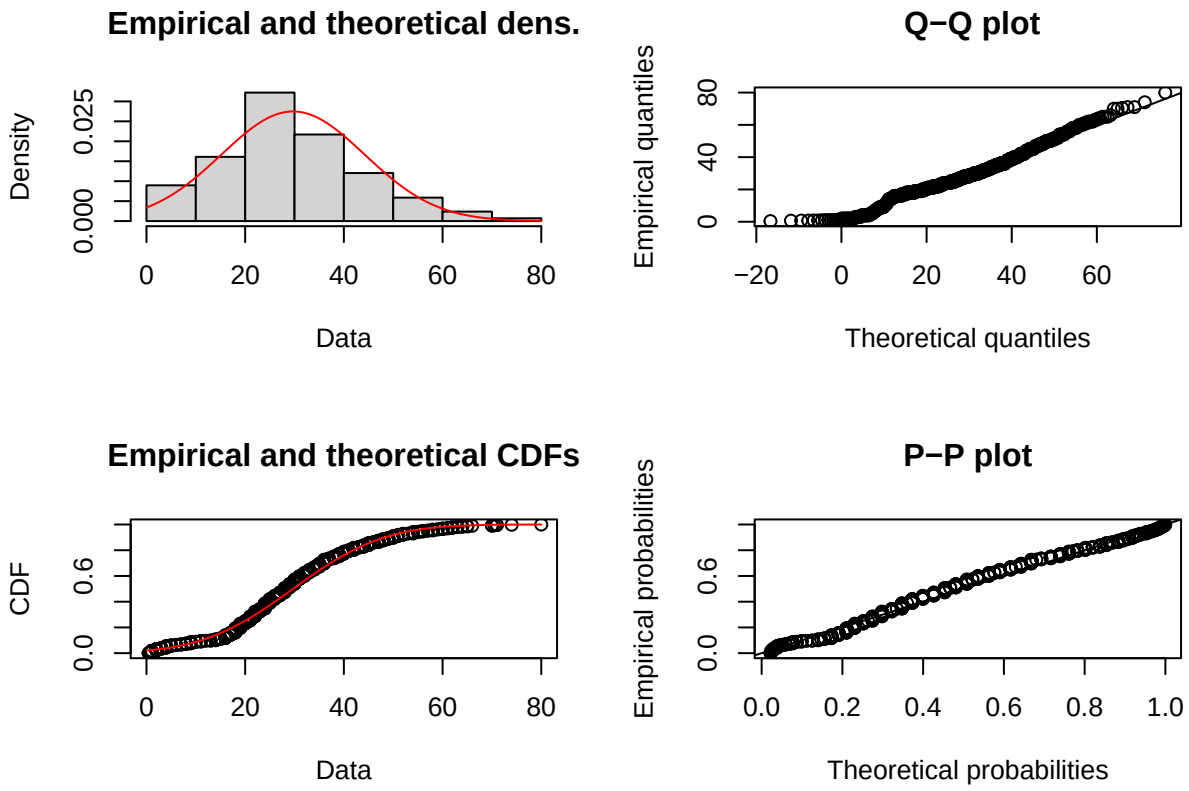


Figure 3

The age variable seems to show relative normality pretty much across the board with the exception of the slightly larger than usual right tail, as we saw in the histogram previously. This right tail can also be seen differring from a normal distribution in the QQ-plot, however, this overall doesn't hurt the relative normality the variable shows.

We can see the mean very very slightly to the right of a gaussian curve and a slightly longer left tail. Given the current size of the sample we could, perhaps, theorize that the set probably comes from a normal distribution.

An interesting thing to think about would be to, with proper statistical analysis, extrapolate the sample data to the population (everyone aboard the titanic) to see if it accurately describes the reality of the whole ship crew plus passengers. It would be out of the scope of this project, but quite interesting to think about.

Fitting a gamma distribution to Fare

```
plot(summary(fitdist(data_n$Fare+1,"gamma")))
```

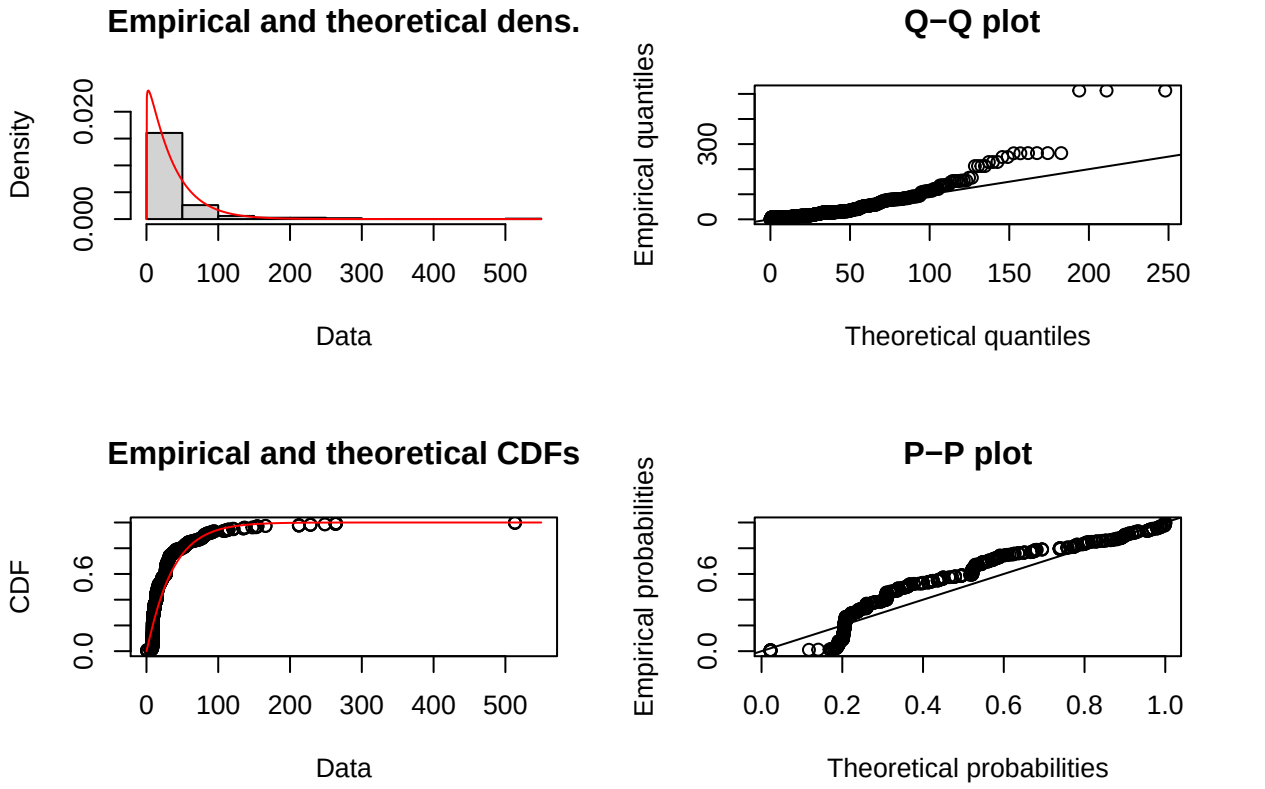


Figure 4

First of all, before plotting, we must sum a constant to Fare, we have chosen 1 to keep values relatively close to the original ones. As some fares are too low and by the nature of the gamma distribution these values would explode.

We can see that the data seems to follow a gamma distribution , with the exception of its incredibly long tail, which to an extent seems to fall above the theoretical quantiles in the QQ-plot, while the PP-plot seems to favour the left tail above the right tail.

We particularly like this variable as it is interesting to play with and seems to say a lot about the passengers aboard the boat and definitely does not seem to exhibit normality as most of the entries concentrate above relatively low prices, which matches very well the proportion of people in each class.

Boarding the ship wasn't exactly cheap, but we know for a fact that luxury tickets were significantly more expensive than third/second class tickets, therefore making the amount of people able to afford such cabins significantly lower.

Boxplot for Age

```
five <- data.frame(x=rep(1,5), five=fivenum(data_n$Age))

ggplot(data=data_n, aes(x=0, y=data_n$Age)) +
    geom_boxplot() +
    theme(text = element_text(size=18)) +
    ylab("Age") +
    scale_x_discrete(breaks = NULL) +
    geom_text(data=five, aes(x=0, y=five, label=five),nudge_x =0.5)
```

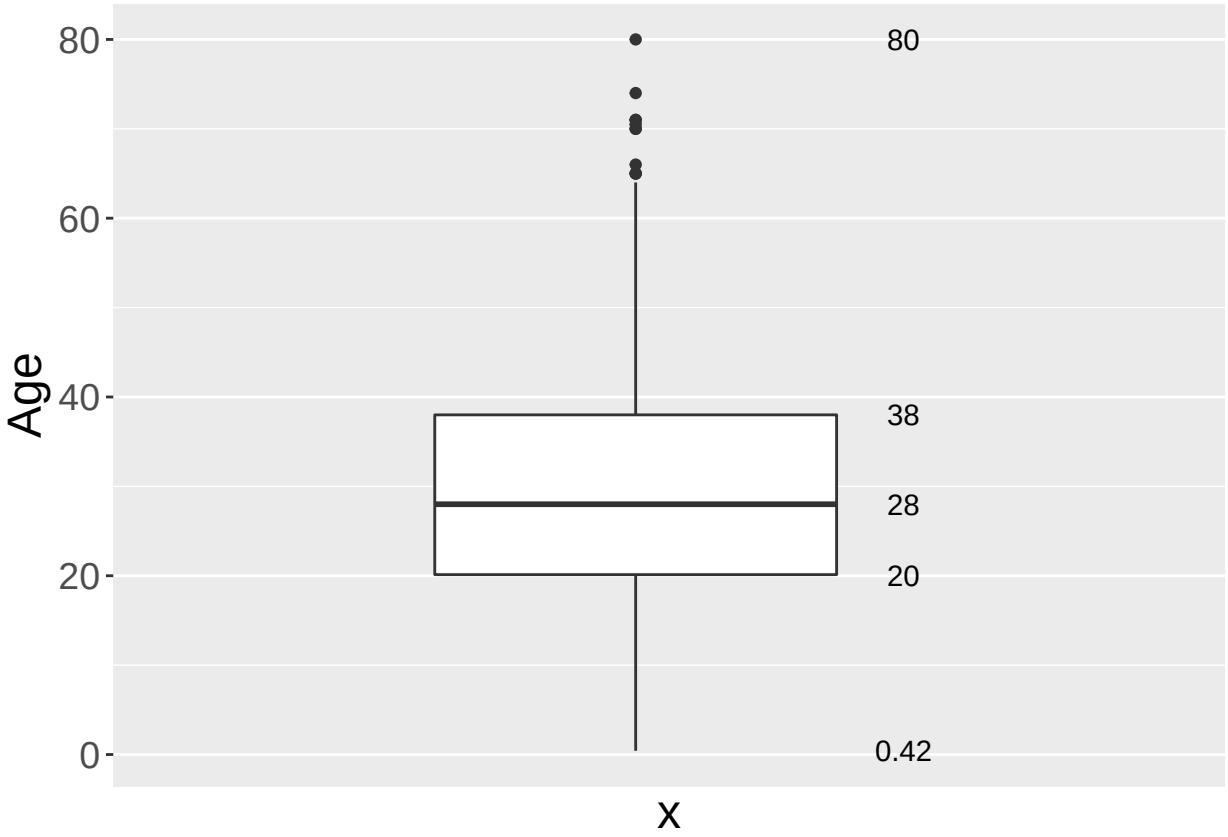


Figure 5

In the Figure 5 we see a boxplot for age with the median at 28, the 25th percentile at 20, the 75th percentile at 38, maximum value at 80 and minimum value at 0.42 (referring to a baby of less than 1 year old).

This shows the wide margin of ages within the ship very clearly. There were all sorts of people of all ages, predominantly men, but definitely all ages.

Older people are a bit more scarce for many reasons, but we could easily attribute such disparity to the life expectancy at the time, which we mentioned previously sat at around 51-55 years old in England.

The relative fragile state of old people during these times, the fact that they might have been significantly more frail than nowadays our old family members are could have discouraged them. Or maybe the cruise trip was simply marketed for younger people and families. However, all these are merely hypotheses.

Boxplot for Fare

```
five <- data.frame(x=rep(1,5), five=fivenum(data_n$Fare))

ggplot(data=data_n, aes(x=0, y=data_n$Fare)) +
    geom_boxplot() +
    theme(text = element_text(size=18)) +
    ylab("Fare") +
    scale_x_discrete(breaks = NULL) +
    geom_text(data=five, aes(x=0, y=five, label=five),nudge_x =0.5, size=1.7)
```

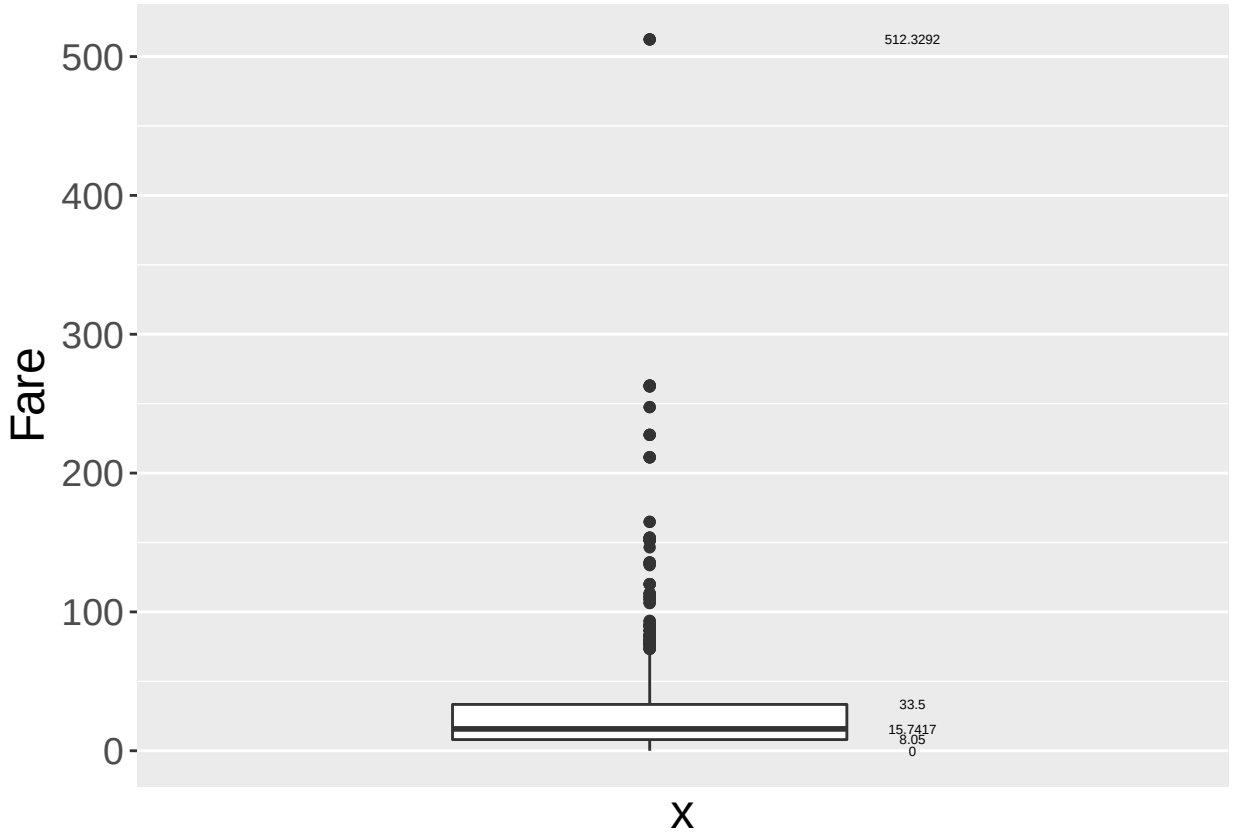


Figure 6

This boxplot for fare shows a really interesting pattern, similar to what the histogram painted. We can see the 25th and 75th percentile at 8.05 and 33.5 respectively, while the values in between represent most of the data and therefore most of the passengers.

All values above 100 are quite scarce, given that even at 100 GBP, the price was already significantly above the 30 GBP base fare for 1st class. These extreme values which peak at a very far maximum of about 512.33 (the maximum fare) represent a very small subset of very wealthy passengers above the ship.

Our minimum value is 0, which might represent children who probably didn't pay a fare, as maybe part of an offer for very very young children, which were, in fact, aboard the ship.

Quantiles for Age

```
plot(quantile(data_n$Age))
```

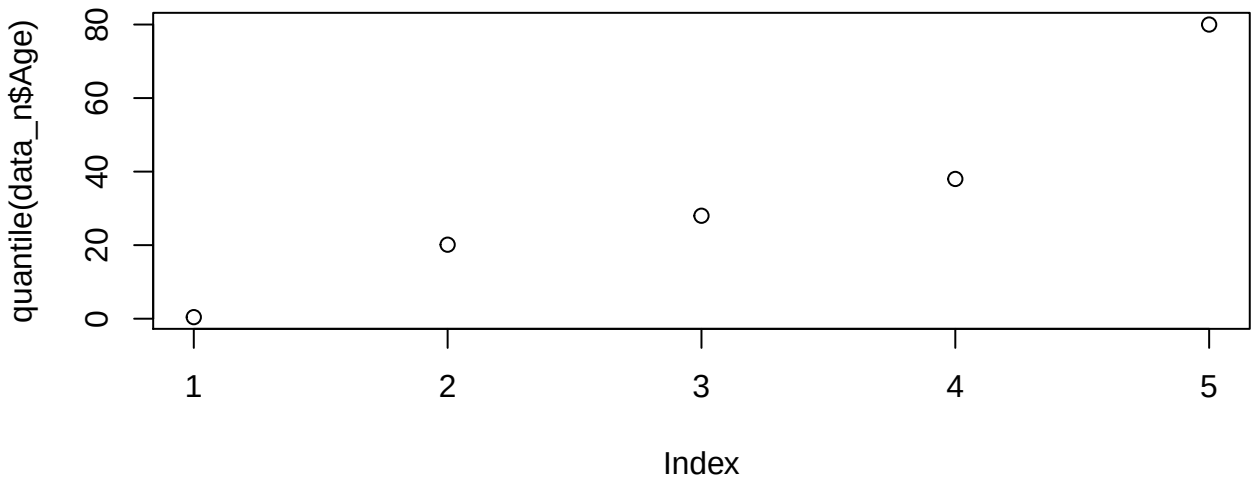


Figure 7

In the following plot we can see the quantiles more clearly for Age.

Quantiles for Fare

```
plot(quantile(data_n$Fare))
```

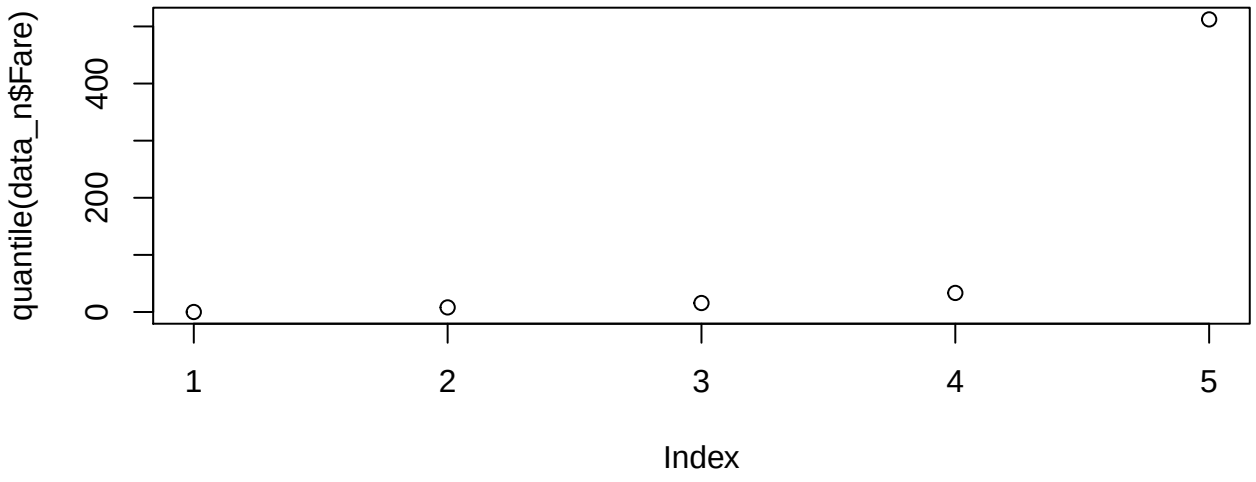


Figure 8

In the following plot we can see the quantiles more clearly for Fare

Both these plots clearly show the extreme nature of maximum values, especially for Fare.

Correlation between Age and Fare

```
log_data_n = data_n %>% mutate(logAge = log(Age), logFare = log(Fare+100))
ggplot(data=log_data_n, aes(x=logFare, y=logAge)) +
    geom_point() +
    theme(text = element_text(size=18))
```

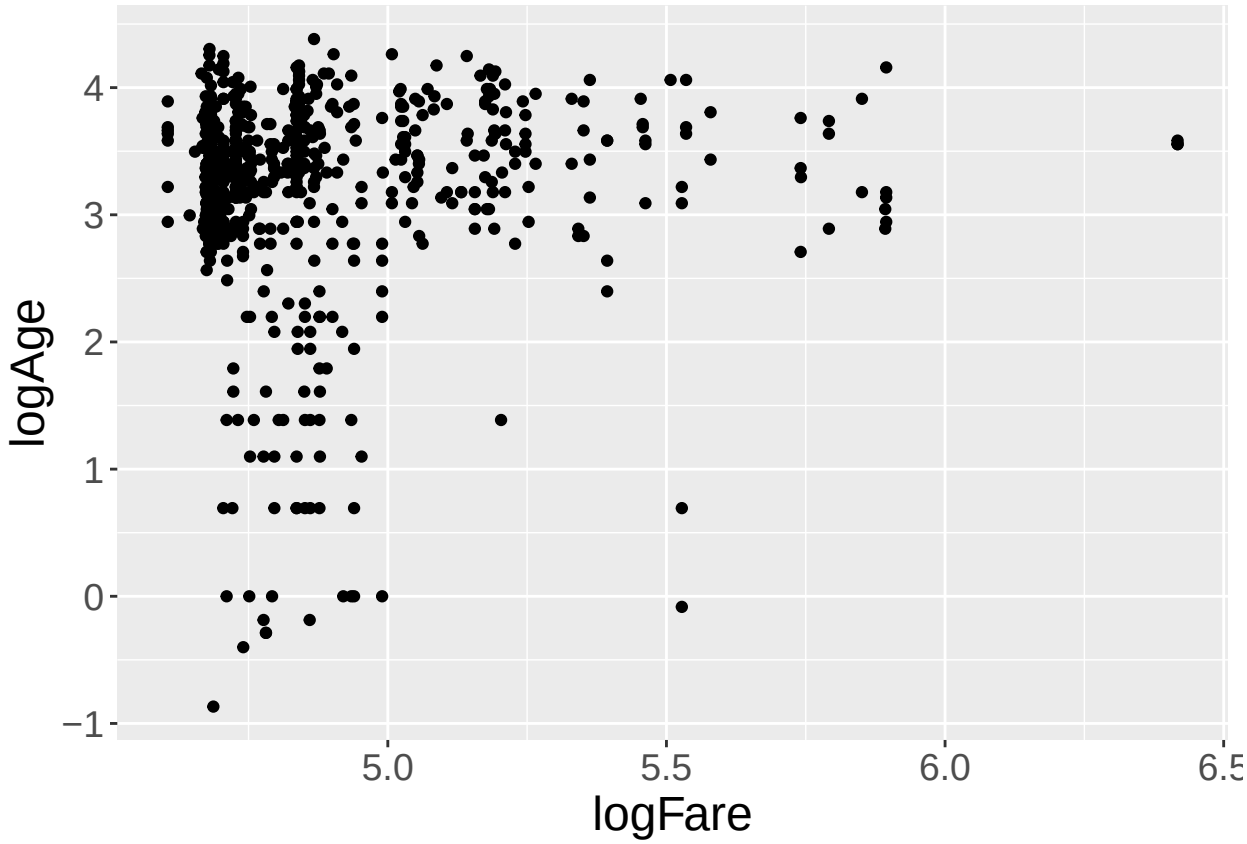


Figure 9

We decided to apply a log transformation for our plot in order to maybe try to make a plot that would better resemble one with a higher spearman correlation coefficient than what a pearson correlation coefficient would yield. However, such coefficient is not significantly larger, as the variables are not quite correlated.

The correlation is extremely low and even if we could draw a line to represent the shape both variables combined would take, we would really not show anything particularly relevant.

We decided to calculate the coefficient itself in order to be sure that it was as low as we expected.

```
cols <- data_n %>% dplyr::select(Age, Fare)
cor(cols, method="spearman")
```

```
##             Age      Fare
## Age  1.0000000 0.1350512
## Fare 0.1350512 1.0000000
```

The result is about 0.135, which is, as expected, extremely low. We can faintly see the shape of somewhat of two typical spearman correlated variables. We can simply conclude that these two variables are not correlated, and thus age couldn't accurately predict fare nor the opposite.

What about older people?

Maybe if we segregated the age of the passengers before calculating the correlation, we could get a more interesting picture, let's try that out.

Correlation between Age and Fare for passengers older than 55 years of age

```
log_data_n = data_n %>%
             filter(Age >= 55) %>%
             mutate(logAge = log(Age), logFare = log(Fare+100))
ggplot(data=log_data_n, aes(x=logFare, y=logAge)) +
    geom_point() +
    theme(text = element_text(size=18))
```

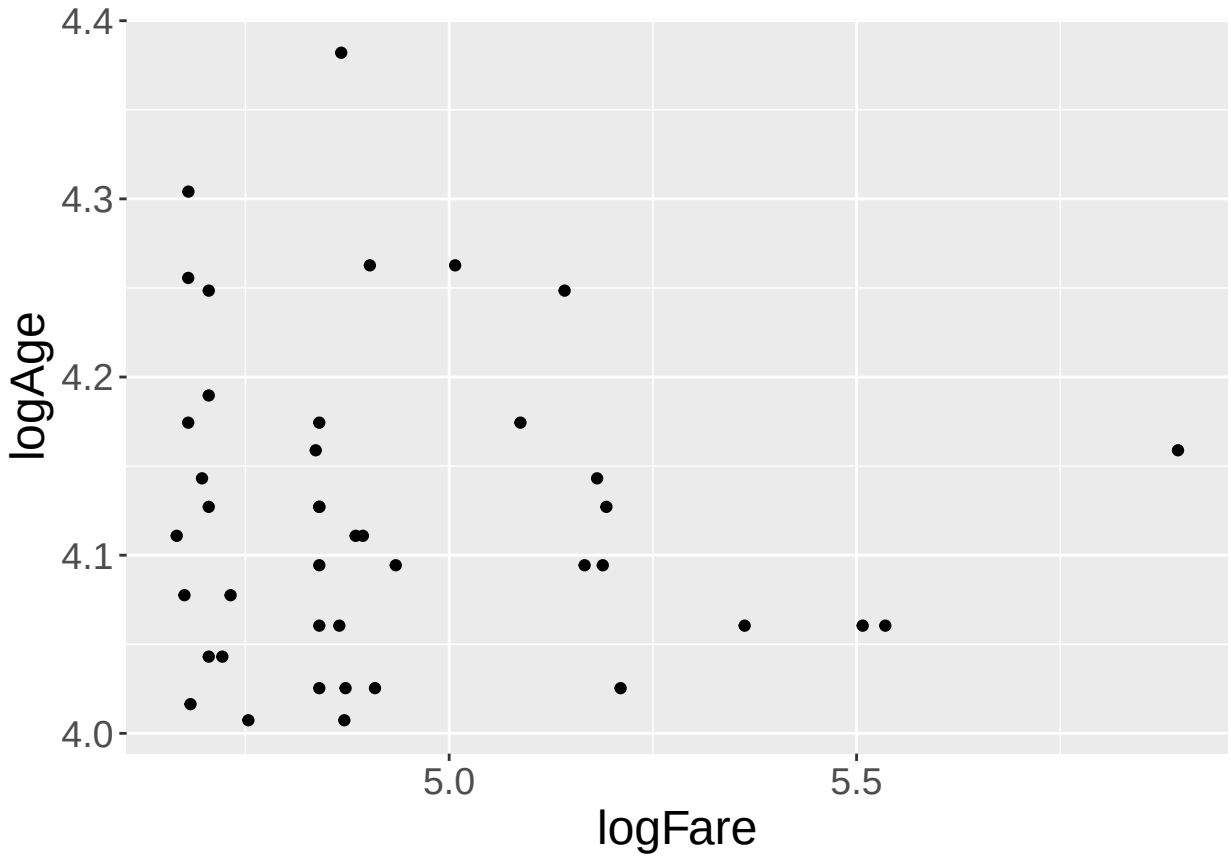


Figure 10

```
cols <- log_data_n %>% dplyr::select(logAge, logFare)
cor(cols, method="spearman")
```

```
##             logAge    logFare
## logAge   1.0000000 -0.1050426
## logFare -0.1050426  1.0000000
```

We stick with our choice of using spearman, but this definitely did not work out for the older passengers.

What about ages between 25th and 75th percentiles?

Correlation between Age and Fare for passengers between 20 and 38 years of age

```
log_data_n = data_n %>%
              filter(Age >= 20 & Age <= 38 ) %>%
              mutate(logAge = log(Age), logFare = log(Fare+100))
ggplot(data=log_data_n, aes(x=logFare, y=logAge)) +
    geom_point() +
    theme(text = element_text(size=18))
```

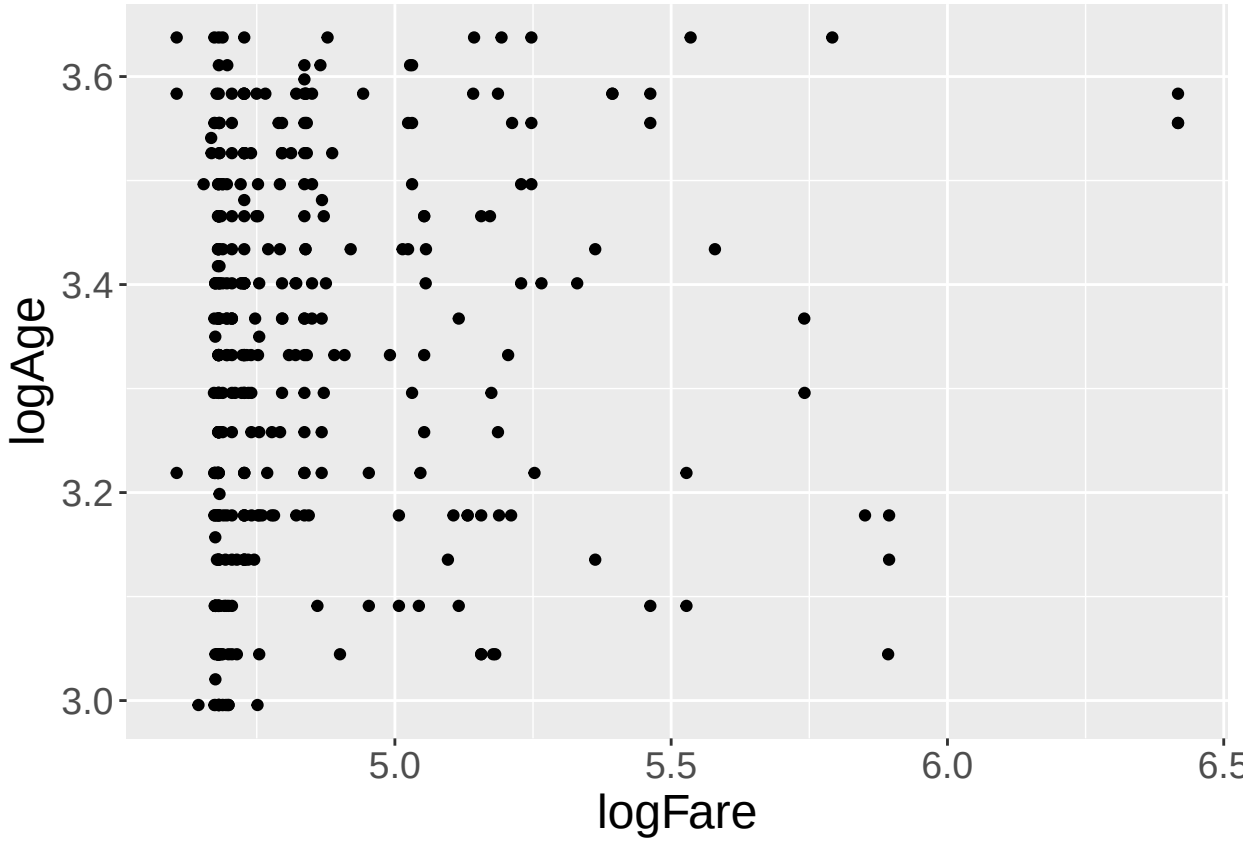


Figure 11

```
cols <- log_data_n %>% dplyr::select(logAge, logFare)
cor(cols, method="spearman")
```

```
##            logAge   logFare
## logAge  1.0000000 0.2342904
## logFare 0.2342904 1.0000000
```

A tad bit better but still definitely not a stronger correlation.

What about ages below the 25th percentile?

Correlation between Age and Fare for passengers under 20 years of age

```
log_data_n = data_n %>%
              filter(Age < 20 ) %>%
              mutate(logAge = log(Age), logFare = log(Fare+100))
ggplot(data=log_data_n, aes(x=logFare, y=logAge)) +
    geom_point() +
    theme(text = element_text(size=18))
```

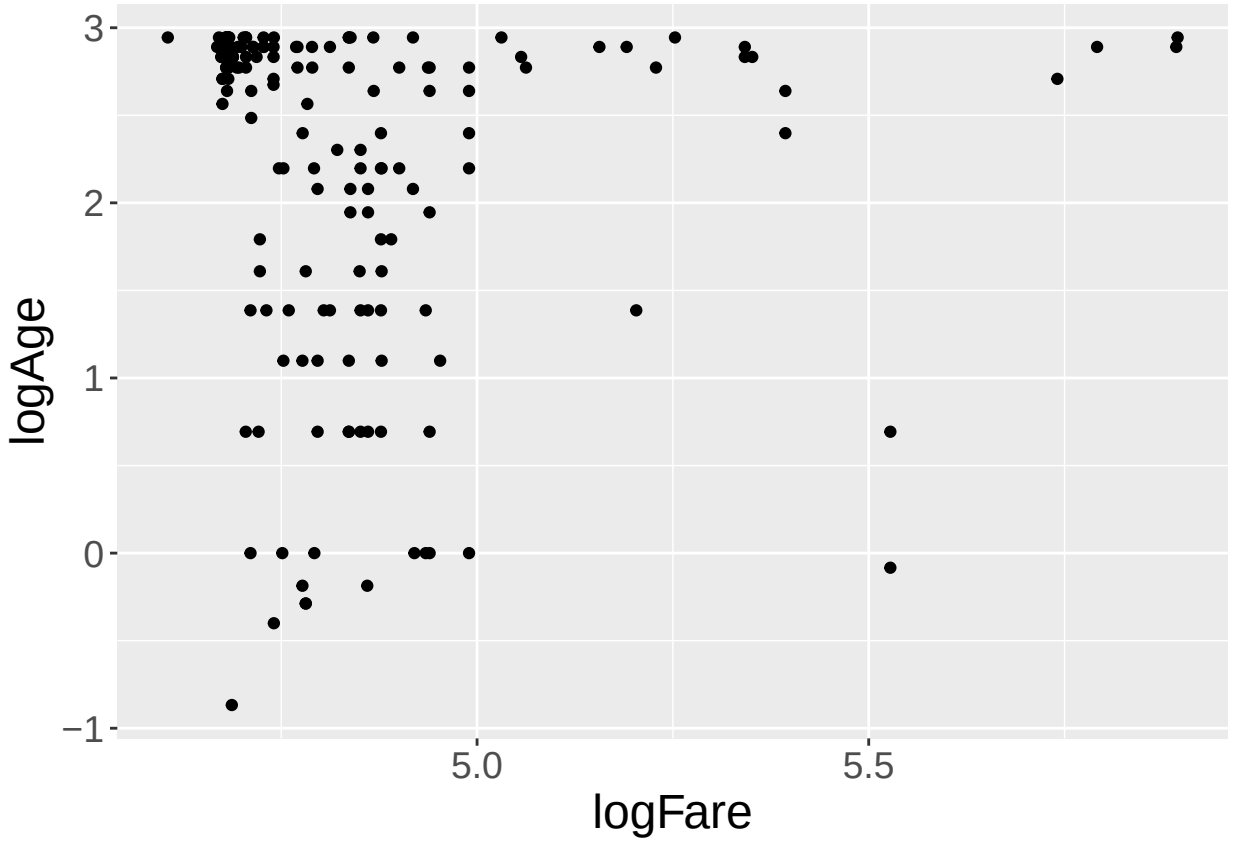


Figure 12

```
cols <- log_data_n %>% dplyr::select(logAge, logFare)
cor(cols, method="spearman")
```

```
##             logAge    logFare
## logAge   1.0000000 -0.3443596
## logFare -0.3443596  1.0000000
```

Interesting!, so this is definitely stronger than our first example. We can say that most young people and kids are probably coming with their parents, therefore likely paying a cheaper ticket.

Performance Analytics plots

With a performance analytics plot we can see a histogram for each selected variable, a scatter plot with a plotted line through the points and the correlation number. Showing the power of custom R libraries!

```
pa <- data_n %>% dplyr::select(Age, Fare)
chart.Correlation(pa, histogram=TRUE, pch=19, method="spearman")
```

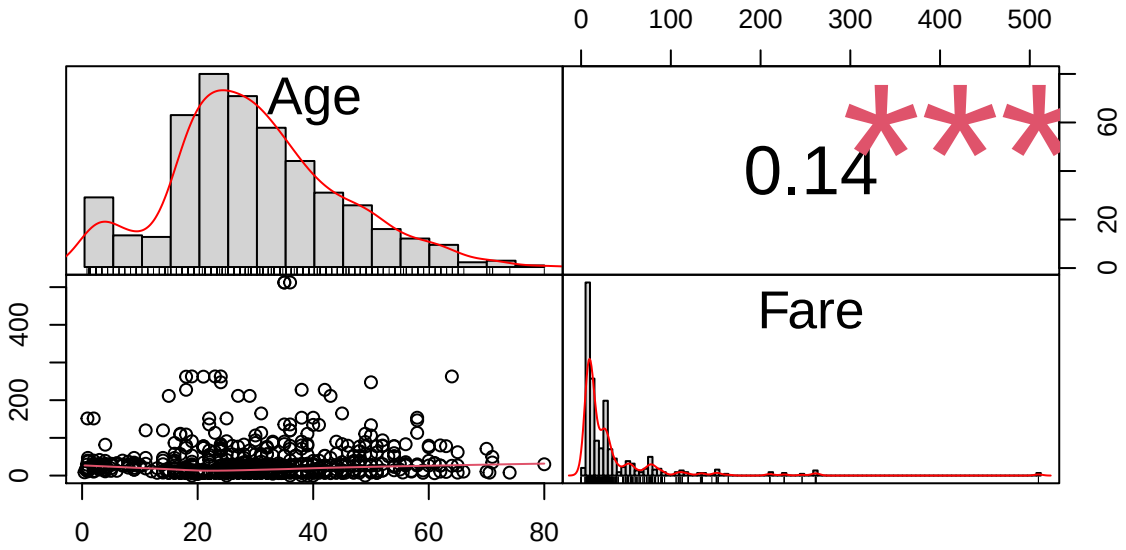


Figure 13

Interesting!, so this is definitely stronger than our first example. We can say that most young people and kids are probably coming with their parents, therefore likely paying a cheaper ticket.

We can try using the dataset with the log transformations to view the scatter plot a bit less... messy

```
pa <- log_data_n %>% dplyr::select(logAge, logFare)
chart.Correlation(pa, histogram=TRUE, pch=19, method="spearman")
```

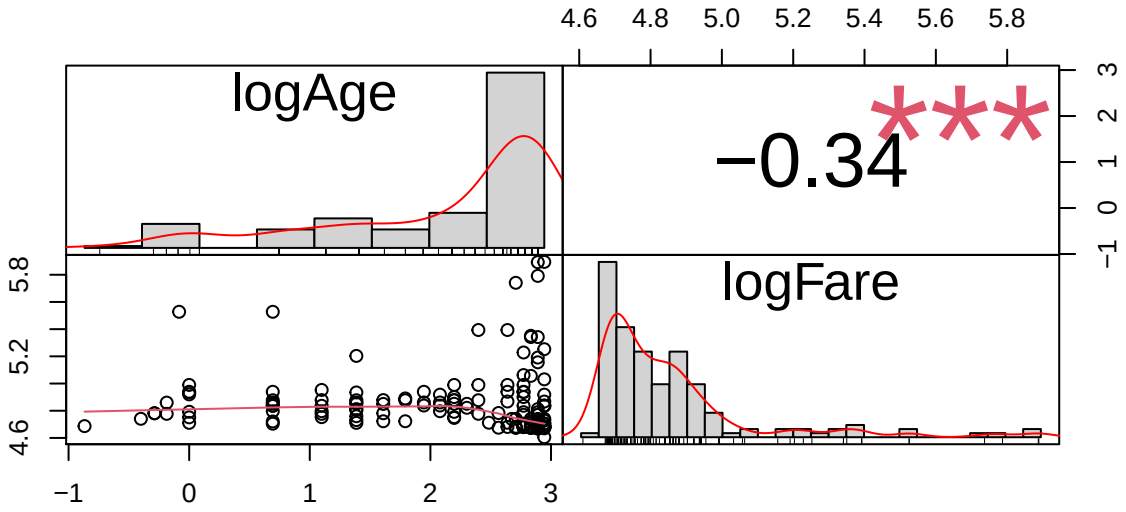


Figure 14

This didn't quite do much to the scatter plot but did give us a better correlation... suspicious.

Proportions of passengers per sex

```
# proportions for sex
sex <- data.frame(prop.table(table(data_n$Sex)))
colnames(sex)[1] <- "sex"
sex
```

```
##       sex      Freq
## 1 female  0.3655462
## 2 male    0.6344538
```

```
# plotting bars
options(repr.plot.width = 8, repr.plot.height = 8)
ggplot(sex, aes(x=sex, y=Freq, fill=sex)) +
 geom_bar(position="dodge", stat="identity") +
 geom_label(aes(y=Freq, label = scales::percent(round(Freq,4), accuracy=0.01)),
            position=position_dodge(width=0.9),
            show.legend=F)+
 theme(text = element_text(size=18))
```

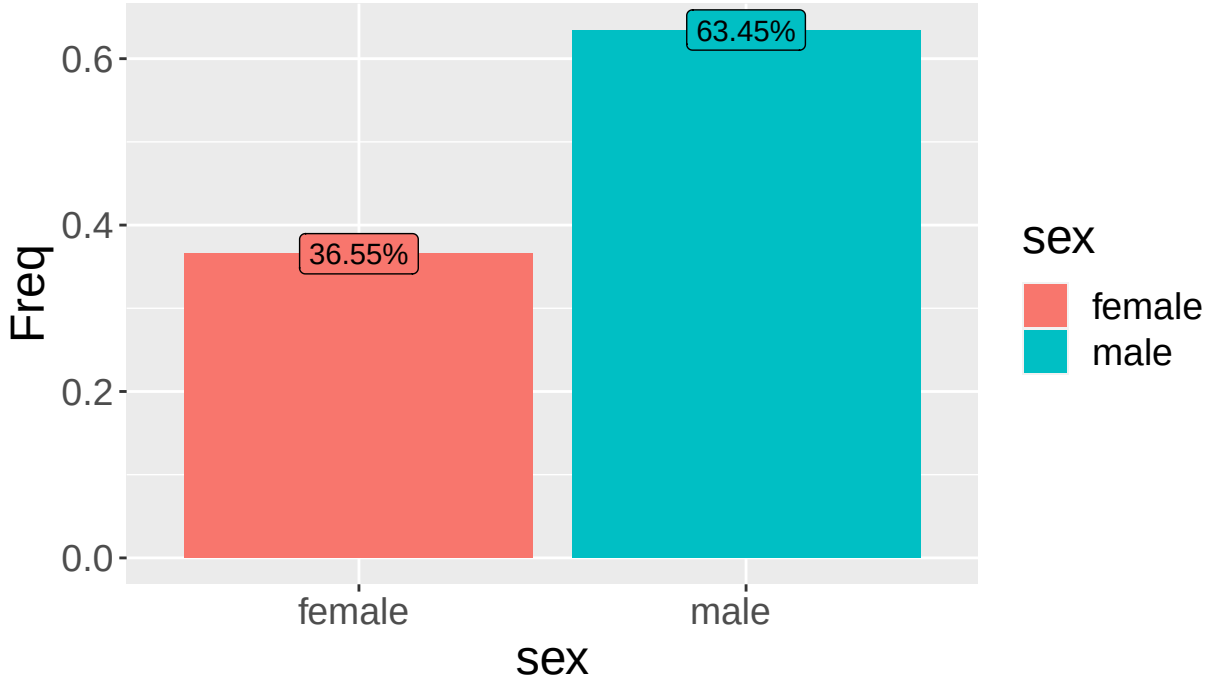


Figure 15

We can clearly see that the large majority of patients were male, we nearly twice as many males than females. We could hypothesize that this could be due to the fact that the majority of the crew was make, or maybe females were less likely to risk boarding a cruise, maybe males were more able to pay for the ships given that most females certainly earned less than males in 1912, there could be many reasons for this. The reality is we can't know for sure, but the numbers show an interesting trend.

Let's see what else the sex variable can offer us when grouping.

Grouping passengers by sex

```
# summarizing fare, age by mean
sex <- data_n %>% group_by(Sex) %>% summarize(Fare = mean(Fare), Age = mean(Age))
sex
```

```
## # A tibble: 2 x 3
##   Sex        Fare   Age
##   <chr>     <dbl> <dbl>
## 1 "female "  47.6  27.9
## 2 "male   "  27.3  30.7
```

```
# melt
sex <- melt(sex, id.vars=c("Sex"), measure.vars=c("Fare","Age")) %>% mutate(variable = as.character(vari

# plotting lines for the 3 variables selected
ggplot(sex, aes(y=value, x=variable, fill=Sex)) +
  geom_bar(position="dodge", stat="identity") +
  theme(text = element_text(size=18))+
  geom_label(aes(y=value, label = round(value,2), accuracy=0.01),
             position=position_dodge(width=0.9),
             show.legend=F)
```

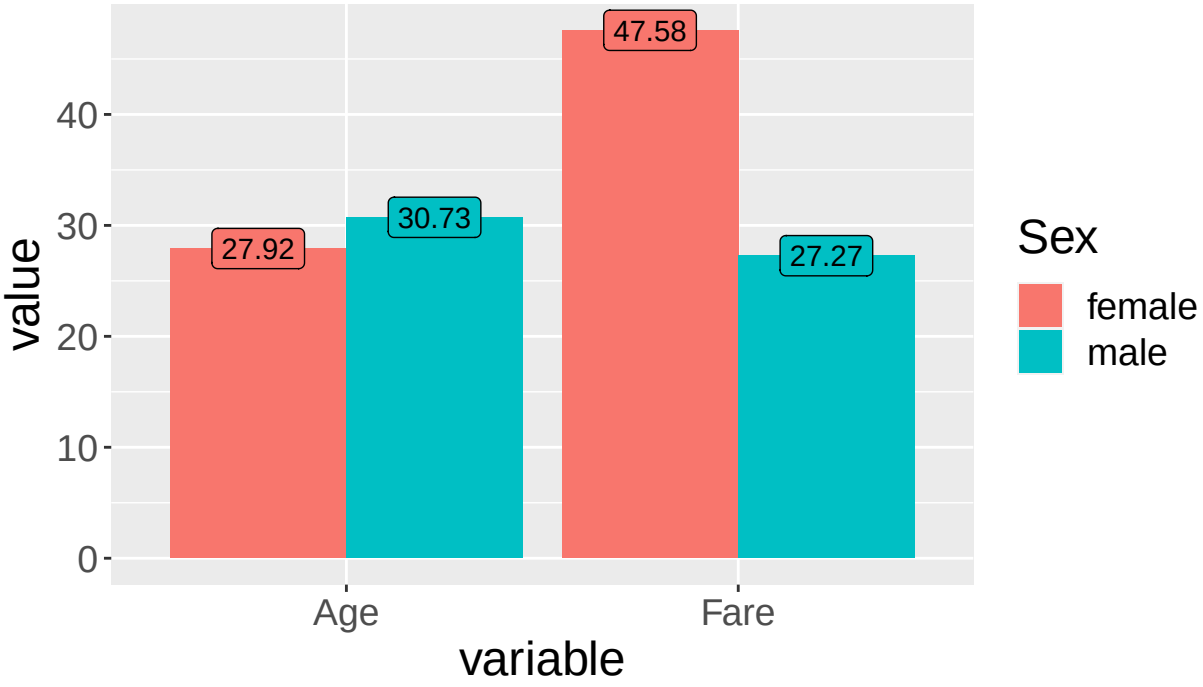


Figure 16

Grouping passengers by sex shows an interesting story.

We can see that females are slightly younger than males, which accurately portrays an interesting reality of our general demography, but specifically the demography of 1912 England, and this is that men used to prefer to marry younger women. This trend seems to remain in the 21st century, but perhaps not as clearly evident as back in the 20th century and before. where this was pretty much the reality in most households.

For some odd reason, females seem to pay significantly more than males to board the ship. And apparently, on average, about twice as much.

Survival rate per sex

```
# summarizing survived by mean
sex <- data_n %>% group_by(Sex) %>% summarize(Survived = mean(Survived))

head(sex)
```

```
## # A tibble: 2 x 2
##   Sex       Survived
##   <chr>        <dbl>
## 1 "female "    0.755
## 2 "male   "    0.205
```

```
# sex groupby survive
ggplot(sex, aes(y=Survived, x=Sex, fill=Sex)) +
  geom_bar(position="dodge", stat="identity")+
  geom_label(aes(y=Survived, label = scales::percent(round(Survived,4), accuracy=0.01)),
             position=position_dodge(width=0.9),
             show.legend=F)+
  theme(text = element_text(size=18))
```

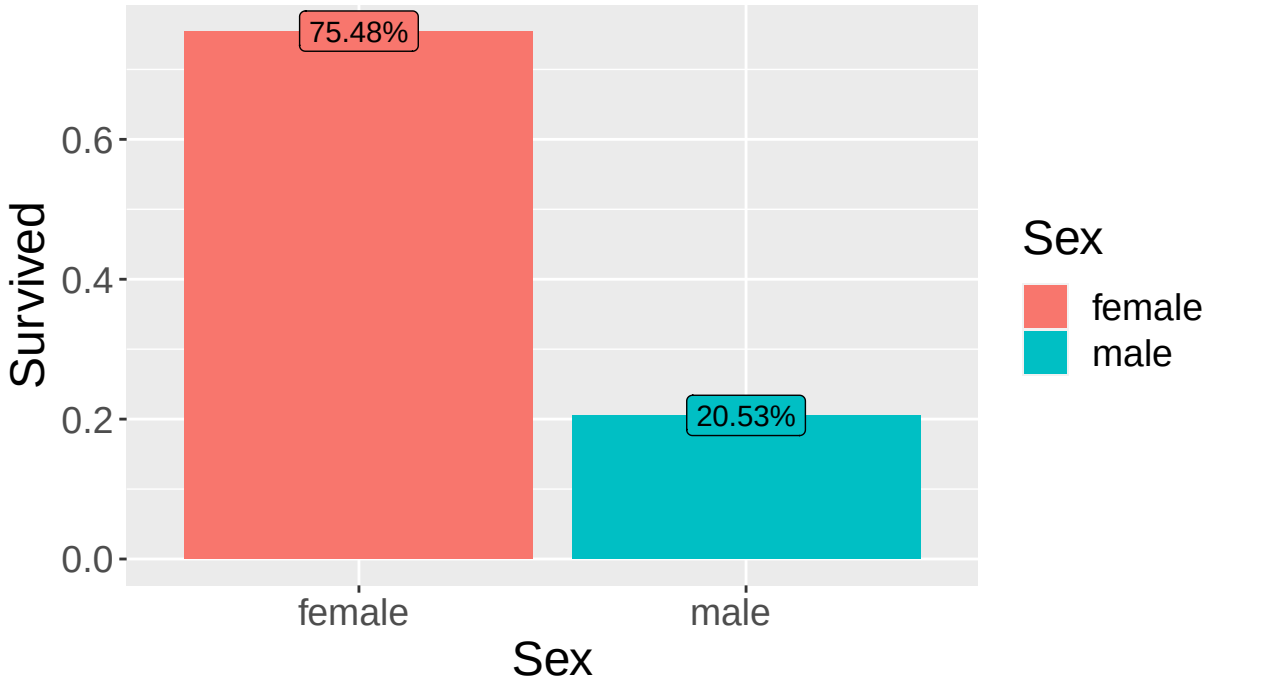


Figure 17

The survival rate per sex as shown in the barplot of Figure 17, we can notice that 75.48% of females survived while 20.53% of males survived. The larger percentage of females surviving shows the typical idea of "save the women and children" that we hear on films all the time and that we take as an attitude in catastrophic situations, it shows they were the priority and this sample backs that up.

Proportions of passengers per class

```
# proportions for pclass
Pclass <- data.frame(prop.table(table(data_n$Pclass)))
colnames(Pclass)[1] <- "Pclass"
Pclass
```

```
##   Pclass      Freq
## 1      1 0.2605042
## 2      2 0.2422969
## 3      3 0.4971989
```

```
# plotting bars
ggplot(Pclass, aes(x=Pclass, y=Freq, fill=Pclass)) +
 geom_bar(position="dodge", stat="identity") +
 geom_label(aes(y=Freq, label = scales::percent(round(Freq,4), accuracy=0.01)),
            position=position_dodge(width=0.9),
            show.legend=F)+
 theme(text = element_text(size=18))
```

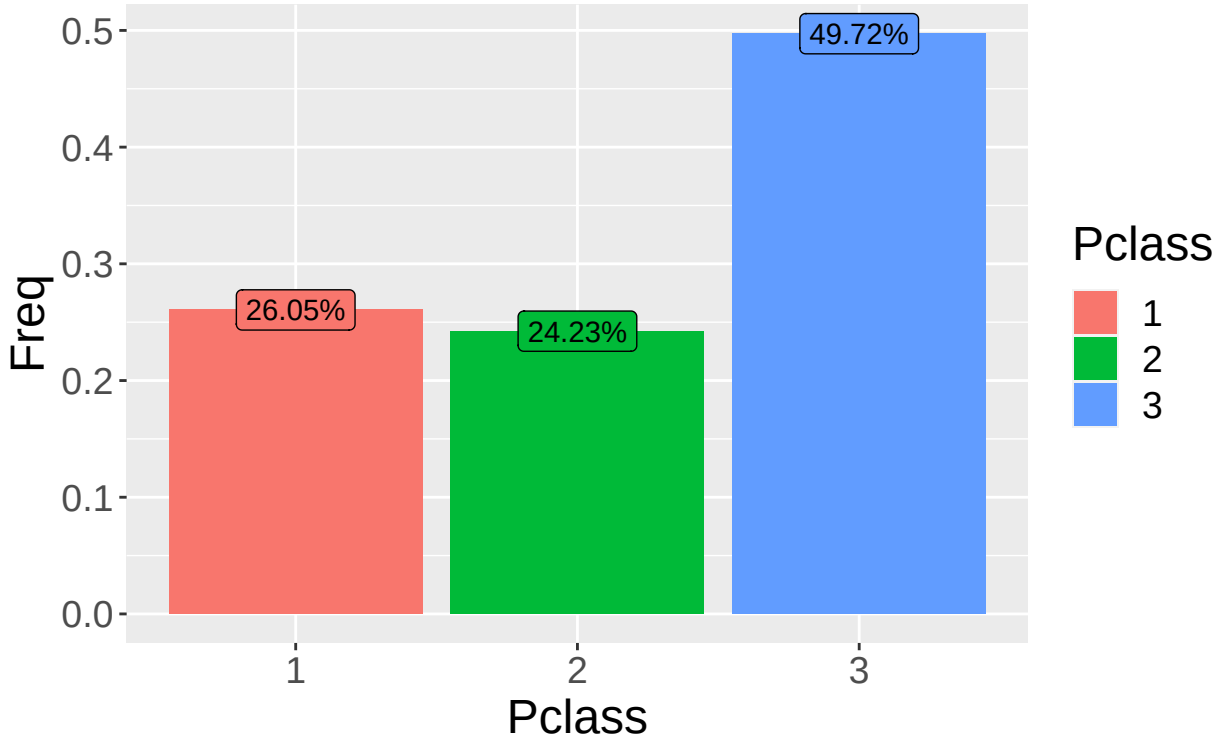


Figure 18

We can clearly see that the large majority of patients were cruising the ship in 3rd class. Passengers cruising in 3rd class make up about half the passengers on the ship, while there's a very similar amount of passengers in 1st and 2nd class. 2nd class seems to have been the least popular class in which to cruise the ship.

Important to note is that in 1st class there were not just passengers for normal 1st class but also for higher "tiers" of first class depending on their position on the boat and other metrics like suite quality.

Grouping passengers per class

```
# summarizing fare, age by mean
pclass <- data_n %>% group_by(Pclass) %>% summarize(Fare = mean(Fare), Age = mean(Age))

# melt
pclass <- melt(pclass, id.vars=c("Pclass"), measure.vars=c("Fare","Age")) %>%
  mutate(variable = as.character(variable), Pclass = as.character(Pclass))
head(pclass)
```

```
##   Pclass variable    value
## 1      1     Fare 87.96158
## 2      2     Fare 21.47156
## 3      3     Fare 13.22944
## 4      1      Age 38.23344
## 5      2      Age 29.87763
## 6      3      Age 25.14062
```

```
# plotting lines for the 3 variables selected
ggplot(pclass, aes(y=value, x=variable, fill=Pclass)) +
  geom_bar(position="dodge", stat="identity") +
  theme(text = element_text(size=18))+
  geom_label(aes(y=value, label = round(value,2), accuracy=0.01),
             position=position_dodge(width=0.9),
             show.legend=F)
```

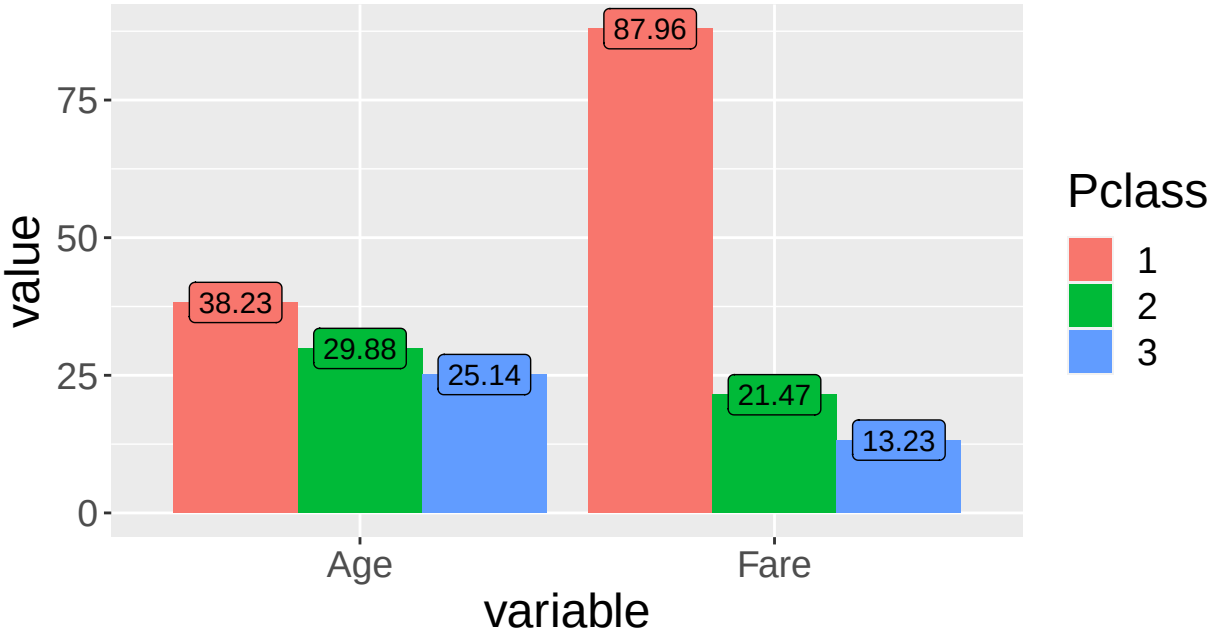


Figure 19

Here in Figure 19 we can see an interesting pattern looking at Fare and Age per passenger class. We see that the average fare for 1st class is about 4 times larger than the average fare for 2nd class and almost 7 times the average fare for a 3rd class ticket.

A similar story happens with age, we see that the mean age per class relates similarly to the Fare. Older people seem to be more willing to pay for more expensive tickets, probably because they have either a higher income or they're able to afford higher priced tickets.

Survival rate per class

```
# summarizing survived by mean
pclass <- data_n %>% group_by(Pclass) %>% summarize(Survived = mean(Survived)) %>% mutate(Pclass = as.c
head(pclass)
```

```
## # A tibble: 3 x 2
##   Pclass Survived
##   <chr>     <dbl>
## 1 1         0.656
## 2 2         0.480
## 3 3         0.239
```

```
# sex groupby survive
ggplot(pclass, aes(y=Survived, x=Pclass, fill=Pclass)) +
  geom_bar(position="dodge", stat="identity")+
  geom_label(aes(y=Survived, label = scales::percent(round(Survived,4), accuracy=0.01)),
             position=position_dodge(width=0.9),
             show.legend=F)+
  theme(text = element_text(size=18))
```

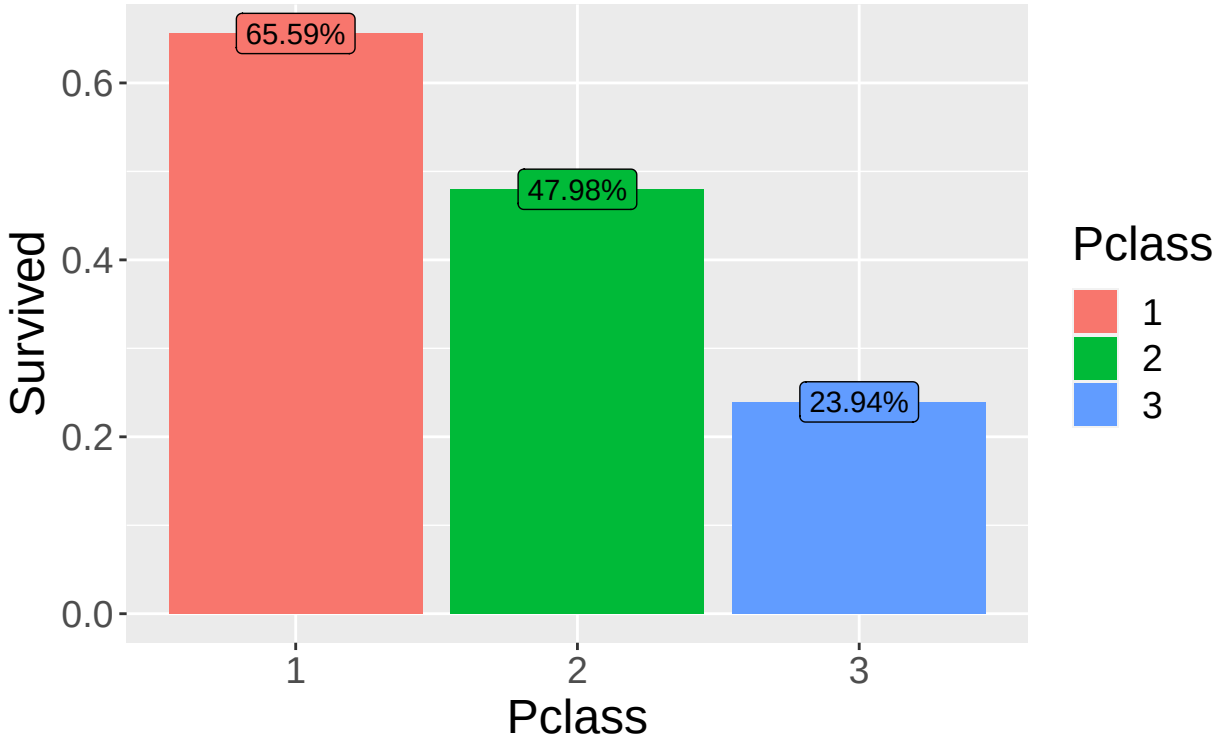


Figure 20

In Figure 20 we see the proportion of survivals per class. We notice that 1st class passengers exhibit a nearly 3x higher survival rate than 3rd class passengers. Highlighting that they were of higher priority when it comes to saving their lives.

There's probably many reasons as to why this is the case. In the instance that a passenger from first class would hold some kind of high position in a large company at the time, or perhaps hold some kind of political position, these people might be prioritized.

Proportions of passengers per port of embarkation

```
# proportions for Embarked
df <- data_n %>% filter(Embarked != '')
Embarked <- data.frame(prop.table(table(df$Embarked)))
colnames(Embarked)[1] <- "Embarked"
Embarked
```

```
##   Embarked       Freq
## 1        C 0.18258427
## 2        Q 0.03932584
## 3        S 0.77808989
```

```
# plotting bars
ggplot(Embarked, aes(x=Embarked, y=Freq, fill=Embarked)) +
 geom_bar(position="dodge", stat="identity") +
 geom_label(aes(y=Freq, label = scales::percent(round(Freq,4), accuracy=0.01)),
            position=position_dodge(width=0.9),
            show.legend=F)+
 theme(text = element_text(size=18))
```

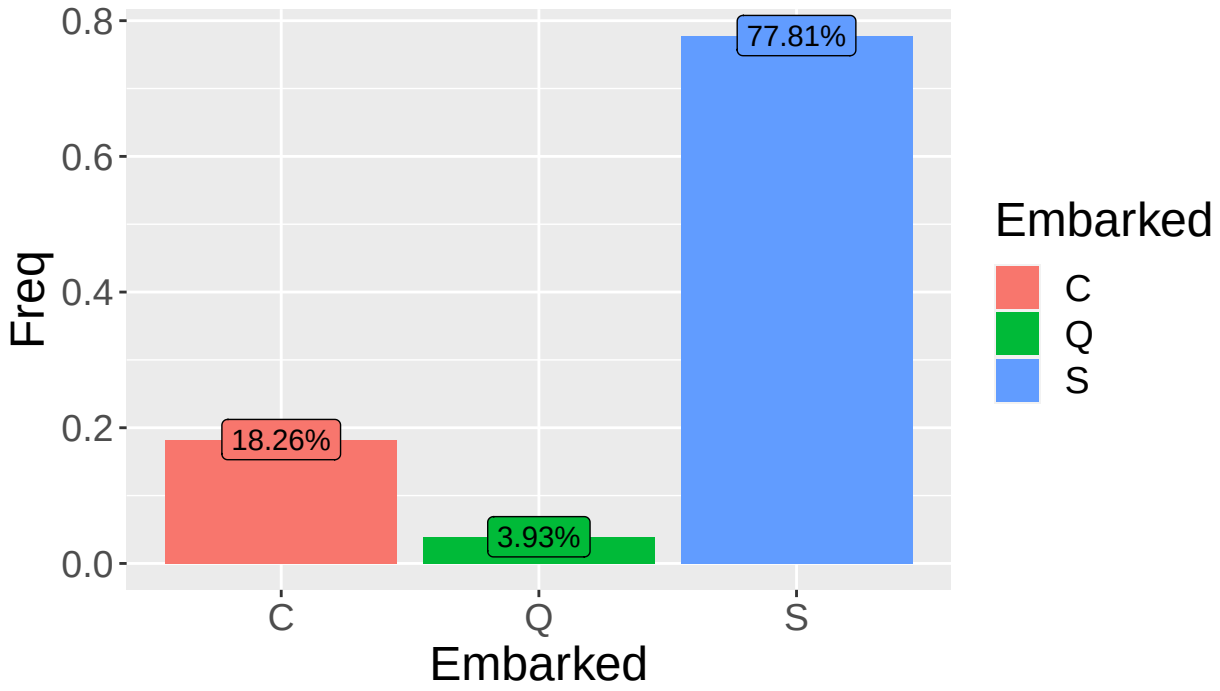


Figure 21

In Figure 21 we see the proportion of passengers embarking from each port of embarkation. Each letter stands for a port:

- C = Cherbourg
- Q = Queenstown
- S = Southampton

We can see clearly that the overwhelming majority of passengers embarked the ship through the Southampton port (77.81%), the majority of the rest through the Cherbourg port and only a few through Queenstown port.

Grouping passengers per port of embarkation

```
# summarizing fare, age by mean
embarked <- data_n %>% group_by(Embarked) %>% summarize(Fare = mean(Fare), Age = mean(Age)) %>%
 filter(Embarked != '')

# melt
embarked <- melt(embarked, id.vars=c("Embarked"), measure.vars=c("Fare","Age")) %>%
  mutate(variable = as.character(variable))
embarked
```

```
##   Embarked variable    value
## 1        C     Fare 68.29677
## 2        Q     Fare 18.26578
## 3        S     Fare 27.47628
## 4        C      Age 30.81477
## 5        Q      Age 28.08929
## 6        S      Age 29.44540
```

```
# plotting lines for the 3 variables selected
ggplot(embarked, aes(y=value, x=variable, fill=Embarked)) +
  geom_bar(position="dodge", stat="identity") +
  theme(text = element_text(size=18))+
  geom_label(aes(y=value, label = round(value,2), accuracy=0.01),
             position=position_dodge(width=0.9),
             show.legend=F)
```

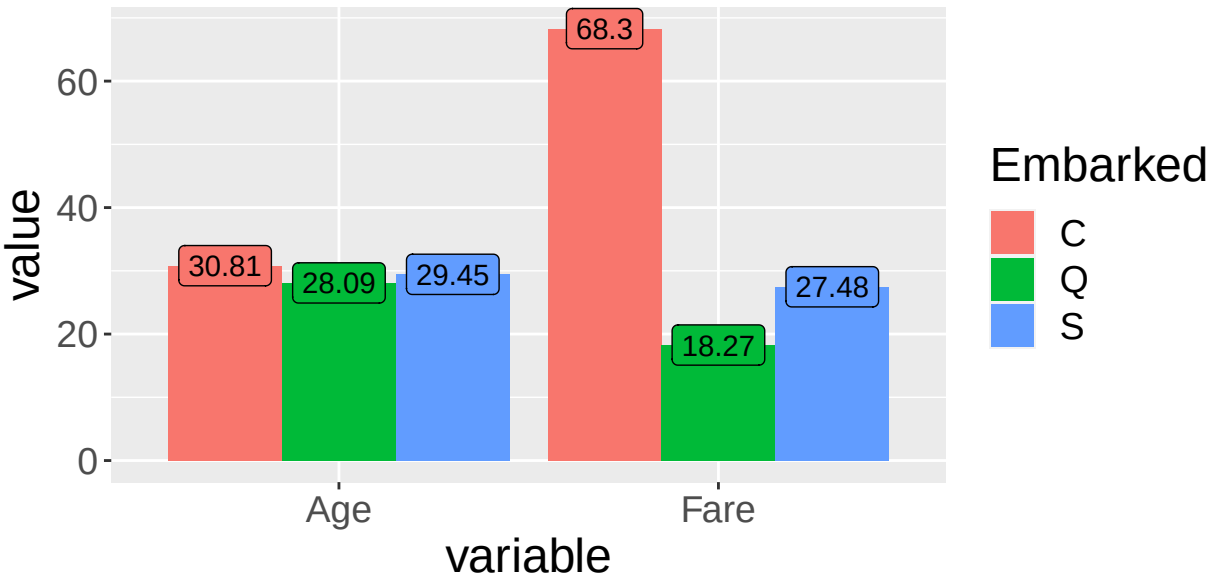


Figure 22

In Figure 22 we see that age doesn't quite play a part in which port of embarkation the passenger used to board the ship. We could say that very very slightly older people seem to board through the Cherbourg port given that older people are also willing to pay more expensive fares and Cherbourg is also the port of embarkation where higher paying passengers boarded the ship, but this might be a stretch.

However we can clearly see that passengers boarding through the Cherbourg port tend to pay higher fares. the other ports seem to be pretty even, with the Queenstown port boarding passengers with overall lower cost tickets.

Survival rate per class

```
# summarizing survived by mean
pclass <- data_n %>% group_by(Pclass) %>% summarize(Survived = mean(Survived)) %>% mutate(Pclass = as.c
head(pclass)
```

```
## # A tibble: 3 x 2
##   Pclass Survived
##   <chr>     <dbl>
## 1 1         0.656
## 2 2         0.480
## 3 3         0.239
```

```
# sex groupby survive
ggplot(pclass, aes(y=Survived, x=Pclass, fill=Pclass)) +
  geom_bar(position="dodge", stat="identity")+
  geom_label(aes(y=Survived, label = scales::percent(round(Survived,4), accuracy=0.01)),
             position=position_dodge(width=0.9),
             show.legend=F)+
  theme(text = element_text(size=18))
```

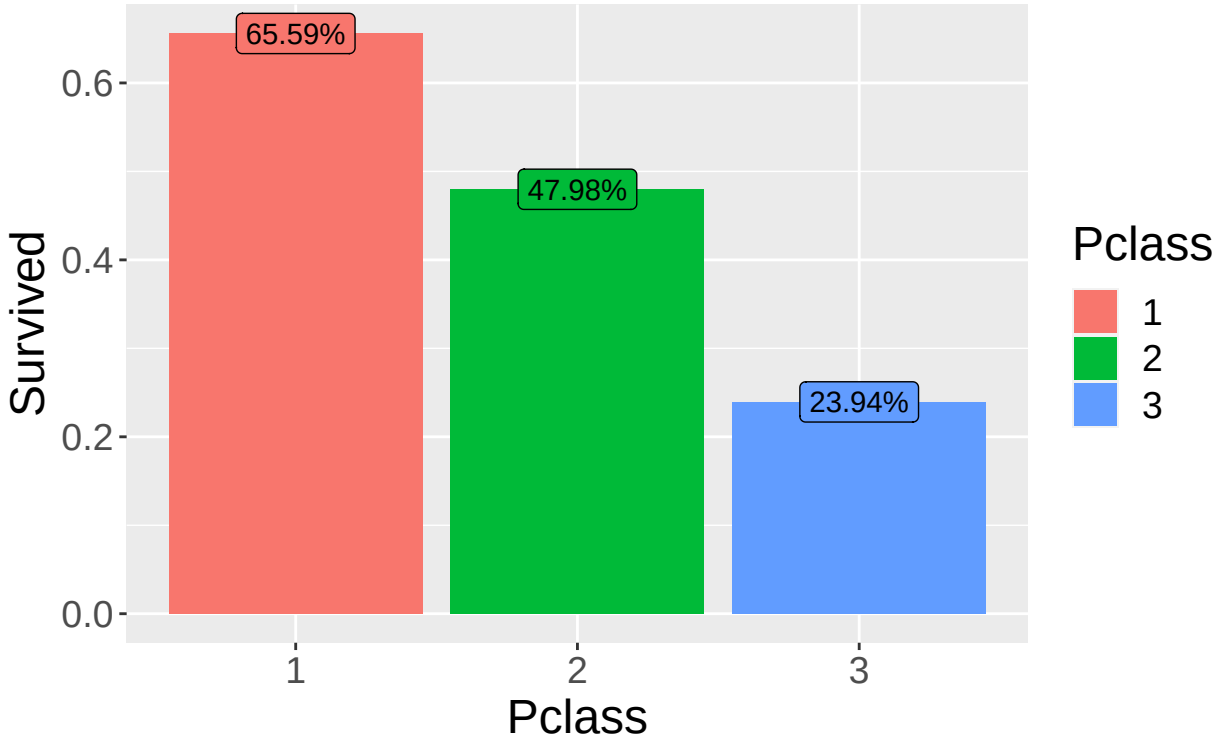


Figure 23

In Figure 23 we see the proportion of survivals per class. We notice that 1st class passengers exhibit a nearly 3x higher survival rate than 3rd class passengers. Highlighting that they were of higher priority when it comes to saving their lives.

There's probably many reasons as to why this is the case. In the instance that a passenger from first class would hold some kind of high position in a large company at the time, or perhaps hold some kind of political position, these people might be prioritized.

General passenger survival rate

```
# proportions of survivors
survived <- data.frame(prop.table(table(data_n$Survived)))
colnames(survived)[1] <- "survived"
survived
```

```
##   survived      Freq
## 1        0 0.5938375
## 2        1 0.4061625
```

```
# plotting bars
ggplot(survived, aes(x=survived, y=Freq, fill=survived)) +
 geom_bar(position="dodge", stat="identity") +
 geom_label(aes(y=Freq, label = scales::percent(round(Freq,4), accuracy=0.01)),
             position=position_dodge(width=0.9),
             show.legend=F)+
 theme(text = element_text(size=18))
```

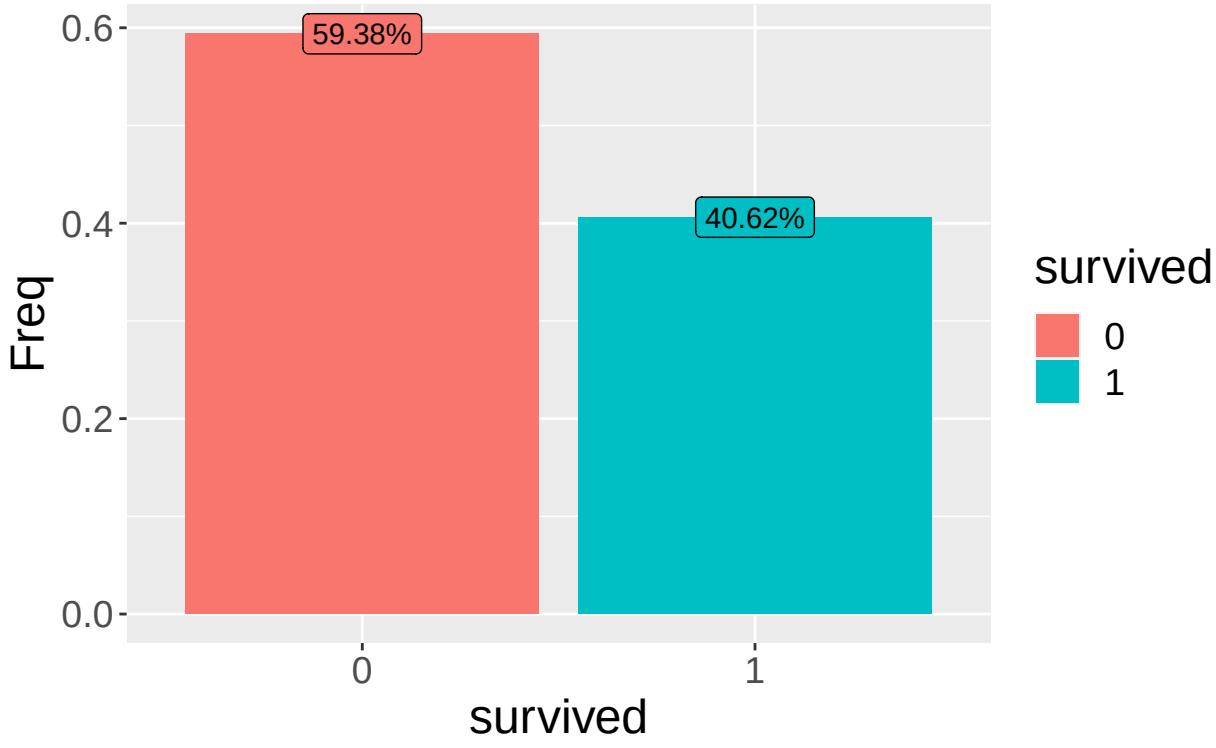


Figure 24

This variable is binary and expresses whether a passenger died or survived, where 0 = Did not survive and 1 = Survived.

In Figure 24 we can see the general survival rate for passengers. About 60% of passengers did not survive the wreckage and about 40% did. Contrary to a popular belief, we imagine most people might think that nobody survived the wreck, but even us were somewhat surprised to see that nearly half the passengers survived the wreckage.

Grouping passengers by the survival variable

```
survived <- data_n %>% group_by(Survived) %>% summarize(Fare = mean(Fare), Age = mean(Age))

# melt
survived <- melt(survived, id.vars=c("Survived"), measure.vars=c("Fare","Age")) %>%
  mutate(variable = as.character(variable), Survived = as.character(Survived))
survived
```

```
##   Survived variable    value
## 1        0     Fare 22.96546
## 2        1     Fare 51.84321
## 3        0      Age 30.62618
## 4        1      Age 28.34369
```

```
# plotting lines for the 3 variables selected
ggplot(survived, aes(y=value, x=variable, fill=Survived)) +
  geom_bar(position="dodge", stat="identity") +
  theme(text = element_text(size=18))+
  geom_label(aes(y=value, label = round(value,2), accuracy=0.01),
             position=position_dodge(width=0.9),
             show.legend=F)
```

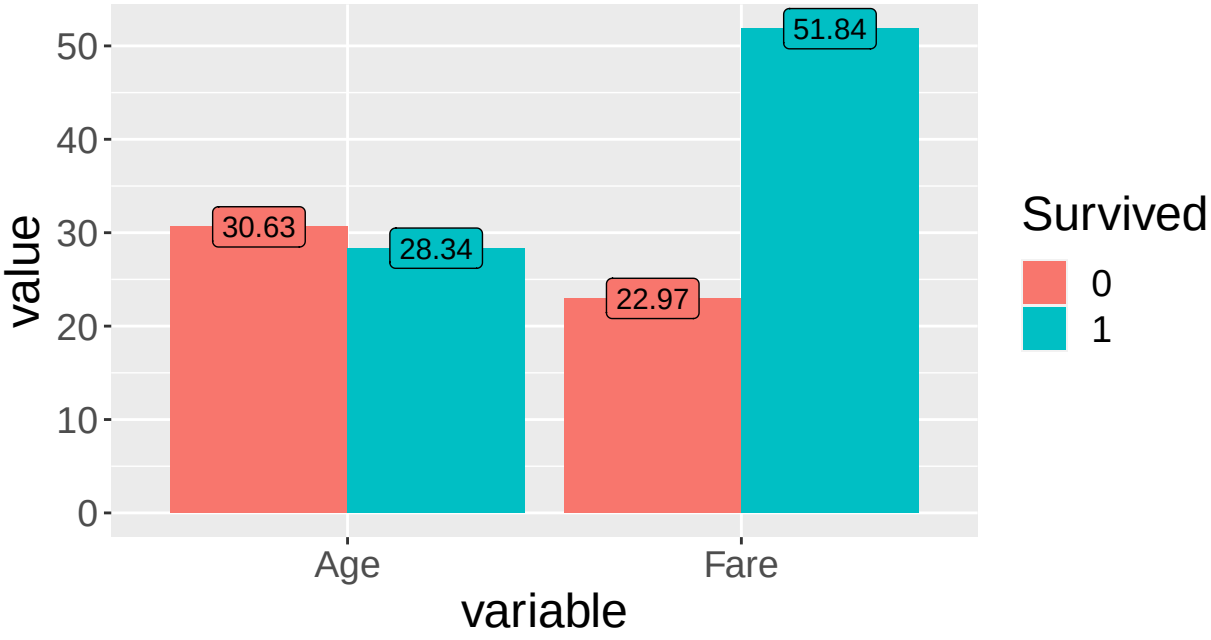


Figure 25

In Figure 25 we can see that similarly to the port of embarkation variable, the mean age for people who died or survived is quite similar. It does not seem to play a part in whether a passenger survived or died at first glance.

The mean fare, however, seems to be quite different for passengers who survived and died. This matches up well with the survived per class plot presented previously, where we notice that passengers who paid for a higher costing ticket had a significantly higher chance to survive the wreckage. Therefore the average cost of fares is 2.5x higher for passengers who survived.

Proportion of passengers per amount of siblings/spouses

```
# proportions for SibSp
SibSp <- data.frame(prop.table(table(data_n$SibSp)))
colnames(SibSp)[1] <- "SibSp"
SibSp
```

```
##   SibSp       Freq
## 1     0 0.659663866
## 2     1 0.256302521
## 3     2 0.035014006
## 4     3 0.016806723
## 5     4 0.025210084
## 6     5 0.007002801
```

```
# plotting bars
ggplot(SibSp, aes(x=SibSp, y=Freq, fill=SibSp)) +
 geom_bar(position="dodge", stat="identity") +
 geom_label(aes(y=Freq, label = scales::percent(round(Freq,4), accuracy=0.01)),
            position=position_dodge(width=0.9),
            show.legend=F)+
 theme(text = element_text(size=18))
```

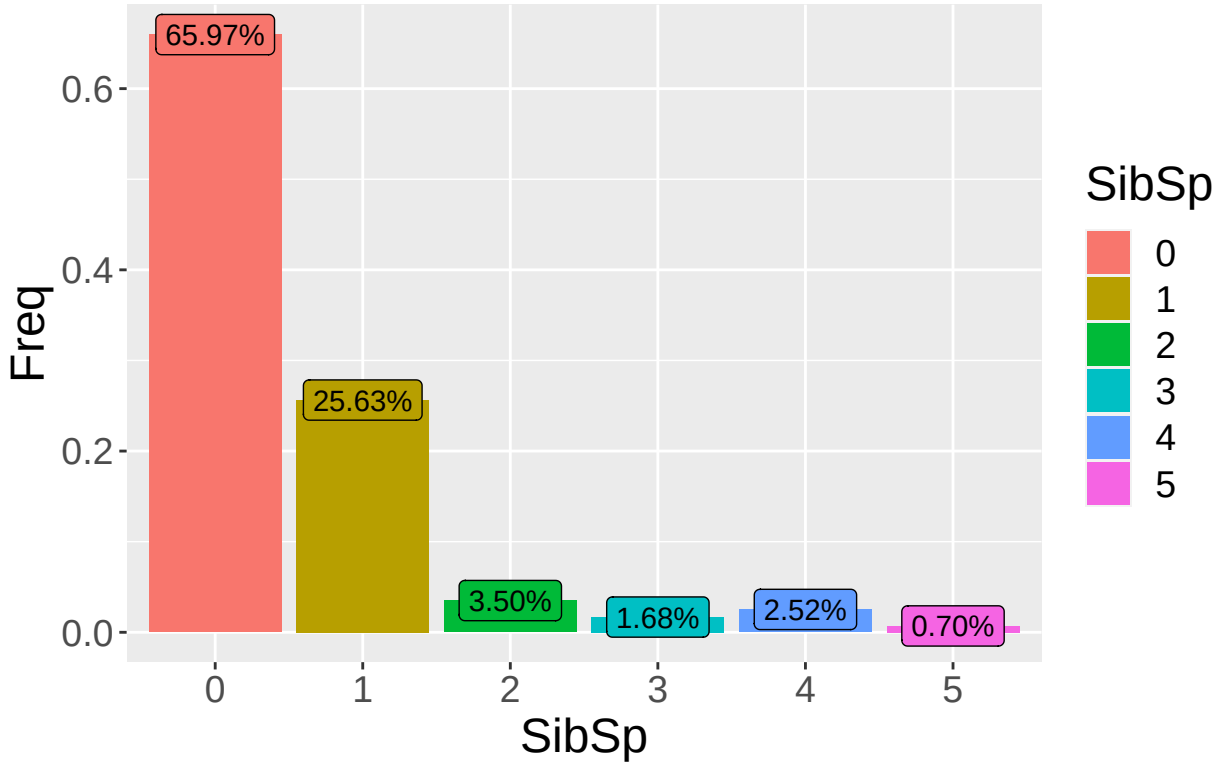


Figure 26

In Figure 26 we can see that the overwhelming majority of passengers, about 2/3 of them did not have any spouses or sibligs/step-siblings on board. Of those who did have at least one, most of them only had one. We could probably assume that most of these with a count of 1 are people traveling with their spouse. However we would have to research each individual aboard the ship.

Proportion of passengers per amount of parents/children

```
# proportions for Parch
Parch <- data.frame(prop.table(table(data_n$Parch)))
colnames(Parch)[1] <- "Parch"
Parch
```

```
##   Parch        Freq
## 1     0 0.729691877
## 2     1 0.154061625
## 3     2 0.095238095
## 4     3 0.007002801
## 5     4 0.005602241
## 6     5 0.007002801
## 7     6 0.001400560
```

```
# plotting bars
ggplot(Parch, aes(x=Parch, y=Freq, fill=Parch)) +
 geom_bar(position="dodge", stat="identity") +
 geom_label(aes(y=Freq, label = scales::percent(round(Freq,4), accuracy=0.01)),
            position=position_dodge(width=0.9),
            show.legend=F)+
 theme(text = element_text(size=18))
```

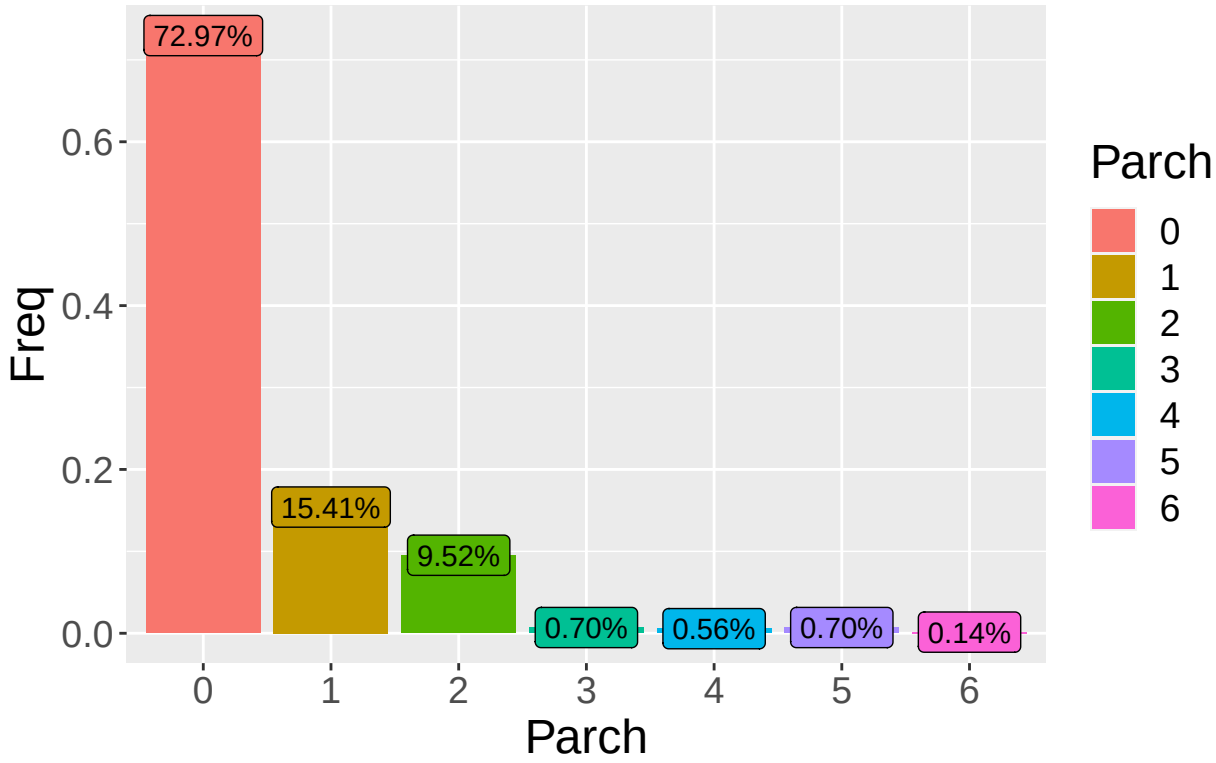


Figure 27

In Figure 27 we can see that the overwhelming majority of passengers did not have either a parent or a child (son/daughter/stepson/stepdaughter) with them on the ship. Of those that did there's a tight split between 1 and 2, possibly a good chunk of them being children with their parents on board or couples with 1-2 kids on board. The rest 3-6 are a very minor subset of the passengers.

Average amount of sublings/spouses per survival status

```
# summarizing sibsp by mean if sibsp >= 1
survived <- data_n %>% group_by(Survived) %>%
 filter(SibSp >= 1) %>%
 summarize(SibSp = mean(SibSp)) %>%
 mutate(Survived = as.character(Survived))
head(survived)
```

```
## # A tibble: 2 x 2
##   Survived SibSp
##   <chr>    <dbl>
## 1 0         1.74
## 2 1         1.24
```

```
# sex groupby survive
ggplot(survived, aes(y=SibSp, x=Survived, fill=Survived)) +
  geom_bar(position="dodge", stat="identity")+
  theme(text = element_text(size=18))+
  geom_label(aes(y=SibSp, label = round(SibSp,2)),
             position=position_dodge(width=0.9),
             show.legend=F)
```

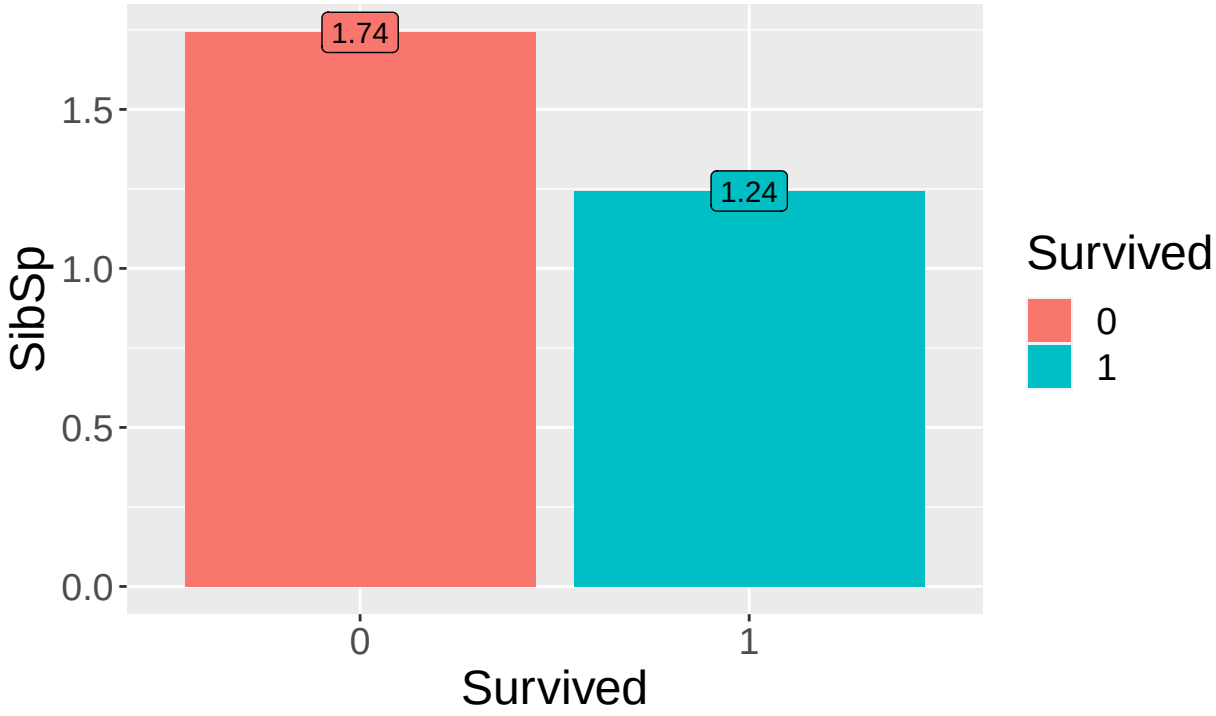


Figure 28

We thought it would be interesting to see the mean amount of siblings/spouses (as long as there's 1 or more aboard the ship) per survival status. We can see that these amount are quite similar, but interestingly, we see that the people that survived tend to have a lower amount of siblings/spouses, suggesting that perhaps those without a close family member on board had a very slight chance to survive higher than those who didn't. We can't conclusively say so, but we could infer that empirically.

Average amount of parents/children per survival status

```
# summarizing parch by mean if parch >= 1
survived <- data_n %>% group_by(Survived) %>%
 filter(Parch >= 1) %>%
 summarize(Parch = mean(Parch)) %>%
 mutate(Survived = as.character(Survived))
head(survived)
```

```
## # A tibble: 2 x 2
##   Survived Parch
##   <chr>    <dbl>
## 1 0         1.74
## 2 1         1.47
```

```
# sex groupby survive
ggplot(survived, aes(y=Parch, x=Survived, fill=Survived)) +
  geom_bar(position="dodge", stat="identity")+
  theme(text = element_text(size=18))+
  geom_label(aes(y=Parch, label = round(Parch,2)),
             position=position_dodge(width=0.9),
             show.legend=F)
```

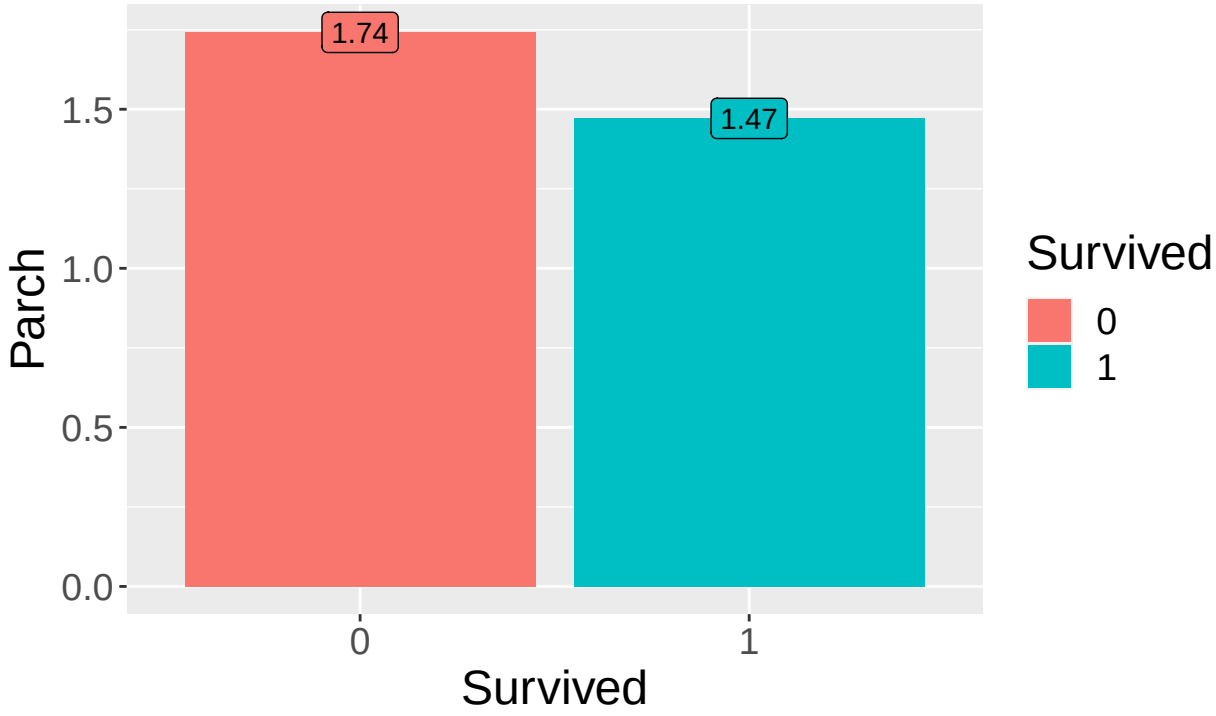


Figure 29

The same story seems to repeat for parents/children but to a much lesser extent. Here there's just no way we can conclusively say that passengers with a parent/children or more had any chance to survive whether they had more or less of such amount of family members on board, but it's somewhat similar to the numbers we saw on sibling/spouse. Where there seems to be a very subtle trend for survivors where they seem to carry less family members on board.

Part 2: Caret package

Caret Package is a comprehensive framework for building machine learning models in R. In this part we are going to use the same dataset we used in the previous part. Firstly, we are going to preprocess the dataset and will manage the NAs. Secondly, we are going to create dummy variables for the categorical variables. Thirdly, we are going create our test and train subsets. Fourthly, we are going to train different machine learning based models and we will compare them. Fifthly, we are going to use that models with our test set and we will how they perform. Lastly, we are going to ensemble that models and we will see their performance in case of we apply a Generalized Linear Model (GLM) with all of them.

The main aim of this part is to predict whether a passenger survived in the Titanic sinking, in order to do this task, we are going to use different information about the passengers and we will try to create a model which predicts as precisely as possible the survival of the passengers.

In order to this part of the project we are going to load the packages that we are going to use:

```
pacman::p_load(VIM,caret,fastDummies,dplyr,RANN,MLeval,caretEnsemble,randomForest)
```

Preprocessing

First of all, we are going to load the dataset:

```
df=read.csv("data/titanic.csv", na.strings = c("",NA))
g=read.csv("data/gender_submission.csv", na.strings = c("",NA))
for (i in g$PassengerId[1]:g$PassengerId[length(g$PassengerId)]){
  df$Survived[i]=g$Survived[i-891]
}
```

Let us see the structure of our dataset:

```
head(df)
```

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

Deleting variables and setting variable types.

In order to predict we are going to get rid of the variables that we are going to use because they do not give us a really useful information about the variable to predict. We are going to delete *Name*, *SibSp*, *Parch*, *Ticket* and *Cabin*.

```
df=df[,-(4)]
df=df[,-(6:8)]
df=df[,-(7)]
```

Our dataset has some categorical variables that are not structured that should be and we have to change their type.

```
str(df)
```

```
## 'data.frame':    1309 obs. of  7 variables:
##  $ PassengerId: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : num  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : num  3 1 3 1 3 3 1 3 3 2 ...
##  $ Sex        : chr  "male   " "female " "female " "female " ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

As it can be observed, *PClass* and *Survived* are categorized as numerical variables but they are really categorical variables. Furthermore, *Sex* variable will be converted to binary variable and then we will factorize it.

```
df$Pclass=as.character(df$Pclass)
df$Sex<-replace(df$Sex,df$Sex=='male',0)
df$Sex<-replace(df$Sex,df$Sex=='female',1)
df$Survived<-replace(df$Survived,df$Survived==1,'YES')
df$Survived<-replace(df$Survived,df$Survived==0,'NO')
df$Survived=as.factor(df$Survived)
df$Sex=as.factor(df$Sex)

str(df)
```

```
## 'data.frame':    1309 obs. of  7 variables:
##  $ PassengerId: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : Factor w/ 2 levels "NO","YES": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass     : chr  "3" "1" "3" "1" ...
##  $ 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 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

Handling with NAs

As it can be observed we have some Nas in our dataset and we cannot use them. Due to that fact, we have to convert that NAs to some values and in order to do that, we are going to predict that variable using k-nearest neighbors (knn) algorithm. This algorithm predicts the value of NAs, in the case of a continuous variable, the algorithm selects the k-nearest neighbors with euclidean distance and takes the median; otherwise, if it is a categorical variable it takes the mode of that k-nearest neighbors. In our case, we have applied this algorithm for *Age*, *Embarked* and *Fare* continuous variables.

```
df=kNN(df,variable=c("Embarked","Age","Fare"))
df=df[,-(8:10)]
anyNA(df)
```

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

We can now observe how we do not have any missing value.

Creating Dummy variables

Dummy variables, also known as One Hot Encoding, is a tool for converting categorical variables into numeric in order for it to be used by the machine learning algorithms. In addition, we must have into account that just replacing the categories with a number may not be meaningful especially if there is no intrinsic ordering amongst the categories. So, the thing that we have to do is create the categorical variable into so many binary variables as the variable has.

We will do this process in *Embarked* and *PClass* categorical variables.

```
df=dummy_cols(df,select_columns = c('Pclass','Embarked'))
df=df[,-c(3,7)]
```

Once we have created the dummy variables of them, we can delete the original variable from the dataset.

Data partition

The last step of the preprocessing will be to create two partition for the machine learning algorithms. We will divide our data in two subsets, **Train** and **Test**, this partitition will follow a distribution of 80-20. The aim of this partition is to use the train dataset to fit the models and the predict the values of test dataset. In this way, we will be able to observe the real performance of our machine learning algorithms.

```
n=createDataPartition(df$PassengerId,p=0.8,list=FALSE)
train=df[n,]
train=df[,-1]
test=df[-n,]
```

Then, we have to define which variable will be the variable that we will predict using diverse algorithms, the variable that we will try to predict will be *Survived* and for that objective we will use *Sex*, *Age*, *Fare*, *Pclass* and *Embarked*.

```
x_train=train[,(2:10)]
y_train=train$Survived
```

Analysis of the predictor variables

Recursive feature elimination

Most machine learning algorithms are able to determine what features are important to predict the Y. But, we might be careful about which variables are significant.

In order to select the important features, we are going to use the recursive feature elimination (RFE). The process follows three steps, firstly, we build a ML model on a training dataset and estimate the feature importance on the test dataset. Secondly, keeping priority to the most important variables, iterate through by building models of given subset sizes, that is, subgroups of most important predictors determined from step 1. Ranking of the predictors is recalculated in each iteration. Lastly, the model performances are compared across different subset sizes to arrive at the optimal number and list of final predictors.

To implement this process we will use repeated cross validation with 5 repetition for every subset analysed. In addtion, the metric that we will use is the accuracy.

```
set.seed(613)

options(warn = -1)
subsets = 1:9
ctrl = rfeControl(functions = rfFuncs, method = "repeatedcv", repeats = 5, verbose = FALSE)
lmProfile = rfe(x = x_train, y = y_train, sizes = subsets, rfeControl = ctrl,metric="Accuracy")
lmProfile
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold, repeated 5 times)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.8548 0.6875    0.02668 0.05764
##          2   0.8531 0.6849    0.02781 0.05996
##          3   0.8545 0.6876    0.02705 0.05848
##          4   0.8663 0.7115    0.02916 0.06355
##          5   0.8651 0.7088    0.02846 0.06200
##          6   0.8646 0.7076    0.02768 0.06054
##          7   0.8669 0.7118    0.02772 0.06102
##          8   0.8667 0.7113    0.02691 0.05934
##          9   0.8716 0.7213    0.02938 0.06498        *
##
## The top 5 variables (out of 9):
##    Sex, Age, Fare, Pclass_3, Pclass_2
```

Looking to the results we can observed that the best results are obtained with 9 variables, but the rfe() function returns us that the best option is to select only 5 variables. However, as we have a small sample for ML problem, we will use all the variables in order to improve our prediction. In addition, as the sample is relatively small, selecting all the variables will not have a real impact on the execution time.

Training and Tuning models.

In this section, we will use different machine learning techniques in order to train our model and then, predict in test dataset.

For all the models that we will apply in this section, we will use the repeated cross validation with 5 repetitions in order to fit our model as precisely as possible.

```
ctrl= trainControl(method='repeatedcv', number=5, savePredictions='final', classProbs=T, summaryFunctio
```

Multivariate Adaptive Regression (MARS)

This technique is a non-parametric regression technique and can be seen as an extension of linear models that automatically models non-linearities and tries to fit our dataset by the combination of linear models.

```
train_mars=train(Survived~.,data=train,method="earth",metric='accuracy',trControl=ctrl)
train_mars
```

```
## Multivariate Adaptive Regression Spline
##
## 1309 samples
##    9 predictor
##    2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 1047, 1047, 1047, 1047, 1048
## Resampling results across tuning parameters:
##
##   nprune  ROC        Sens       Spec
##    2      0.8399481  0.9006135  0.7792826
##    6      0.8965455  0.8981595  0.7914657
##   11      0.8869195  0.8981595  0.7954855
##
## Tuning parameter 'degree' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were nprune = 6 and degree = 1.
```

Random Forests (RF)

This technique is an ensemble learning method that constructs a multitude of decision trees at training time and outputting the class that is the mode of the classes in the case of classification or mean prediction of the individual trees in the case of regression.

```
train_rf=train(Survived~.,data=train,method="rf",metric='ROC',trControl=ctrl)
train_rf
```

```
## Random Forest
##
## 1309 samples
```

```
##    9 predictor
##    2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 1047, 1047, 1048, 1047, 1047
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##   2     0.9120759  0.9263804  0.7691198
##   5     0.9121402  0.9202454  0.7975881
##   9     0.9035393  0.8981595  0.7874459
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.
```

Support vector machine (SVM)

This technique is a supervised learning model with associated learning algorithms that analyze data. The aim of this technique is that a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, in that way we can use them for classification or regression.

```
train_svmRadial=train(Survived~.,data=train,method="svmRadial",metric='ROC',trControl=ctrl)
train_svmRadial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1309 samples
##    9 predictor
##    2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 1047, 1048, 1047, 1047, 1047
## Resampling results across tuning parameters:
##
##   C     ROC        Sens       Spec
##   0.25  0.8868158  0.8981595  0.7814265
##   0.50  0.8938516  0.9006135  0.7834673
##   1.00  0.8931221  0.9006135  0.7894867
##
## Tuning parameter 'sigma' was held constant at a value of 0.1901948
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1901948 and C = 0.5.
```

Adaptive Boosting (AdaBoost)

This technique is a supervised learning technique that reduces the bias and variance of the model. This technique uses classification trees and then, it reduced it bias and variance.

```
train_ada=train(Survived~.,data=train,method='adaboost',metric='ROC',trControl=ctrl)
train_ada
```

```
## AdaBoost Classification Trees
##
## 1309 samples
##    9 predictor
##    2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 1048, 1047, 1047, 1047, 1047
## Resampling results across tuning parameters:
##
##   nIter  method         ROC        Sens       Spec
##    50    Adaboost.M1    0.9115082  0.8883436  0.8118738
##    50    Real adaboost  0.7539640  0.8858896  0.7875283
##   100    Adaboost.M1    0.9105522  0.8797546  0.8098330
##   100    Real adaboost  0.7405842  0.8834356  0.7915481
##   150    Adaboost.M1    0.9106758  0.8723926  0.8199546
##   150    Real adaboost  0.7076836  0.8809816  0.7956504
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were nIter = 50 and method = Adaboost.M1.
```

XGBosst booster: DART

This technique uses classification trees and then, it adopts a dropout method from neural networks in order to boost regression trees.

```
train_xgb=train(Survived~.,data=train,method='xgbDART',metric='ROC',trControl=ctrl)
```

ROC curves

ROC curves are a graphical plot that illustrate the diagnostic ability of a binary classifier system as its discrimination thresold is varied. ROC curves are created by plotting the sensitivity against the specificity at various threshold settings. The models will be better, the greater the area that holds the right part of the curve is.

```
evalm(train_mars,showplots = FALSE,title = "MARS")$roc
```

```
## ***MLeval: Machine Learning Model Evaluation***
## Input: caret train function object
## Averaging probs.
## Group 1 type: repeatedcv
## Observations: 1309
## Number of groups: 1
## Observations per group: 1309
## Positive: YES
## Negative: NO
```

```
## Group: Group 1
## Positive: 494
## Negative: 815
## ***Performance Metrics***
## Group 1 Optimal Informedness = 0.697754650902859
## Group 1 AUC-ROC = 0.89
```

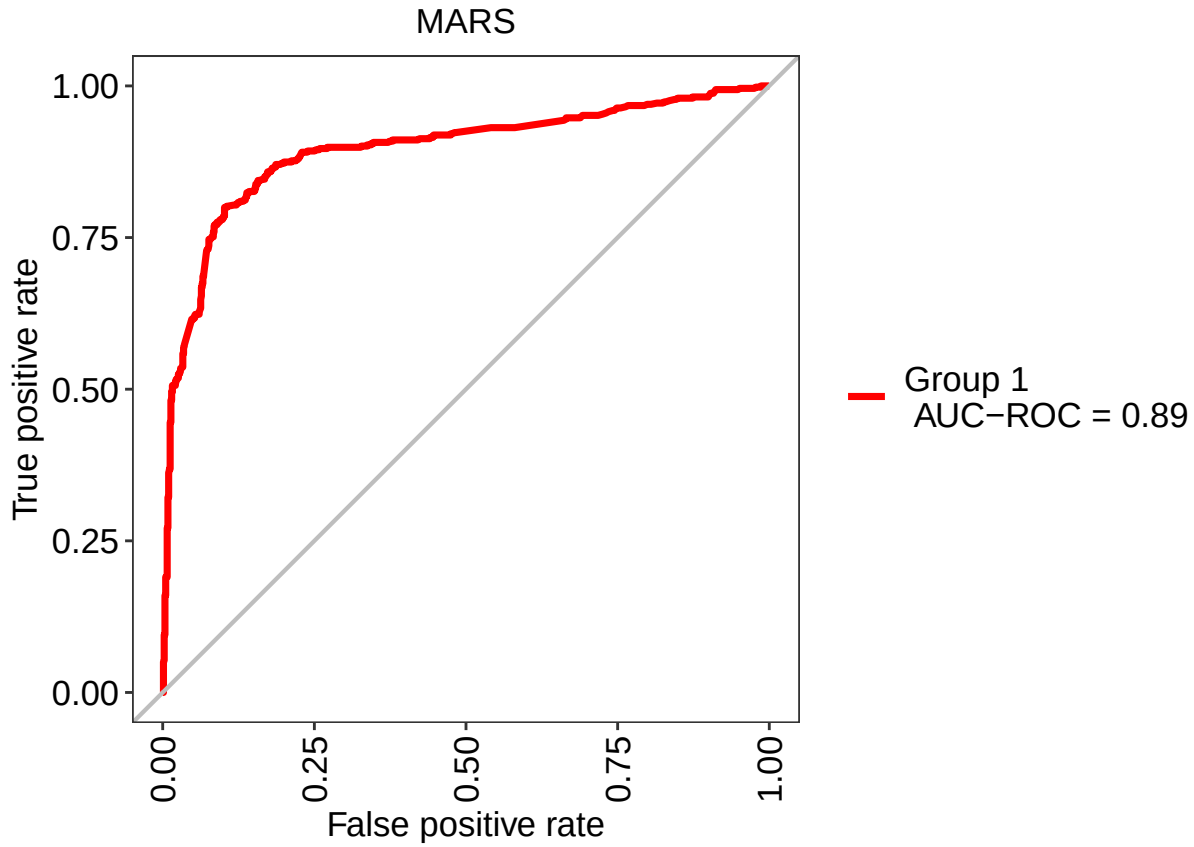


Figure 30

```
evalm(train_rf,showplots = FALSE,title = "Random Forests")$roc
```

```
## ***MLeval: Machine Learning Model Evaluation***
## Input: caret train function object
## Averaging probs.
## Group 1 type: repeatedcv
## Observations: 1309
## Number of groups: 1
## Observations per group: 1309
## Positive: YES
## Negative: NO
## Group: Group 1
```

```
## Positive: 494
## Negative: 815
## ***Performance Metrics***
## Group 1 Optimal Informedness = 0.71701895134249
## Group 1 AUC-ROC = 0.91
```

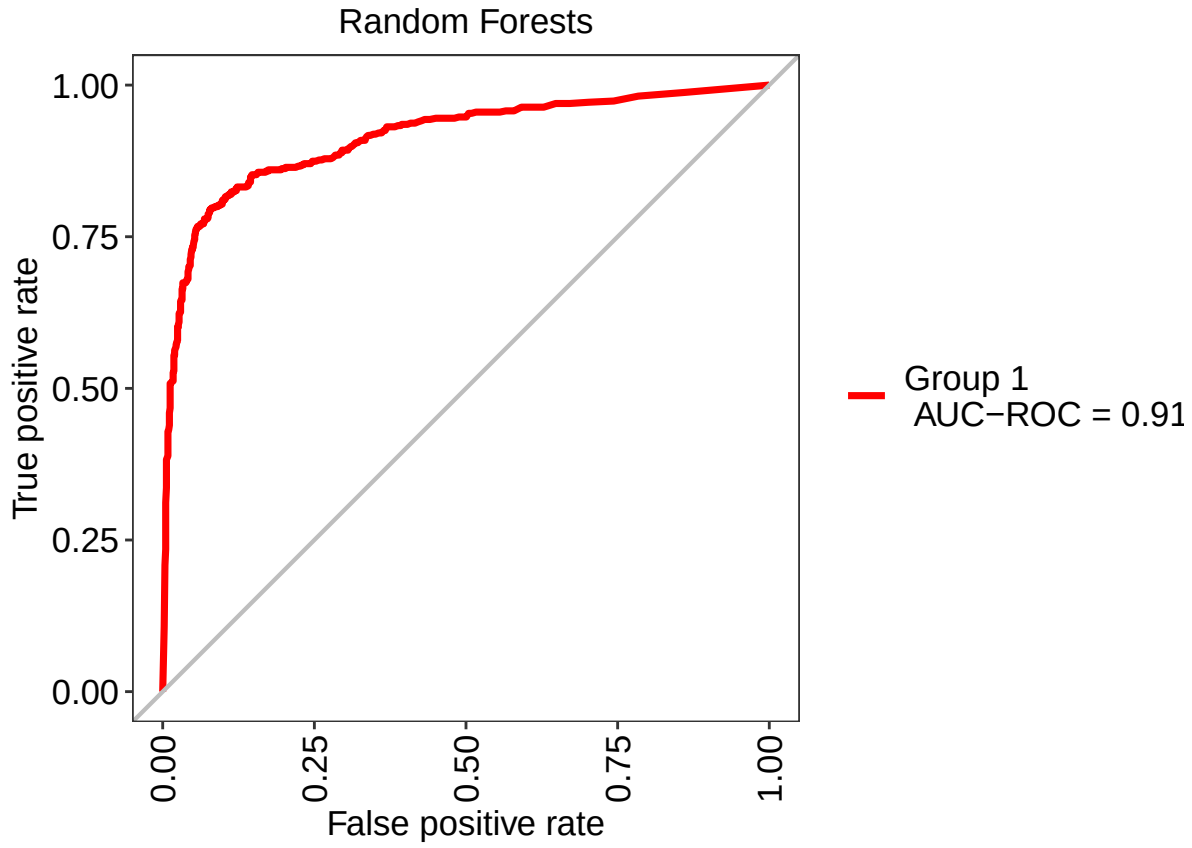


Figure 31

```
evalm(train_svmRadial,showplots = FALSE,title = "SVM")$roc
```

```
## ***MLeval: Machine Learning Model Evaluation***
## Input: caret train function object
## Averaging probs.
## Group 1 type: repeatedcv
## Observations: 1309
## Number of groups: 1
## Observations per group: 1309
## Positive: YES
## Negative: NO
## Group: Group 1
## Positive: 494
```

```
## Negative: 815
## ***Performance Metrics***
## Group 1 Optimal Informedness = 0.705422120662676
## Group 1 AUC-ROC = 0.89
```

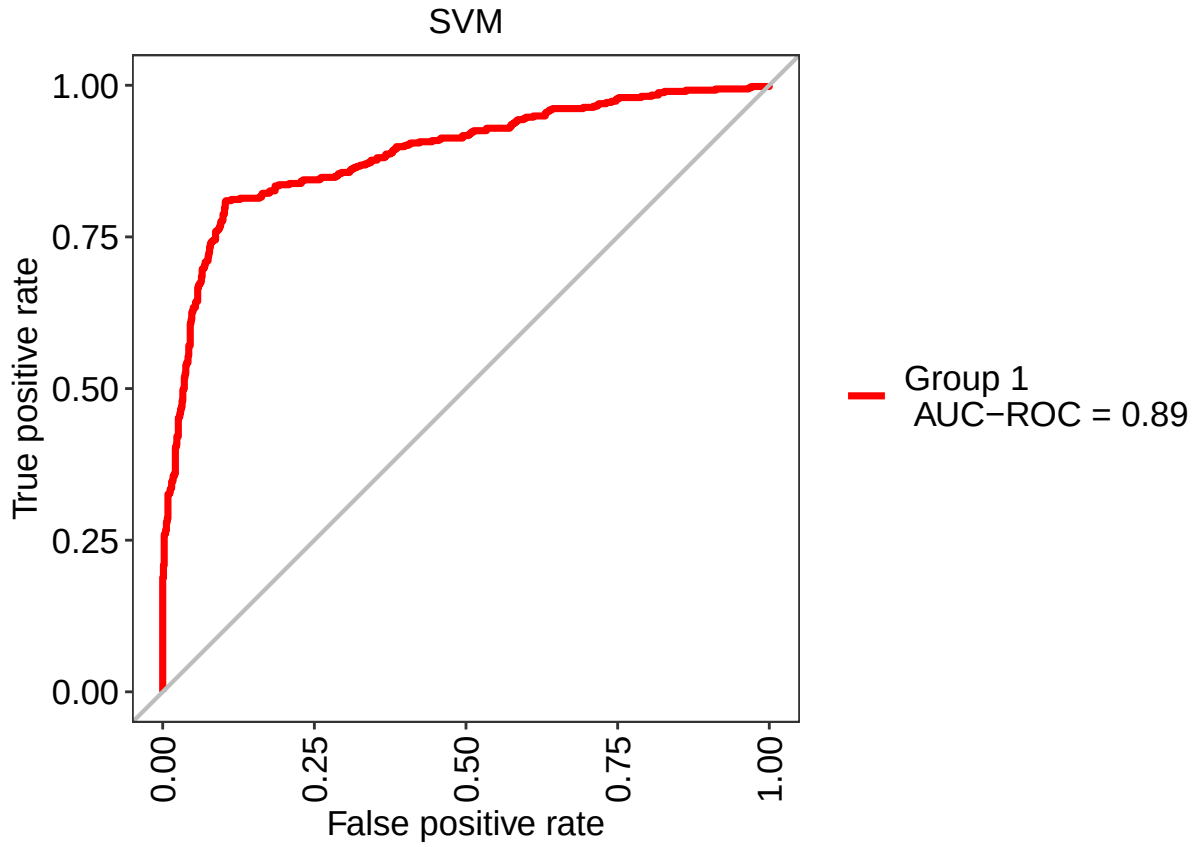


Figure 32

```
evalm(train_ada,showplots = FALSE,title = "AdaBoost")$roc
```

```
## ***MLeval: Machine Learning Model Evaluation***
## Input: caret train function object
## Averaging probs.
## Group 1 type: repeatedcv
## Observations: 1309
## Number of groups: 1
## Observations per group: 1309
## Positive: YES
## Negative: NO
## Group: Group 1
## Positive: 494
## Negative: 815
```

```
## ***Performance Metrics***
## Group 1 Optimal Informedness = 0.710821887186111
## Group 1 AUC-ROC = 0.91
```

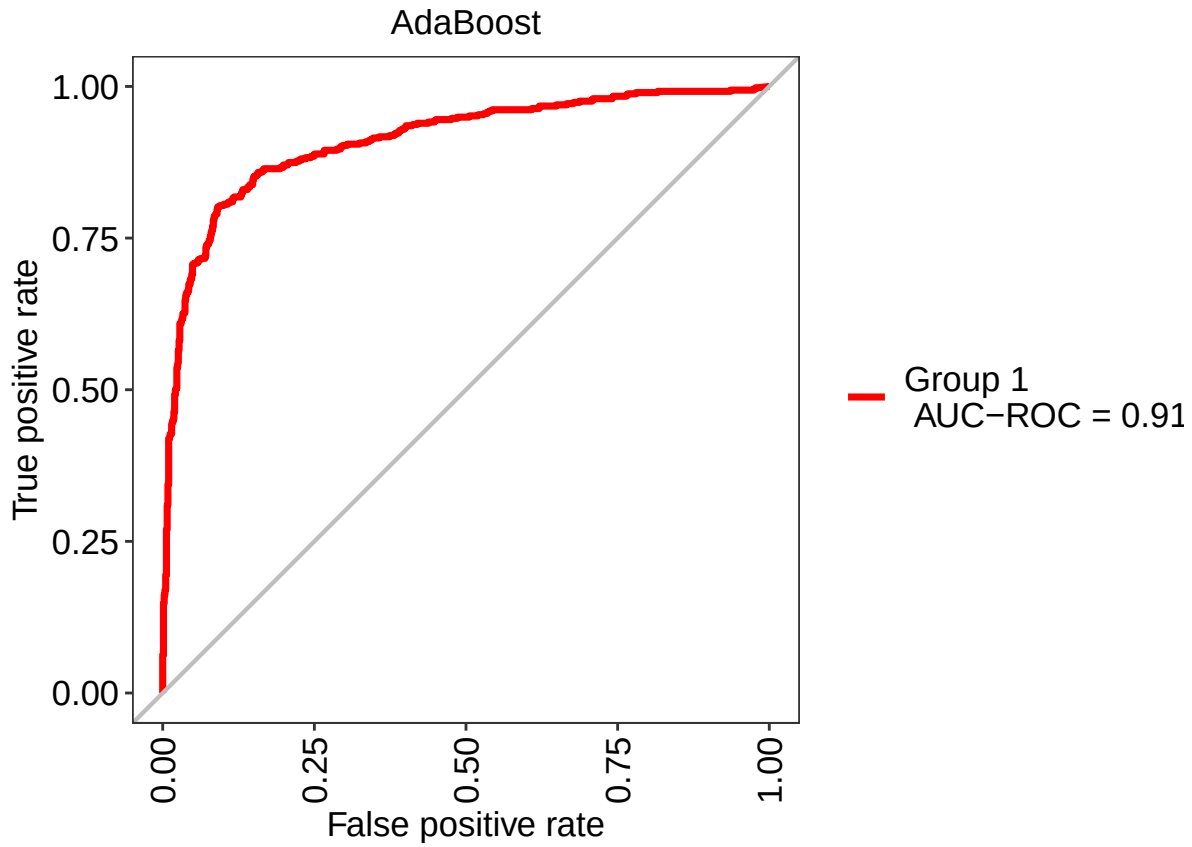


Figure 33

```
evalm(train_xgb,showplots = FALSE,title = "xgbDart")$roc
```

```
## ***MLeval: Machine Learning Model Evaluation***
## Input: caret train function object
## Averaging probs.
## Group 1 type: repeatedcv
## Observations: 1309
## Number of groups: 1
## Observations per group: 1309
## Positive: YES
## Negative: NO
## Group: Group 1
## Positive: 494
## Negative: 815
## ***Performance Metrics***
```

```
## Group 1 Optimal Informedness = 0.728610814435806
## Group 1 AUC-ROC = 0.92
```

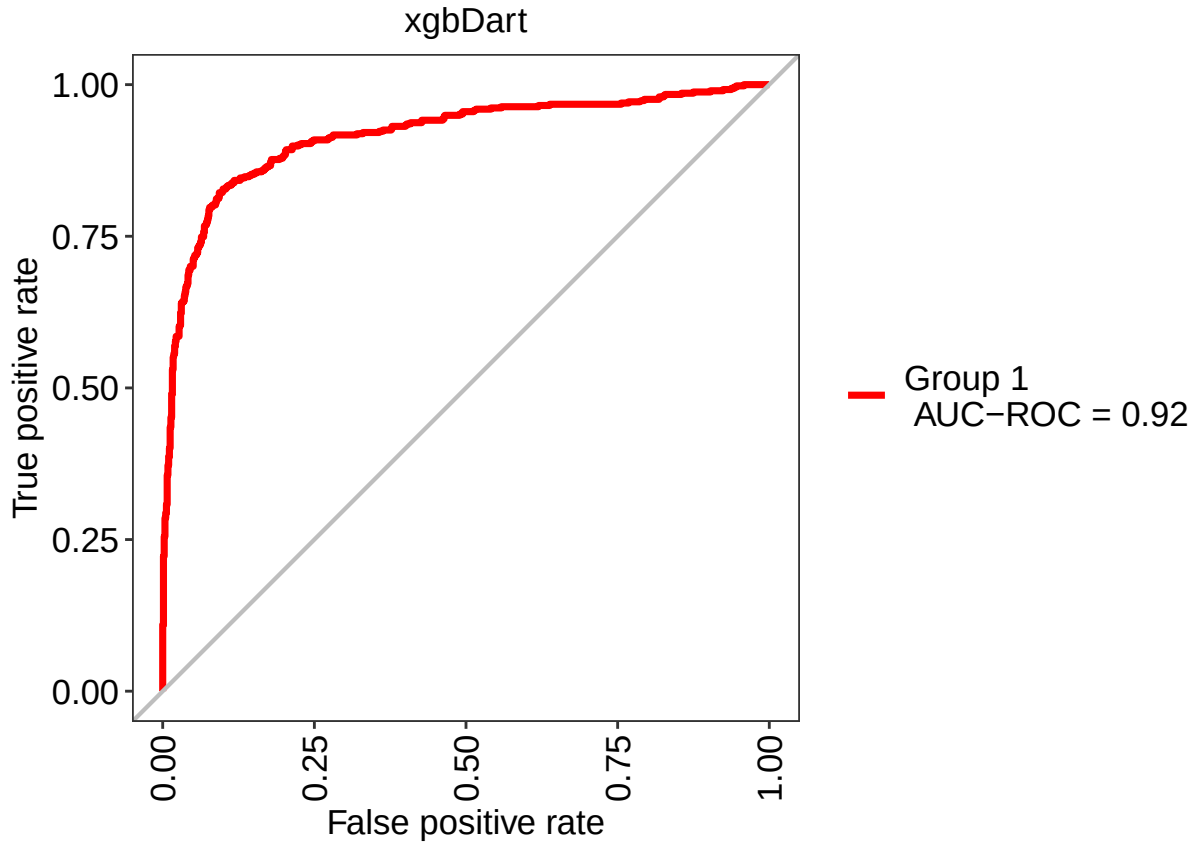


Figure 34

Looking to the curves, we can observe how the worse methods are **SVM** and **MARS**, and the rest of method obtain very similar results, being **Random Forests** and **xgbDART** the best, having an area of 0.92.

Prediction and Confusion matrices

We are going to predict the test results using the techniques below and will calculate the confusion matrices and some statistics. Furthermore, we are going to interpret the results obtained.

```
p_mars=predict(train_mars,test)
confusionMatrix(reference=test$Survived,data=p_mars,mode="everything",positive = "YES")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  153  19
##        YES   7  81
##
##                Accuracy : 0.9
##                  95% CI : (0.8569, 0.9336)
##     No Information Rate : 0.6154
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7839
##
##  Mcnemar's Test P-Value : 0.03098
##
##             Sensitivity : 0.8100
##             Specificity : 0.9563
##          Pos Pred Value : 0.9205
##          Neg Pred Value : 0.8895
##               Precision : 0.9205
##                  Recall : 0.8100
##                      F1 : 0.8617
##              Prevalence : 0.3846
##          Detection Rate : 0.3115
##    Detection Prevalence : 0.3385
##       Balanced Accuracy : 0.8831
##
##        'Positive' Class : YES
##
```

```
p_rf=predict(train_rf,test)
confusionMatrix(reference=test$Survived,data=p_rf,mode="everything",positive = "YES")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  158  10
##        YES   2  90
##
##                Accuracy : 0.9538
##                  95% CI : (0.9208, 0.9759)
##     No Information Rate : 0.6154
##     P-Value [Acc > NIR] : < 2e-16
##
```

```
##                   Kappa : 0.901
##
##  Mcnemar's Test P-Value : 0.04331
##
##             Sensitivity : 0.9000
##             Specificity : 0.9875
##          Pos Pred Value : 0.9783
##          Neg Pred Value : 0.9405
##               Precision : 0.9783
##                  Recall : 0.9000
##                      F1 : 0.9375
##              Prevalence : 0.3846
##          Detection Rate : 0.3462
##    Detection Prevalence : 0.3538
##       Balanced Accuracy : 0.9438
##
##        'Positive' Class : YES
##
```

```
p_svmRadial=predict(train_svmRadial,test)
confusionMatrix(reference=test$Survived,data=p_svmRadial,mode="everything",positive = "YES")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  153  18
##        YES   7  82
##
##                Accuracy : 0.9038
##                  95% CI : (0.8613, 0.9368)
##     No Information Rate : 0.6154
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7926
##
##  Mcnemar's Test P-Value : 0.0455
##
##             Sensitivity : 0.8200
##             Specificity : 0.9563
##          Pos Pred Value : 0.9213
##          Neg Pred Value : 0.8947
##               Precision : 0.9213
##                  Recall : 0.8200
##                      F1 : 0.8677
##              Prevalence : 0.3846
##          Detection Rate : 0.3154
##    Detection Prevalence : 0.3423
##       Balanced Accuracy : 0.8881
##
##        'Positive' Class : YES
##
```

```
p_ada=predict(train_ada,test)
confusionMatrix(reference=test$Survived,data=p_ada,mode="everything",positive = "YES")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  160   5
##        YES   0  95
##
##                Accuracy : 0.9808
##                  95% CI : (0.9557, 0.9937)
##     No Information Rate : 0.6154
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.959
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 0.9500
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9697
##               Precision : 1.0000
##                  Recall : 0.9500
##                      F1 : 0.9744
##              Prevalence : 0.3846
##          Detection Rate : 0.3654
##    Detection Prevalence : 0.3654
##       Balanced Accuracy : 0.9750
##
##        'Positive' Class : YES
##
```

```
p_xgb=predict(train_xgb,test)
confusionMatrix(reference=test$Survived,data=p_xgb,mode="everything",positive = "YES")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  155  16
##        YES   5  84
##
##                Accuracy : 0.9192
##                  95% CI : (0.8792, 0.9493)
##     No Information Rate : 0.6154
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8258
##
##  Mcnemar's Test P-Value : 0.0291
##
##             Sensitivity : 0.8400
##             Specificity : 0.9688
##          Pos Pred Value : 0.9438
##          Neg Pred Value : 0.9064
##               Precision : 0.9438
```

```
##                  Recall : 0.8400
##                      F1 : 0.8889
##              Prevalence : 0.3846
##          Detection Rate : 0.3231
##    Detection Prevalence : 0.3423
##       Balanced Accuracy : 0.9044
##
##        'Positive' Class : YES
##
```

Firstly, we can assure that the methods with the worst results are the one with the worst ROC curves, **MARS** and **SVM**. However, **xgbDART** does not obtain the results we could expect from it, it is quite a rare situation because we do not have overfitting for sure, as this method contains several dropouts. In fact, the best two methods are **Random Forests** and **AdaBoost**, both having and outstanding performance in specificity values. In addition, as we have a balanced dataset, we can look to the accuracy as a good estimator of the performance of the model.

Ensembling model

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. Ensembles combine multiple hypotheses to form a better hypothesis. The term ensemble is usually reserved for methods that generate multiple hypotheses using the same base learner.

```
ctrlE = trainControl(method="repeatedcv", number=10, repeats=5, savePredictions=TRUE, classProbs=TRUE)
algorithmList = c('rf', 'adaboost', 'earth', 'xgbDART', 'svmRadial')
models = caretList(Survived~.,data=train, trControl=ctrlE, methodList=algorithmList)
results = resamples(models)
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: rf, adaboost, earth, xgbDART, svmRadial
## Number of resamples: 50
##
## Accuracy
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf        0.8091603 0.8549618 0.8697299 0.8719507 0.8908217 0.9469697    0
## adaboost  0.7615385 0.8310922 0.8473282 0.8481065 0.8699794 0.9160305    0
## earth     0.7938931 0.8399983 0.8615385 0.8587996 0.8709663 0.9236641    0
## xgbDART   0.8230769 0.8625954 0.8778626 0.8765250 0.8861468 0.9236641    0
## svmRadial 0.7938931 0.8409091 0.8582501 0.8583369 0.8702290 0.9312977    0
##
## Kappa
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf        0.5907785 0.6814358 0.7204655 0.7228351 0.7655637 0.8859822    0
## adaboost  0.5100900 0.6387461 0.6834182 0.6773373 0.7208430 0.8199425    0
## earth     0.5580407 0.6590656 0.6984937 0.6970789 0.7271151 0.8369836    0
## xgbDART   0.6155824 0.6984899 0.7344784 0.7328315 0.7576683 0.8356336    0
## svmRadial 0.5580407 0.6581004 0.6934187 0.6959698 0.7278633 0.8526802    0
```

Once, we have created the ensembling, we can then, by Generalized Linear Model, add all the five models and create a new model that theoretically will perform better.

```
stackControl = trainControl(method="repeatedcv", number=10, repeats=5, savePredictions=TRUE, classProbs=
stack_glm = caretStack(models, method="glm", metric="ROC", trControl=stackControl)
stack_glm
```

```
## A glm ensemble of 5 base models: rf, adaboost, earth, xgbDART, svmRadial
##
## Ensemble results:
## Generalized Linear Model
##
## 6545 samples
##    5 predictor
##    2 classes: 'NO', 'YES'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 5890, 5891, 5890, 5891, 5890, 5891, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8733694  0.7224137
```

Once we have created the model, we will predict the test results and then, we will see how the model performed.

```
p_com=predict(stack_glm,test)
confusionMatrix(reference=test$Survived,data=p_com,mode="everything",positive = "YES")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  159  15
##        YES   1  85
##
##                Accuracy : 0.9385
##                  95% CI : (0.902, 0.9644)
##     No Information Rate : 0.6154
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8665
##
##  Mcnemar's Test P-Value : 0.001154
##
##             Sensitivity : 0.8500
##             Specificity : 0.9938
##          Pos Pred Value : 0.9884
##          Neg Pred Value : 0.9138
##               Precision : 0.9884
##                  Recall : 0.8500
##                      F1 : 0.9140
##              Prevalence : 0.3846
##          Detection Rate : 0.3269
##    Detection Prevalence : 0.3308
##       Balanced Accuracy : 0.9219
##
##        'Positive' Class : YES
##
```

Looking to the results, we could think at first glance that this model is worst than, for instance, **adaBoost**, as we see that the accuracy is smaller. That is a fact, but we must gave into account that this model is a better model overall, even if it does not get the best accuracy value.

Conclusion

In this part of the project we have learned how to do machine learning algorithms, and prepare our data in order to obtain the best results we can.

Firstly, we have seen how important is to prepocess all the data in order to enhance the performance of our algorithm. In this process, we need to select the variables that we will use for the prediction, convert the categorical with plenty variables into dummy variables and divide the dataset into train and test set. Furthermore, if we have missing values we will need to deal with them, and in this project, we did that predicting them with k-nearest neighbors algorithm.

Secondly, any machine learning problem will need to fit the model, in this process we will need to determine the way that will do the training. In our case we used the repeated cross validation in every step of the fitting. Furthermore, we have seen five different techniques to do the prediction: **MARS**, **Random Forests**, **SVM**, **AdaBoost** and **xgbDart**. Overall, five of them obtain a good performance, even if there are some of them that are better.

Lastly, we did an ensemble of these five techniques, as we may imagine this ensemble stacked by a generalized linear model should perform better that the techniques did on their own. But, we have seen that this is not what happened in the reality.