Titanic Survival Part 1: EDA in R

Marcelo Sanches

July 4, 2019

Contents

[LIST CONTENTS]

Summary

[EXPLAIN PROJECT]

In Part 1 of the Titanic Survival project I conduct **Exploratory Data Analisys (EDA)** of the Kaggle Titanic train dataset in RStudio, through summary tables and visualizations, performing minor pre-processing as needed.

In Part 2 of the project I intend to write a Python script that succintly performs all essential pre-processing steps, emulating a production environment.

---

1. Basic EDA

First we load the training data. We will not look at the test data until it is time to test; failing to do so would consist in *data snooping*. In the spirit of the Titanic Kaggle kernel, I added the Kaggle Titanic datasets a level up on an `input/` directory.

```
# Load training set
rm(list=ls())
train <- read.csv("../input/train.csv", na.strings="")
```

Basic EDA consists of looking at the dataset's structure, summary, top rows, and bottom rows:

```
str(train)
```

```
## '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       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 559 520 629 417 58
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 86 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 147 levels "A10","A14","A16",..: NA 82 NA 56 NA NA 130 NA NA NA ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
```

There are 891 passengers and 11 attributes (PassengerId is just an index) for each passenger.

- **Survived**, coded as integer, is a binary indicator and the target outcome or dependent variable we are predicting.
- **Pclass**, coded as intger, is an ordinal variable for the passenger class; we could change it to a factor variable with three levels (1st, 2nd, and 3rd class).
- **Name** has one level per passenger and a common approach would be to extract the title ('Mr.','Mrs.') from it so as to cull down the number of levels; we will also consider name length.
- **Sex** is coded as categorical and could be transformed into an indicator such as `is_male`.
- **Age** is numeric and seems to have missing values.
- **SibSp** is ordinal (# of siblings or spouses) and could be converted from integer to factor.
- **Parch** is ordinal (# of parents or children) and could be converted to factor; it could be combined with SibSp to represent "# of relatives".
- **Ticket** is categorical with 681 levels and needs to be culled down if it provides useful information.
- **Fare** is numerical and a proxy for wealth or social status.
- **Cabin** is categorical with 147 levels and needs to be culled down, perhaps by extracting the deck.
- **Embarked** is categorical with 3 levels, C for Cherbourg, Q for Queenstown, S for Southhampton, the three ports of embarkation.

A summary would be more useful if certain variables were converted to factor right away. Because `PassengerId` is just an index we can drop it, but first check that is has no duplicates or non-stepwise values ("trust but verify!"):

```r
# trust but verify
sum(duplicated(train$PassengerId)) == 0
```

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

```r
sum(train$PassengerId == 1:891) == 891
```

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

```r
# drop PassengerId and convert appropriate variables to factor
train$PassengerId <- NULL
train$SurvivedFac <- factor(train$Survived) # for plotting as categorical outcome
train$SurvivedNum <- as.numeric(train$Survived) # for plotting as numerical outome
train$Survived <- NULL # drop original
train$Pclass <- factor(train$Pclass)
train$SibSp <- factor(train$SibSp)
train$Parch <- factor(train$Parch)
# summary
summary(train)
```

```
##  Pclass                                     Name         Sex
##  1:216   Abbing, Mr. Anthony                  :  1   female:314
##  2:184   Abbott, Mr. Rossmore Edward          :  1   male  :577
##  3:491   Abbott, Mrs. Stanton (Rosa Hunt)     :  1
##          Abelson, Mr. Samuel                  :  1
##          Abelson, Mrs. Samuel (Hannah Wizosky):  1
##          Adahl, Mr. Mauritz Nils Martin       :  1
##          (Other)                              :885
##       Age         SibSp   Parch        Ticket         Fare
##  Min.   : 0.42   0:608   0:678   1601    :  7   Min.   :  0.00
##  1st Qu.:20.12   1:209   1:118   347082  :  7   1st Qu.:  7.91
##  Median :28.00   2: 28   2: 80   CA. 2343:  7   Median : 14.45
##  Mean   :29.70   3: 16   3:  5   3101295 :  6   Mean   : 32.20
##  3rd Qu.:38.00   4: 18   4:  4   347088  :  6   3rd Qu.: 31.00
```

```
##  Max.   :80.00   5:  5   5:  5   CA 2144 :  6   Max.   :512.33
##  NA's   :177     8:  7   6:  1   (Other) :852
##          Cabin     Embarked   SurvivedFac  SurvivedNum
##  B96 B98    :  4   C    :168   0:549       Min.   :0.0000
##  C23 C25 C27:  4   Q    : 77   1:342       1st Qu.:0.0000
##  G6         :  4   S    :644               Median :0.0000
##  C22 C26    :  3   NA's:  2               Mean   :0.3838
##  D          :  3                           3rd Qu.:1.0000
##  (Other)    :186                           Max.   :1.0000
##  NA's       :687
```

Note the high number of missing values in `Age` and `Cabin`. The 2 NAs in `Embarked` can be easily dealt with by filling in with the most common port of embarkation. Class representation in `Survived`,`Pclass`, and `Sex` appear to be relatively trouble free, yet in `SibSp`, `Parch`, and `Embarked` we see more skewed distributions. `Fare` is skewed as well.

```r
head(train)
```

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

```r
tail(train)
```

```
##     Pclass                                     Name    Sex Age SibSp Parch
## 886      3     Rice, Mrs. William (Margaret Norton) female  39     0     5
## 887      2                    Montvila, Rev. Juozas   male  27     0     0
## 888      1             Graham, Miss. Margaret Edith female  19     0     0
## 889      3 Johnston, Miss. Catherine Helen "Carrie" female  NA     1     2
## 890      1                    Behr, Mr. Karl Howell   male  26     0     0
## 891      3                      Dooley, Mr. Patrick   male  32     0     0
##         Ticket   Fare Cabin Embarked SurvivedFac SurvivedNum
## 886     382652 29.125  <NA>        Q           0           0
## 887     211536 13.000  <NA>        S           0           0
## 888     112053 30.000   B42        S           1           1
## 889 W./C. 6607 23.450  <NA>        S           0           0
## 890     111369 30.000  C148        C           1           1
## 891     370376  7.750  <NA>        Q           0           0
```

EDA also consists of plotting, but before that we should cleanup a bit our messy data, since plotting `Name` and `Ticket` as they are would be useless, for example.

---

2. Preliminary Data Cleaning

- Cleaning up Name

A `Title` attribute would be much more useful. First we use regex to extract these titles:

```
train$Title <- vector("character",length=nrow(train))
for (i in 1:nrow(train)) {
    x <- as.character(train$Name[i])
    m <- regexec(",(\\s+\\w+)+\\.", x)
    train$Title[i] <- unlist(strsplit(unlist(regmatches(x,m))," "))[2]
}
# fixing a particular case
train$Title[train$Title == "the"] <- "the Countess"
# looking at unique titles
unique(train$Title)
```

```
##  [1] "Mr."           "Mrs."          "Miss."         "Master."
##  [5] "Don."          "Rev."          "Dr."           "Mme."
##  [9] "Ms."           "Major."        "Lady."         "Sir."
## [13] "Mlle."         "Col."          "Capt."         "the Countess"
## [17] "Jonkheer."
```

There are 17 levels which seem unnecessary as some of these titles are specific and rare, so we can bin them into two rare categories, one for males and one for females, since the probability of survival is highly dependent on gender.

Note that some decisions are simplifications, there is a female doctor (Dr. Alice Leader, who survived) yet I assigned 'Dr.' to the mostly rare male title category.

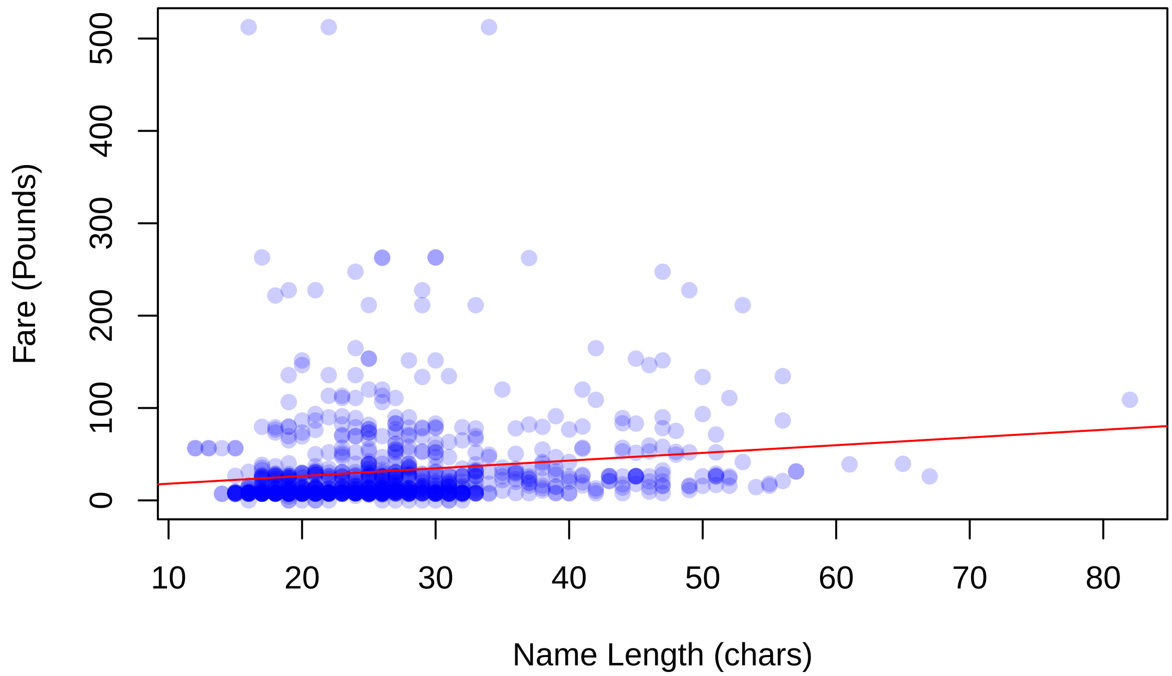
```
rareMale <- c("Don.","Rev.","Dr.","Major.","Master.", "Sir.","Col.","Capt.","Jonkheer.")
rareFemale <- c("Mme.","Ms.","Lady.","Mlle.","the Countess")
for (i in 1:nrow(train)) {
    train$Title[i] <- ifelse(train$Title[i] %in% rareMale, "rareMale",
                      ifelse(train$Title[i] %in% rareFemale, "rareFemale", train$Title[i]))
}
# unique titles
unique(train$Title)
```

```
## [1] "Mr."         "Mrs."        "Miss."       "rareMale"   "rareFemale"
```

Before dropping name entirely, we can informally test a common assumption that the length of a name is associated positively with higher socio-economic status and therefore survivability.

```
# create NameLength attribute
train$NameLength <- vector("numeric", nrow(train))
for (i in 1:nrow(train)) {
    train$NameLength[i] <- nchar(as.character(train$Name)[i])
}
# see whether this attribute has any meat to it
plot(train$NameLength, train$Fare,
    pch=19, col=rgb(0,0,1,alpha=0.2),
```

```
    xlab="Name Length (chars)",
    ylab="Fare (Pounds)")
abline(lm(train$Fare ~ train$NameLength), col="red")
```



While the evidence isn't particularly strong, we might as well keep `NameLength` in the mix just to see whether it improves modeling later on. Now we can drop `Name`.

```
# dropping Name
train$Name <- NULL
```

- Cleaning up Ticket

```
train$Ticket <- as.character(train$Ticket)
# removing ending digits
train$TicketClean <- vector("character", nrow(train))
for (i in 1:nrow(train)) {
    pattern <- "[0-9]+$"
    m <- regexec(pattern, train$Ticket[i])
    digits <- regmatches(train$Ticket[i], m)
    train$TicketClean[i] <- trimws(sub(digits, "", train$Ticket[i]), which="right")
}
# remove periods, fwd slashes, blanks, iteratively
for (i in 1:nrow(train))    train$TicketClean[i] <- gsub("[.]", "", train$TicketClean[i])
for (i in 1:nrow(train))    train$TicketClean[i] <- gsub("[/]", "", train$TicketClean[i])
for (i in 1:nrow(train))    train$TicketClean[i] <- gsub("[\\s]", "", train$TicketClean[i])
# cleanup manually to bin similar entries
for (i in 1:nrow(train)) {
```

```r
train$TicketClean[i] <- ifelse(train$TicketClean[i] == "STONO2", "SOTONO2",
                         ifelse(train$TicketClean[i] == "STONO 2", "SOTONO2",
                         ifelse(train$TicketClean[i] == "SCAH Bale", "SCAH",
                         ifelse(train$TicketClean[i] == "SCPari", "SCPARIS",
                         ifelse(train$TicketClean[i] == "", "Other", train$TicketClean[i])))))
}
# look at distribution within levels
table(train$TicketClean)
```

```
##
##      A4      A5      AS       C      CA CASOTON      Fa      FC     FCC
##       7      21       1       5      41       1       1       1       5
##    LINE   Other      PC      PP     PPP      SC    SCA4    SCAH    SCOW
##       4     661      60       3       2       1       1       3       1
## SCPARIS     SOC     SOP    SOPP SOTONO2 SOTONOQ      SP    SWPP      WC
##      11       6       1       3      20      15       1       2      10
##     WEP
##       3
```

Since the “Other” level is so overbearingly dominant and there are too many unrepresented categories, it is unlikely that a lot of useful information can be gathered from `Ticket` so we just drop this attribute altogether.

```r
train$Ticket <- NULL
train$TicketClean <- NULL
# let's look at the data now to get our bearings
head(train)
```

```
##   Pclass    Sex Age SibSp Parch    Fare Cabin Embarked SurvivedFac
## 1      3   male  22     1     0  7.2500  <NA>        S           0
## 2      1 female  38     1     0 71.2833   C85        C           1
## 3      3 female  26     0     0  7.9250  <NA>        S           1
## 4      1 female  35     1     0 53.1000  C123        S           1
## 5      3   male  35     0     0  8.0500  <NA>        S           0
## 6      3   male  NA     0     0  8.4583  <NA>        Q           0
##   SurvivedNum Title NameLength
## 1           0   Mr.         23
## 2           1  Mrs.         51
## 3           1 Miss.         22
## 4           1  Mrs.         44
## 5           0   Mr.         24
## 6           0   Mr.         16
```

- Cleaning up Cabin

Cabin has 687 NAs and 147 levels yet cabin locations might be important in determining survivability, since the accident happened late at night when people were mostly in their cabins, and lower-letter cabins were near the deck while higher-letter cabins were near the keel where the ship hit the iceberg.

```r
train$CabinClean <- vector("character", nrow(train))
for (i in 1:nrow(train)) {
  # ID digits and white space
    pattern <- "[0-9]*|\\s"
  # reduce to only first letter given multiple cabins
    train$CabinClean[i] <- substr(gsub(pattern, "", train$Cabin[i]),1,1)
    # bin letters higher than F to the F category
    high_cabins <- toupper(letters[letters >"f"])
    if (train$CabinClean[i] %in% high_cabins) train$CabinClean[i] <- "F"
```

```r
}
table(train$CabinClean)
```

```
##
##  A  B  C  D  E  F
## 15 47 59 33 32 18
```

```r
# replace old Cabin
train$Cabin <- factor(train$CabinClean)
train$CabinClean <- NULL
summary(train$Cabin)
```

```
##    A    B    C    D    E    F NA's
##   15   47   59   33   32   18  687
```

We now have good representations in all cabins and not too many levels but still a lot of missing values, we'll deal with those later as needed.

---

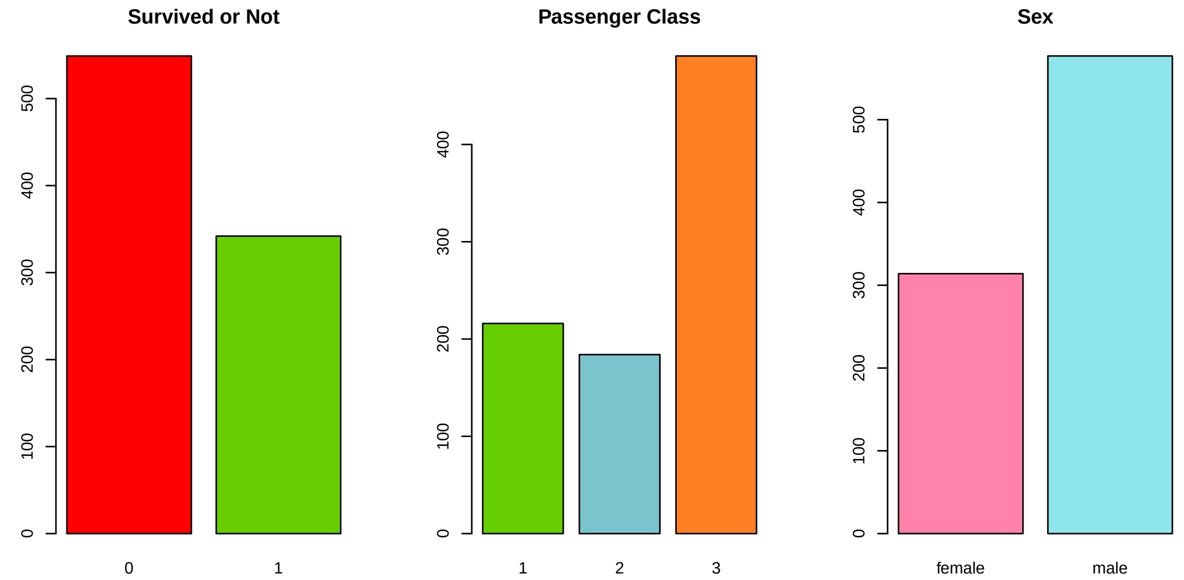
3. Graphical Exploratory Data Analysis

Now that we have the data in a basic shape for graphical EDA, we can try understanding the underlying distributions and associations of this training set better, remembering that this is just a sample so our findings are not necessarily representative of the truth, albeit in our case the sample is quite large, but I am always careful about making strong conclusions about a population when using sample data.

Univariate EDA

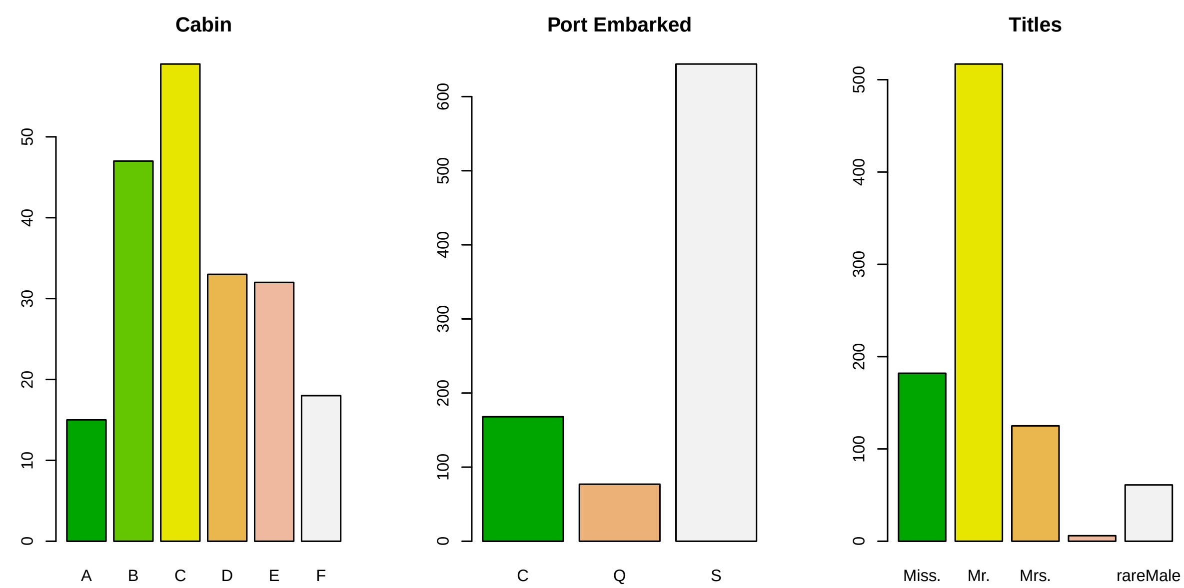
In this section we look at feature distributions one at a time. A few quick plots of the class imbalances in Survived, Passenger Class, and Sex give us a better understanding of these attributes:

```r
par(mfrow=c(1,3))
plot(train$SurvivedFac,main="Survived or Not", col=c("red","chartreuse3"))
plot(train$Pclass,main="Passenger Class", col=c("chartreuse3", "cadetblue3", "chocolate1"))
plot(train$Sex,main="Sex", col=c("palevioletred1","cadetblue2"))
```



Majorities did not survive, were in class 3 and were males, so being female and higher class is an indicator of survival, as expected.

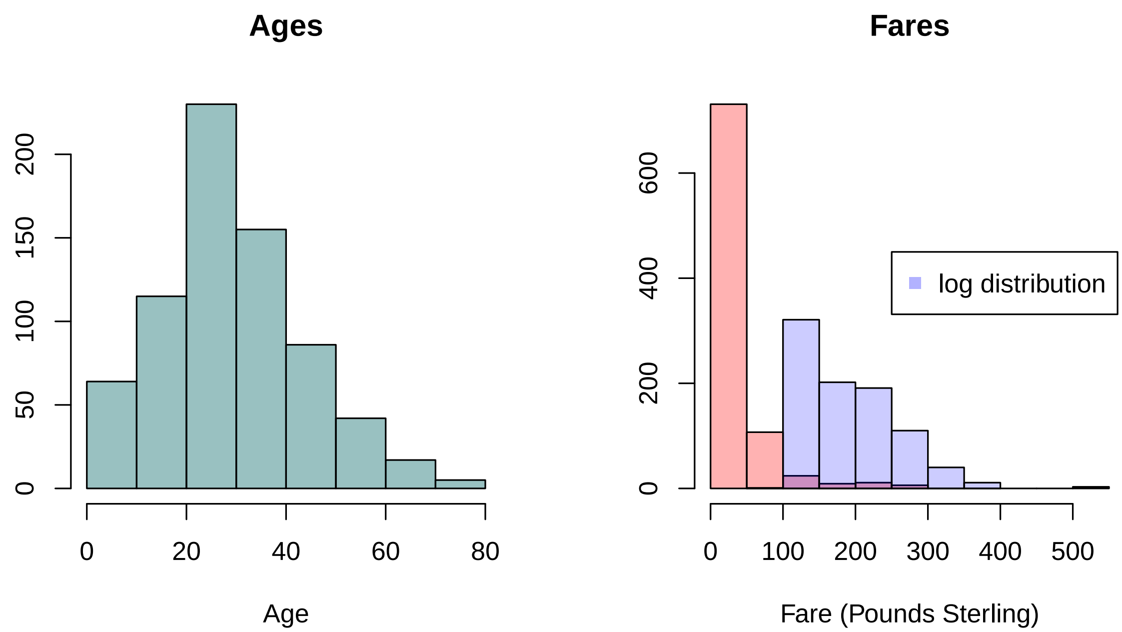
```
par(mfrow=c(1,3))
plot(train$Cabin, main="Cabin", col=terrain.colors(6))
plot(train$Embarked, main="Port Embarked", col=terrain.colors(3))
train$Title <- factor(train$Title)
plot(train$Title, main="Titles", col=terrain.colors(5))
```



Most people were in Cabin C and yet the distribution is not too skewed, while a vast majority embarked in Southhampton. Most titles are Mr., and rare female titles are barely represented.

Tackling the distributions of the two numerical variables `Age` and `Fare`:

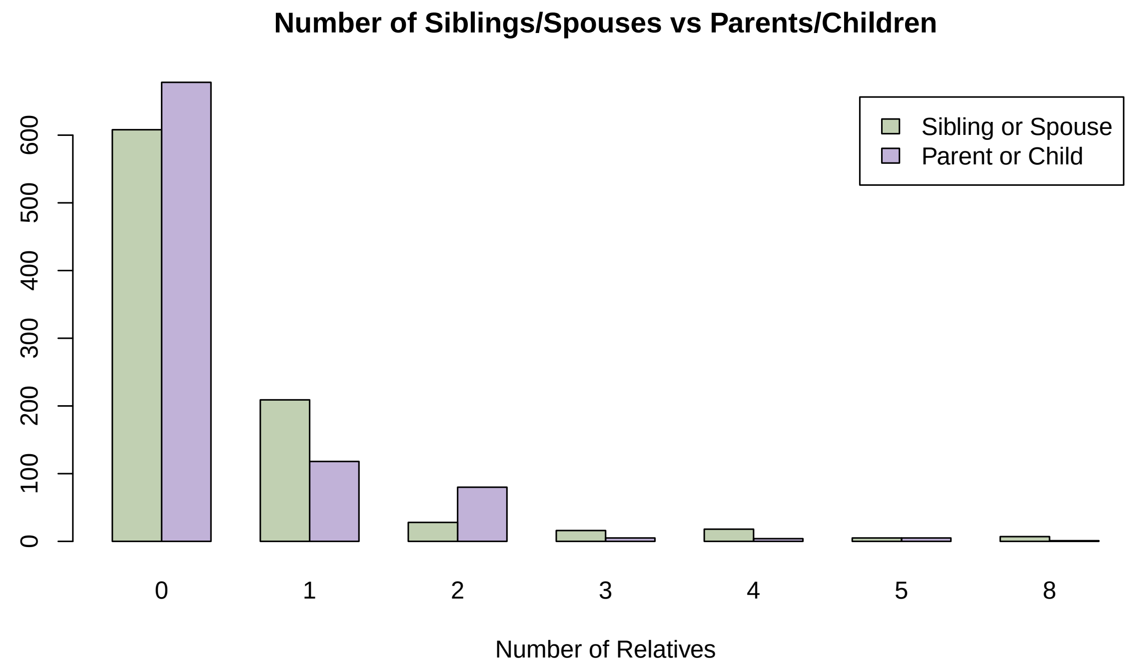
```
par(mfrow=c(1,2))
hist(train$Age, xlab='Age', main="Ages", ylab="", col=rgb(0,0.4,0.4,0.4))
hist(train$Fare, xlab="Fare (Pounds Sterling)", ylab="", main="Fares", col=rgb(1,0,0,0.3))
hist(log(train$Fare)*max(train$Fare)/8, col=rgb(0,0,1,0.2), ylab="", add=TRUE)
legend(250, 450, pch=15, col=rgb(0,0,1,0.3), "log distribution")
```



Ages are as expected roughtly normally distributed, with some older folks positively skewing the distribution. We will consider binning this variable and selecting age groups such as Children, Teenagers, Adults, and Elderly, after imputation of missing values.

Fares are, as expected, quite skewed, so we could consider taking the log (in blue, scaled up for ease of comparison).

```
SS <- table(train$SibSp)
PC <- table(train$Parch)
counts <- rbind(SS,PC)
rownames(counts) <- c("Sibling or Spouse", "Parent or Child")
par(mfrow=c(1,1))
barplot(counts, main="Number of Siblings/Spouses vs Parents/Children",
  xlab="Number of Relatives", ylab="", col=c(rgb(0.2,0.4,0,0.3),rgb(0.2,0,0.5,0.3)),
  legend = rownames(counts), beside=TRUE)
```



Since the distributions are similarly skewed, we could potentially combine the Siblings/Spouse and Parents/Children attributes into a single "Number of Relatives" attribute.

Bivariate EDA

Looking at our variables again:

```
names(train)
```

```
##  [1] "Pclass"      "Sex"         "Age"         "SibSp"       "Parch"
##  [6] "Fare"        "Cabin"       "Embarked"    "SurvivedFac" "SurvivedNum"
## [11] "Title"       "NameLength"
```

There are $(n*(n-1))/2 = (11*10)/2 = 55$ possible bivariate combinations (regardless of order) of our 11 variables. We can compute bivariate and higher-order combinations with the `combn()` function:

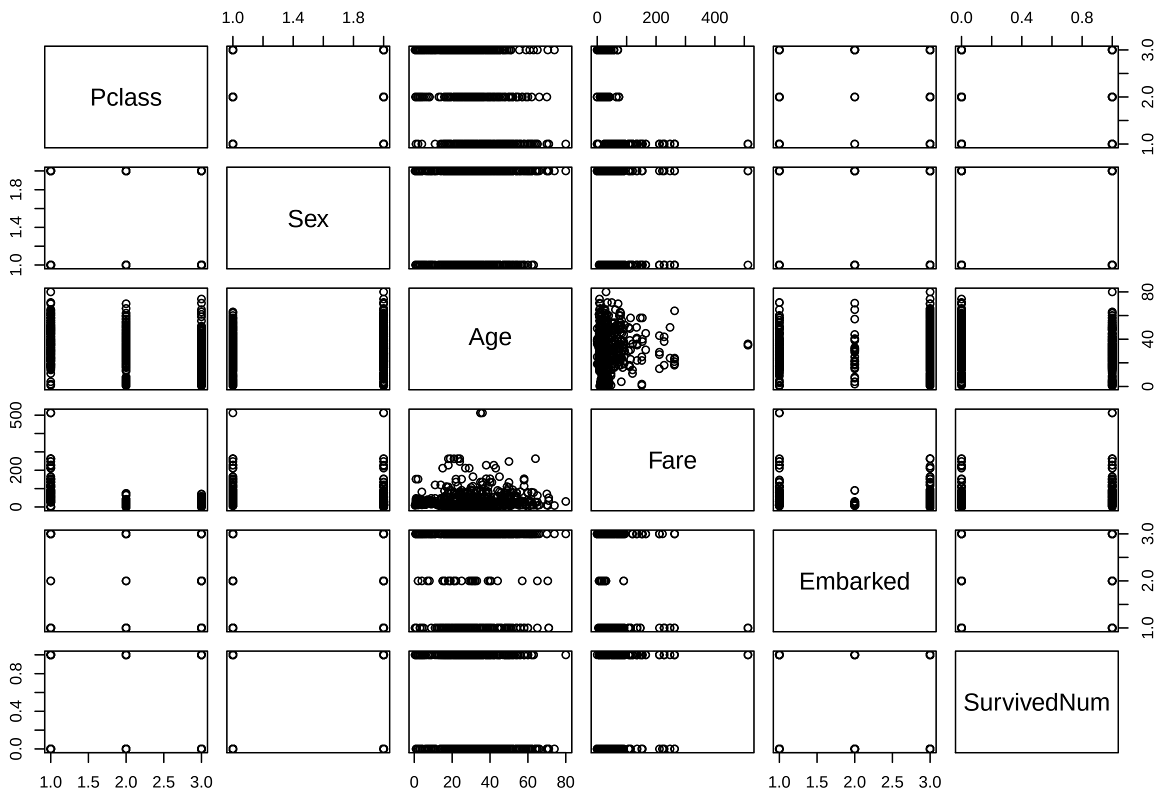
```
combn(11, 2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    1    1    1    1    1    1    1    1    1     1     2     2     2
## [2,]    2    3    4    5    6    7    8    9   10    11     3     4     5
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]     2     2     2     2     2     2     3     3     3     3     3
## [2,]     6     7     8     9    10    11     4     5     6     7     8
##      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
## [1,]     3     3     3     4     4     4     4     4     4     4     5
## [2,]     9    10    11     5     6     7     8     9    10    11     6
##      [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
## [1,]     5     5     5     5     5     6     6     6     6     6     7
## [2,]     7     8     9    10    11     7     8     9    10    11     8
```

```
##       [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55]
## [1,]     7     7     7     8     8     8     9     9    10
## [2,]     9    10    11     9    10    11    10    11    11
```

One way to plot all of these at once is using a scatterplot matrix. The `plot()` function will do this in R, when passed a **data frame**. Since it is hard to visualize 55 combinations, let's narrow down to a few choice attributes:

```
chosen <- c("SurvivedNum", "Pclass", "Sex","Age","Fare","Embarked")
plot(train[,colnames(train) %in% chosen])
```



Notice how numerical attributes (`Age` and `Fare`) combine well into a scatterplot, yet other attributes are not plotted exactly as we might want. Since the problem space will only increase with higher-dimensional combinations, we select only a few choice pairs to consider. One approach is to compare each of the other ten features with our `Survived` outcome.

Interactions with Survival

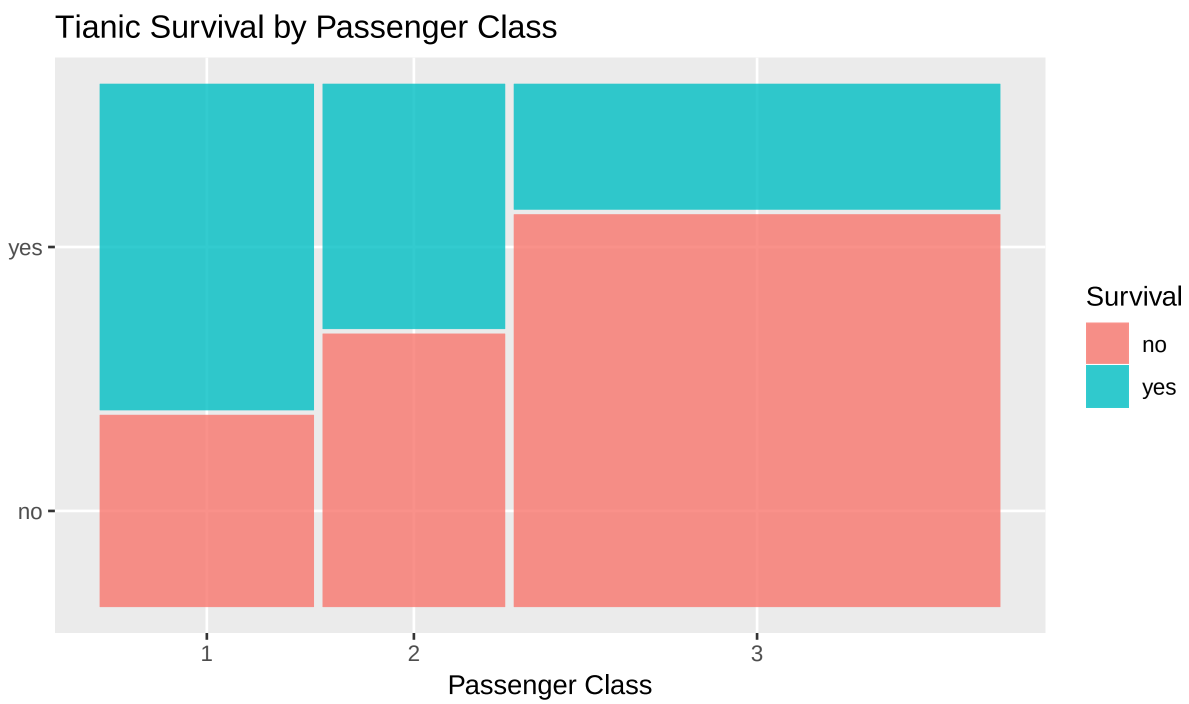
- 1 Survived & Pclass

A mosaic plot shows neatly this interaction:

```
# 1. Survived and Pclass
suppressMessages(require(ggplot2))
suppressMessages(require(ggmosaic))
```

```
## Warning: package 'ggmosaic' was built under R version 3.5.3
```

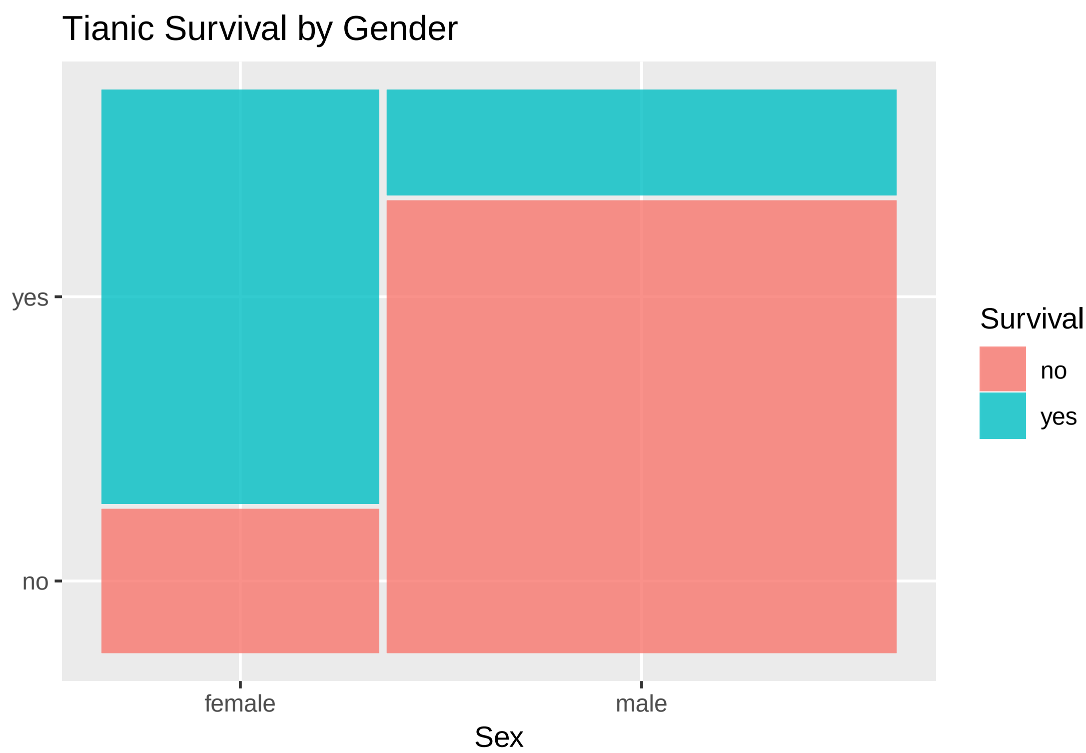
```r
Survival <- ifelse(train$SurvivedNum==1,"yes","no") # for ggplot
PclassFac <- factor(train$Pclass)
ggplot(data=train) +
  geom_mosaic(aes(x=product(Survival, PclassFac),fill=Survival)) +
  labs(x='Passenger Class', y='', title='Tianic Survival by Passenger Class')
```



It would seem that folks in first class and second class had it better than those in third class. What the mosaic plot shows is also the comparative size of the populations of these three classes (in our training sample of course).

- 2 Survived & Sex:

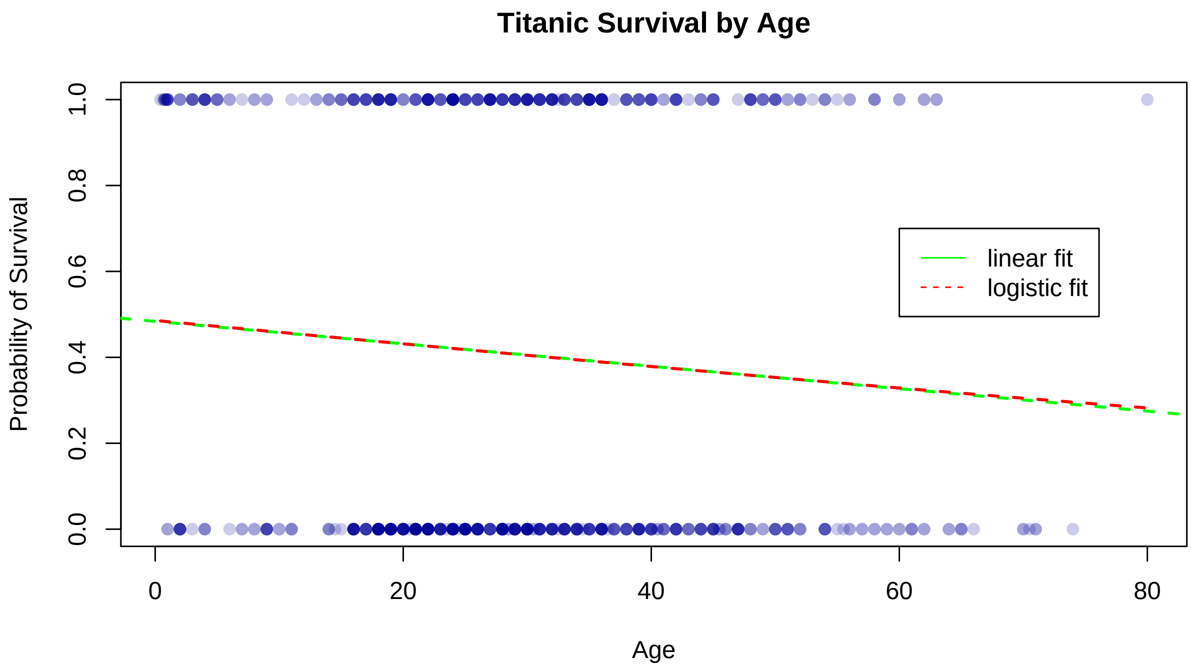
```r
ggplot(data=train) +
  geom_mosaic(aes(x=product(Survival, Sex),fill=Survival)) +
  labs(x='Sex', y='', title='Tianic Survival by Gender')
```



Females were much more likely to survive, and the majority of the passengers was male.

- 3 Survived & Age

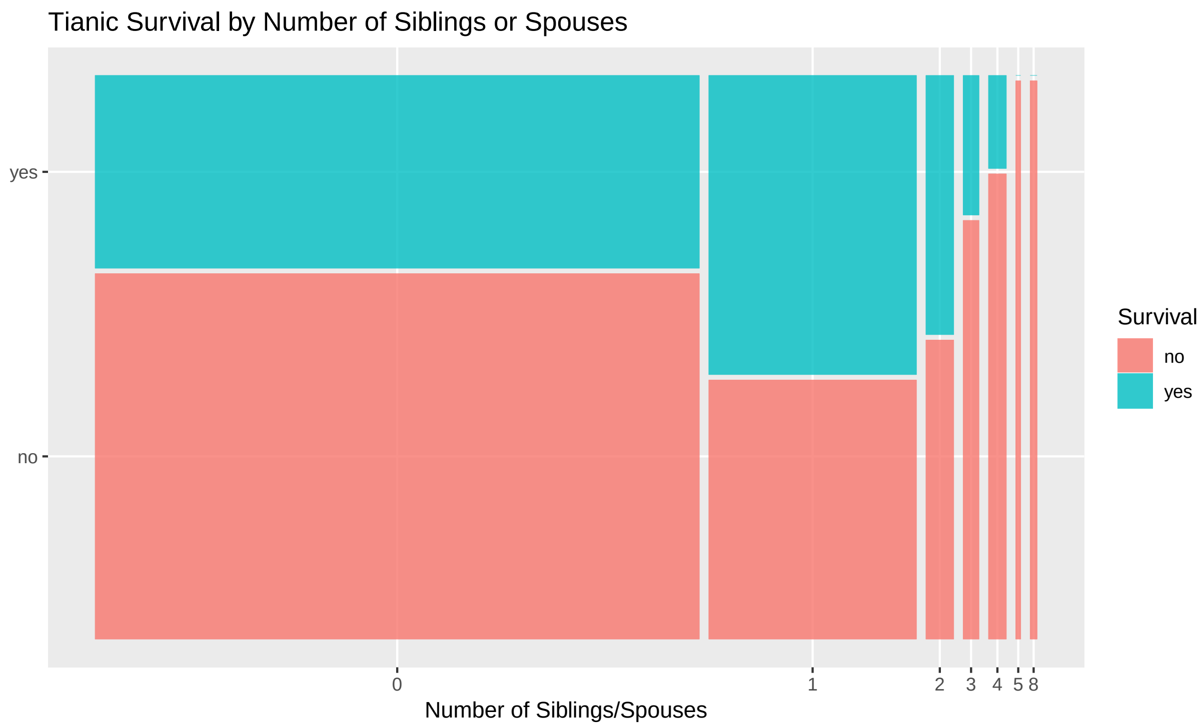
```
plot(train$SurvivedNum~train$Age, pch=19, col=rgb(0,0,.6,.2),
    main="Titanic Survival by Age",ylab="Probability of Survival", xlab="Age")
linmod=lm(SurvivedNum~Age,data=train)
abline(linmod, col="green", lwd=2, lty=2)
g=glm(SurvivedNum~Age,family='binomial',data=train)
curve(predict(g,data.frame(Age=x),type="resp"),col="red",lty=2,lwd=2,add=TRUE)
legend(60,0.7,c("linear fit","logistic fit"), col=c("green","red"), lty=c(1,2))
```



As expected, the probability of survival declines with age, as shown by the linear fit, which is quite similar to the logistic fit (a sinusoidal curve) as survival is not rare and the distribution of ages is roughly normal (as we noted in the univariate EDA), so we observe mid-range probabilities.

- 4 Survived & SibSp

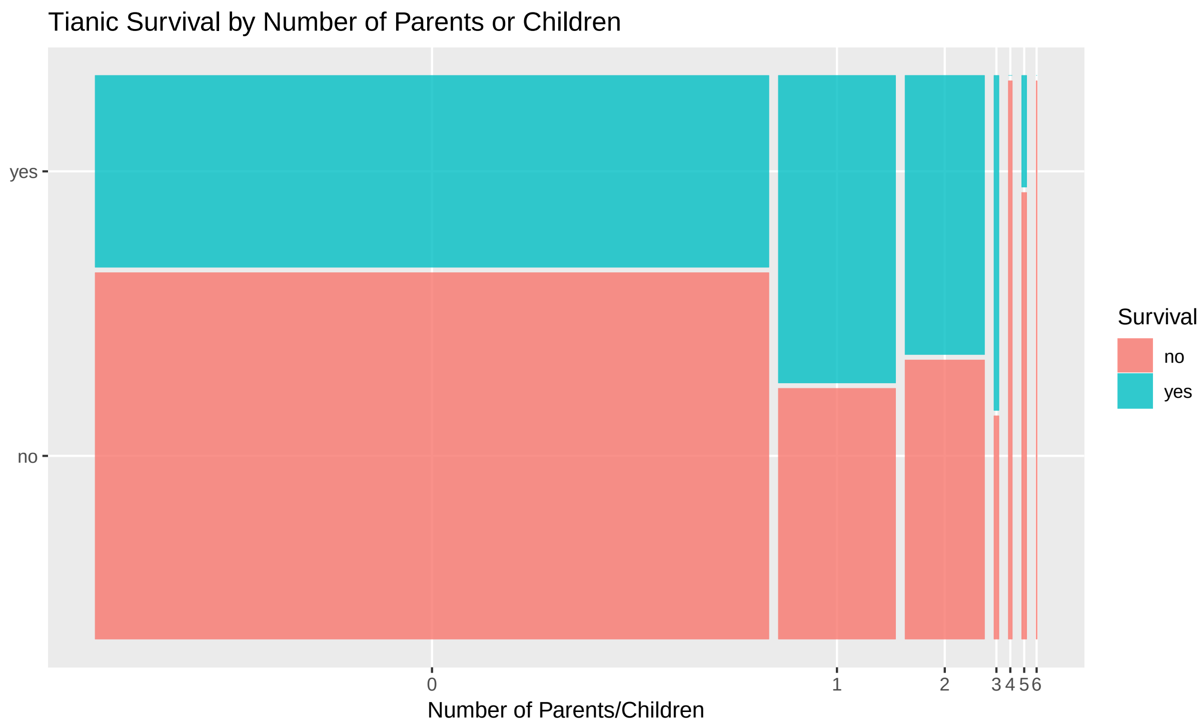
```
ggplot(data=train) +
  geom_mosaic(aes(x=product(Survival, SibSp),fill=Survival)) +
  labs(x='Number of Siblings/Spouses', y='',
  title='Tianic Survival by Number of Siblings or Spouses')
```



Having one sibling or spouse is most indicative of survival, followed by two, then none, then four and up. The probability of survival is very low for higher numbers but our confidence that this is the case should decrease because there is gradually less evidence for this effect, given the smaller sample sizes as shown in the mosaic plot.

- 5 Survived & Parch

```
ggplot(data=train) +
  geom_mosaic(aes(x=product(Survival, Parch),fill=Survival)) +
  labs(x='Number of Parents/Children', y='',
  title='Tianic Survival by Number of Parents or Children')
```

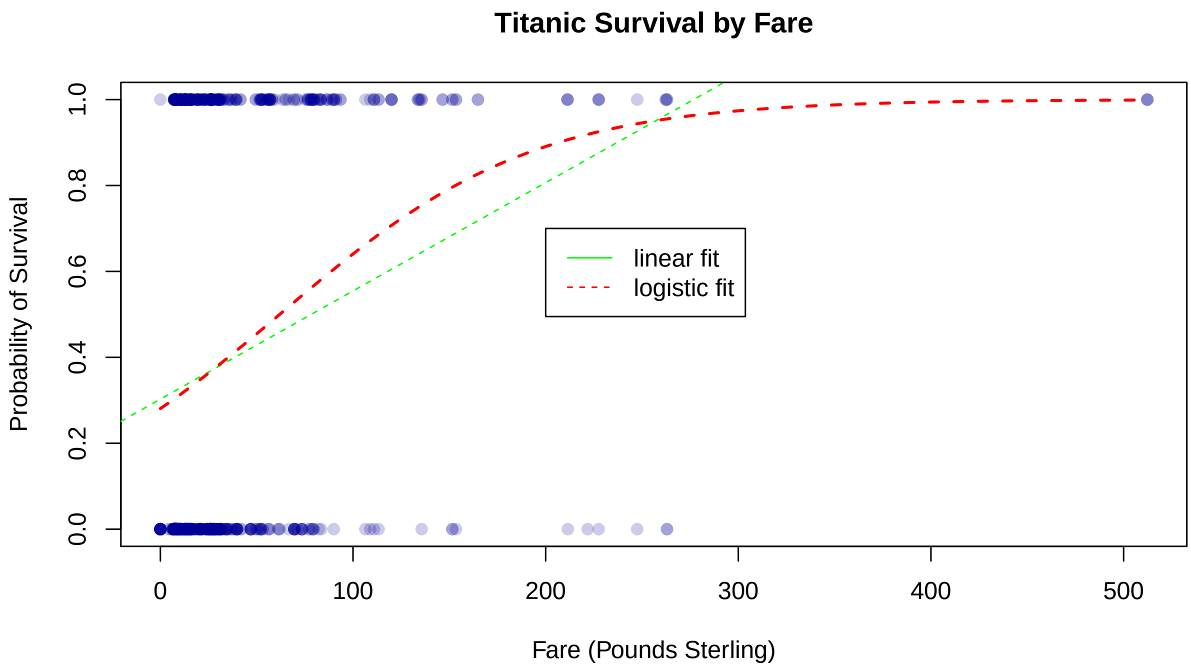


Similar results to those observed in the previous plot are seen, except for the higher probability of survival for someone with 3 (presumably) children, yet again, since the sample sizes are small, we should not take this finding too seriously.

When **feature engineering** we will take into account these findings to select the best method to create our indicator variables.

- 6 Survived & Fare

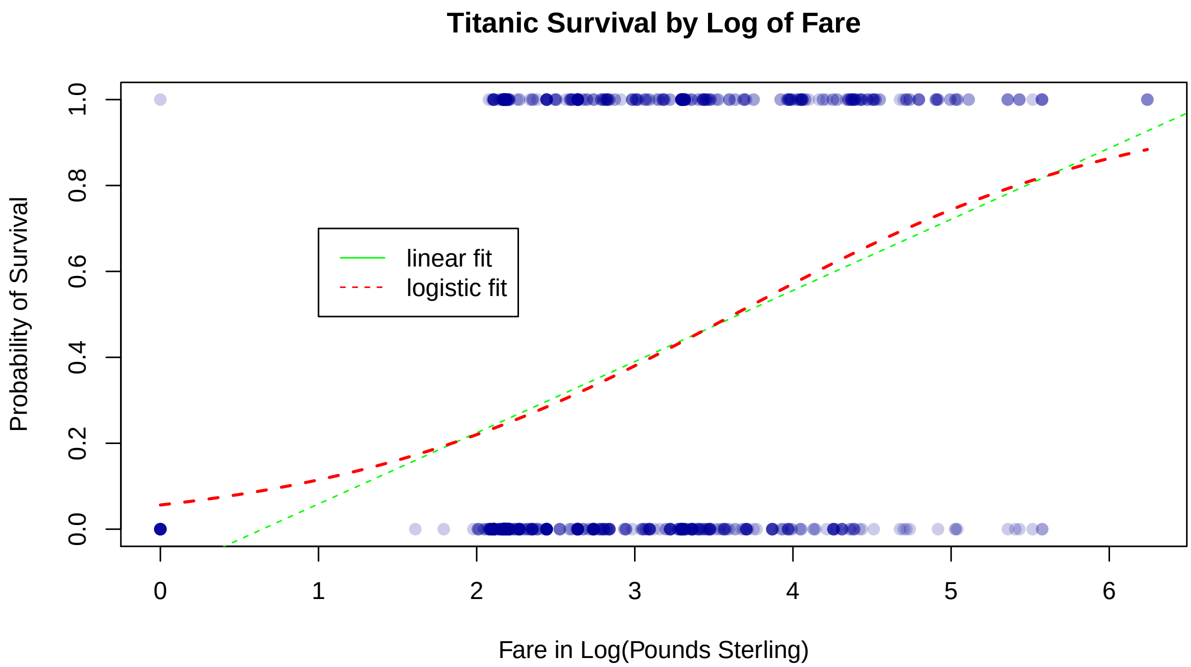
```
plot(train$SurvivedNum~train$Fare, pch=19, col=rgb(0,0,.6,.2),
    main="Titanic Survival by Fare",ylab="Probability of Survival", xlab="Fare (Pounds Sterling)")
linmod=lm(SurvivedNum~Fare,data=train)
abline(linmod, col="green", lwd=1, lty=2)
g=glm(SurvivedNum~Fare,family='binomial',data=train)
curve(predict(g,data.frame(Fare=x),type="resp"),col="red",lty=2,lwd=2,add=TRUE)
legend(200,0.7,c("linear fit","logistic fit"), col=c("green","red"), lty=c(1,2))
```



Unlike the plot of Survival by Age, we observe extreme probabilities given the skewed distribution of Fare, which shows how survival is increasingly more probable the higher the fare.

We can explore creating a log of Fare which could be used in modeling, as some models (i.e. linear models) would benefit from this logged variable as opposed to the original Fare attribute.

```
# creating Log of Fare attribute
train$FareLog <- log(train$Fare+1) # adding a pound to avoid Inf log values for 0 fares
# plot
plot(train$SurvivedNum~train$FareLog, pch=19, col=rgb(0,0,.6,.2),
    main="Titanic Survival by Log of Fare",
    ylab="Probability of Survival", xlab="Fare in Log(Pounds Sterling)")
linmod=lm(SurvivedNum~FareLog,data=train)
abline(linmod, col="green", lwd=1, lty=2)
g=glm(SurvivedNum~FareLog,family='binomial',data=train)
curve(predict(g,data.frame(FareLog=x),type="resp"),col="red",lty=2,lwd=2,add=TRUE)
legend(1,0.7,c("linear fit","logistic fit"), col=c("green","red"), lty=c(1,2))
```

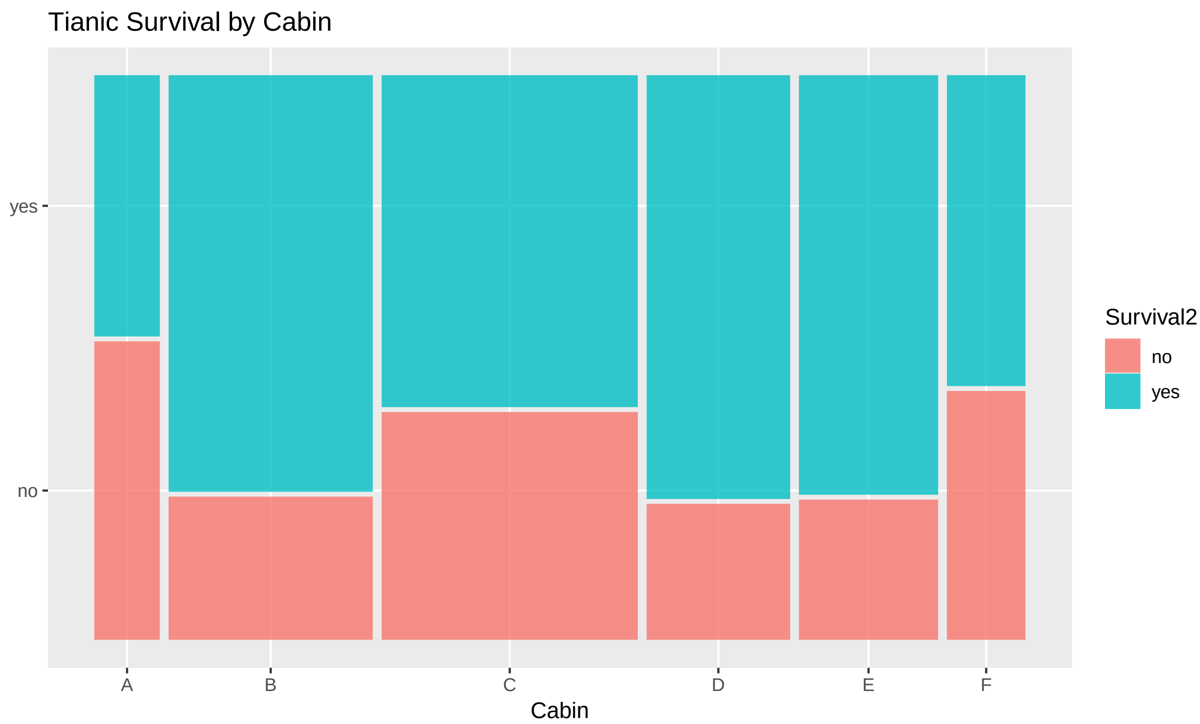


The `FareLog` variable will indeed be useful for linear modeling.

- 7 Survived & Cabin

Since our data has so many missing values for cabin, our confidence in the results of this plot should be decreased.

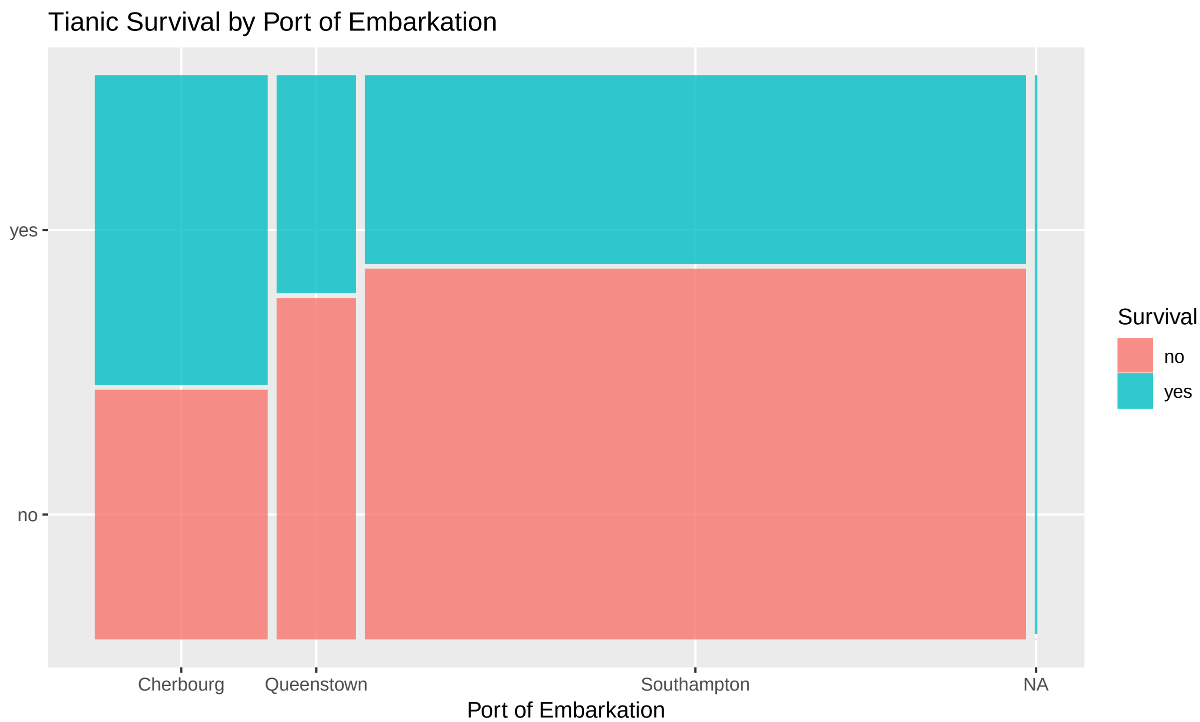
```
# creating a copy of train with no missing values for cabin
train2 <- train[!is.na(train$Cabin),]
Survival2 <- ifelse(train2$SurvivedNum==1,"yes","no")
ggplot(data=train2) +
   geom_mosaic(aes(x=product(Survival2, Cabin),fill=Survival2)) +
   labs(x='Cabin', y='', title='Tianic Survival by Cabin')
```



It would appear that perhaps cabin is not as associated with survivability as we had hoped for, given that A cabins are on the deck and F cabins near the keel where the ship hit the iceberg.

- 8 Survived & Embarked

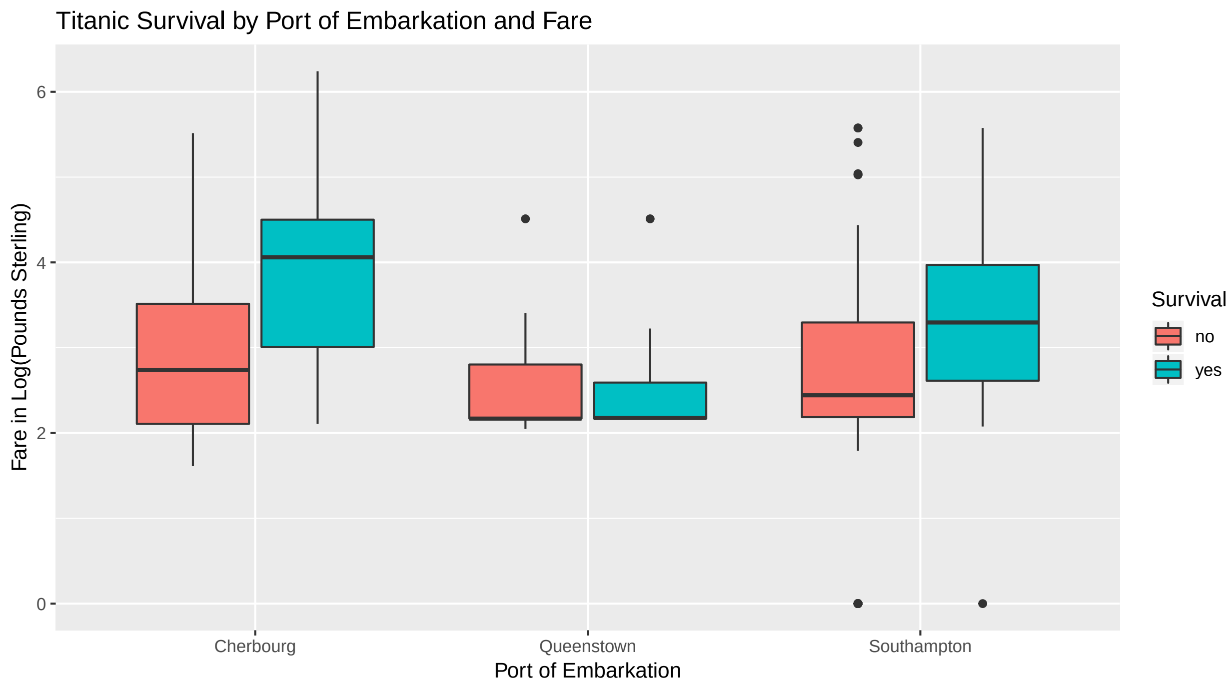
```
PortEmbarked <- ifelse(train$Embarked=="C","Cherbourg",
                ifelse(train$Embarked=="Q","Queenstown", "Southampton"))
dat <- data.frame(Survival, PortEmbarked)

ggplot(data=dat) +
   geom_mosaic(aes(x=product(Survival, PortEmbarked),fill=Survival)) +
   labs(x='Port of Embarkation', y='',
   title='Tianic Survival by Port of Embarkation')
```



There seems to be some evidence that having embarked in Southhampton is an indicator of higher probability of survival.

We can explore port of embarkation in a more nuanced manner by considering the fares paid at each port, and whether survivability appears to me more associated with the fare or the port embarked. We use the log of fares since it would be hard to observe any differences in the boxplots given the highly skewed distribution of fare.

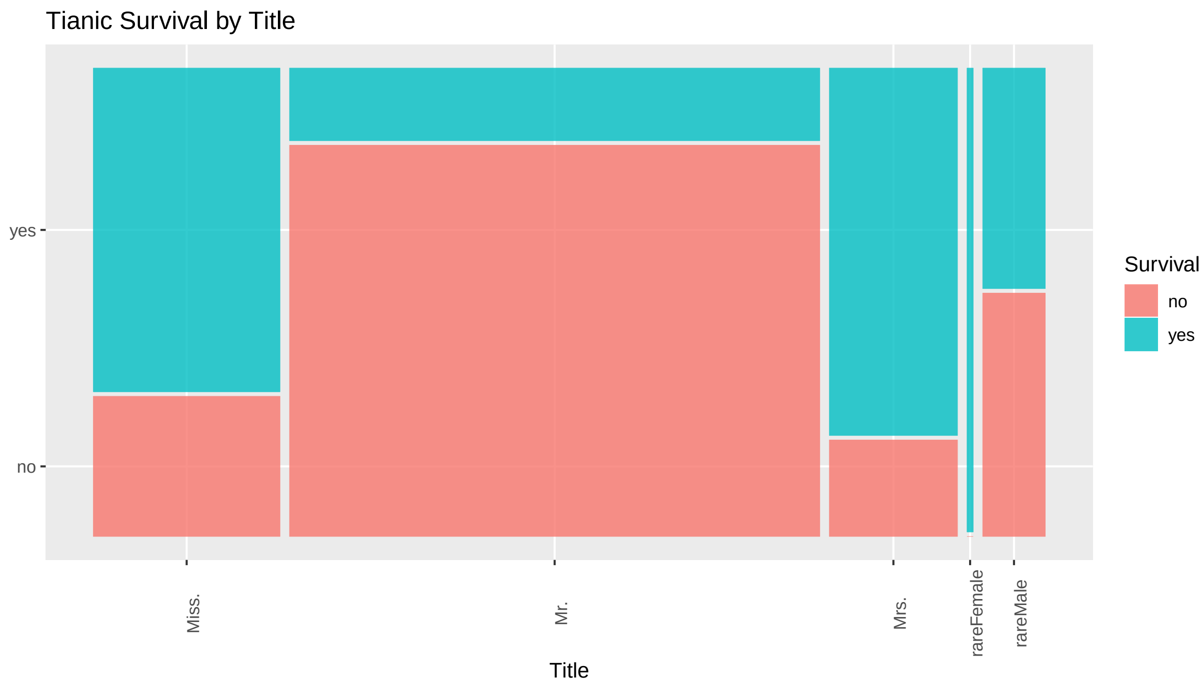
```
dat$FareLog <- train$FareLog
dat <- dat[!is.na(dat$PortEmbarked),]
ggplot(data=dat) +
   geom_boxplot(aes(x=PortEmbarked,y=FareLog, fill=Survival)) +
   labs(x='Port of Embarkation', y='Fare in Log(Pounds Sterling)',
   title='Titanic Survival by Port of Embarkation and Fare')
```



Several curiosities pop out in this plot. First, Southhampton's higher survivability is not entirely associated with fare, since Cherbourg seems to have a higher survivability when considering fare. Second, Queenstown's seems to go against common sense in that higher fares aren't necessarily associated with higher survivability. Lastly, the difference in survivability according to fare varies from port to port, for example, we see a more pronounced difference in Cherbourg, and almost no difference in Queenstown.

- 9 Survival and Title

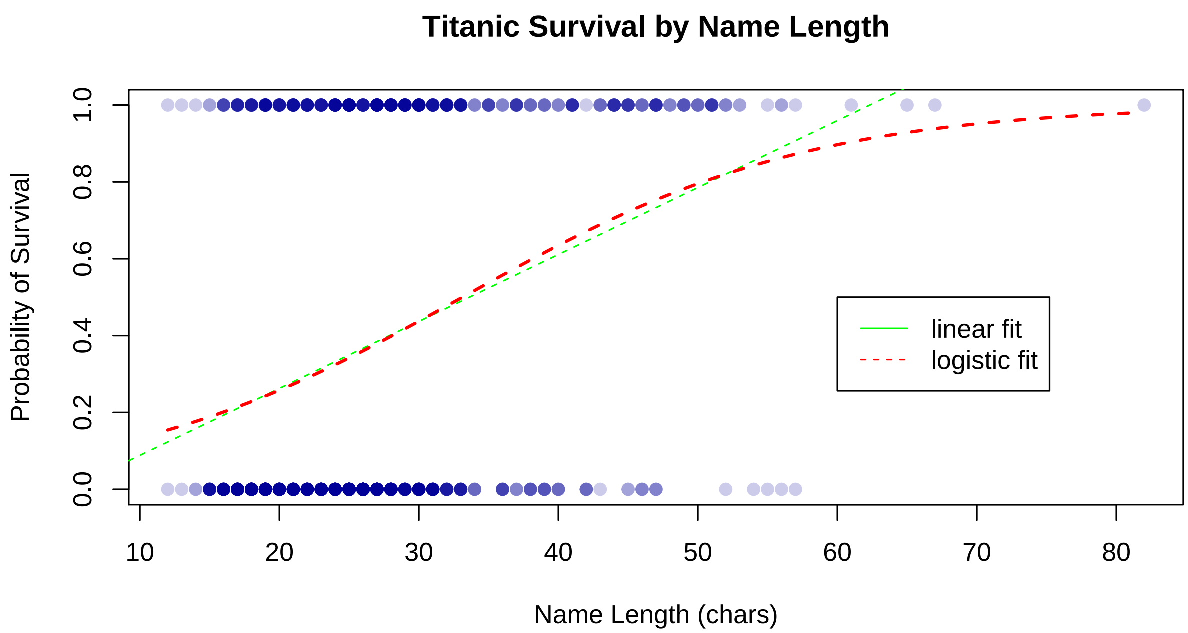
```
ggplot(data=train) +
   geom_mosaic(aes(x=product(Survival, Title),fill=Survival)) +
   labs(x='Title', y='',
   title='Tianic Survival by Title') +
   theme(axis.text.x = element_text(angle = 90))
```



Title can be seen as a proxy for gender and as we've seen, females survived a lot better than males. It is worth keeping this attribute as it shows some granularity in what kinds of folks survived better within gender groups, i.e. those with rare titles.

- 10. Survived and NameLength

```
plot(train$SurvivedNum~train$NameLength, pch=19, col=rgb(0,0,.6,.2),
    main="Titanic Survival by Name Length",
    ylab="Probability of Survival", xlab="Name Length (chars)")
linmod=lm(SurvivedNum~NameLength,data=train)
abline(linmod, col="green", lwd=1, lty=2)
g=glm(SurvivedNum~NameLength,family='binomial',data=train)
curve(predict(g,data.frame(NameLength=x),type="resp"),col="red",lty=2,lwd=2,add=TRUE)
legend(60,0.5,c("linear fit","logistic fit"), col=c("green","red"), lty=c(1,2))
```



Longer names do appear to have some association with higher probabilities of survival so we are also keeping this feature. It might just be capturing the association of longer names and wealth, but since in machine learning we do not care about multicollinearity issues, we will test whether to keep this attribute or not during our feature selection modeling phase.

Multi-dimensional EDA

The higher-dimensional problem space of combinations with 11 variables is as follows:

```r
# Combinations of 3 or more variables quickly explode
vars <- 1:11
for (i in 2:9) {
    num <- length(combn(vars,i))/i
    print(paste("There are ", num, "combinations of 11 variables taken", i, "at a time."))
}
```

```
## [1] "There are  55 combinations of 11 variables taken 2 at a time."
## [1] "There are  165 combinations of 11 variables taken 3 at a time."
## [1] "There are  330 combinations of 11 variables taken 4 at a time."
## [1] "There are  462 combinations of 11 variables taken 5 at a time."
## [1] "There are  462 combinations of 11 variables taken 6 at a time."
## [1] "There are  330 combinations of 11 variables taken 7 at a time."
## [1] "There are  165 combinations of 11 variables taken 8 at a time."
## [1] "There are  55 combinations of 11 variables taken 9 at a time."
```

The number of combinations is complementary (adding up to 11): $6 = 5$, $7 = 4$, etc., so when considering combinations of 9 variables, we are just considering the complement of 2 variables.

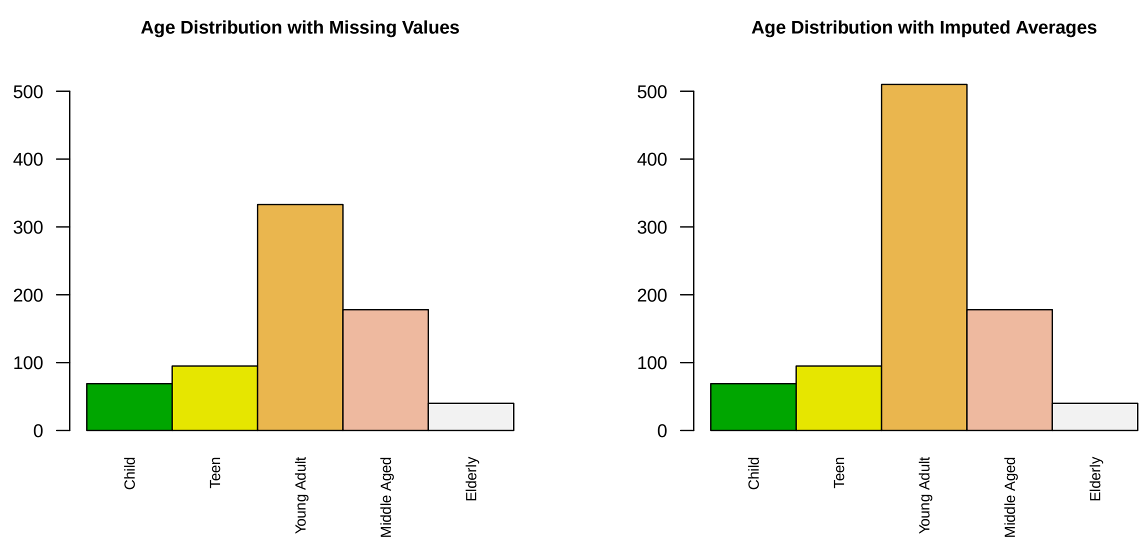
Of the 165 trivariate combinations, 45 alone are possibilities that interact with our outcome `Survival` – clearly too many to consider, so our approach will by means of necessity be minimal and select, but I wanted to make the size of the entire enterprise known.

Age Categories and Imputation

To make matters worse, I am breaking down the `Age` continuous variable into four new features by age group: `Child` $(0 - 12)$ `Teen` $(13 - 19)$, `YoungAdult` $(20 - 35)$, `MiddleAged` $(36 - 55)$, and `Elderly` $(56 - 80)$. While some of these age groups might seem young by 2019's standards, at the time of the Titanic's maiden voyage, I believe they more accurately reflect the respective age groups, as evidenced by the distribution below. This breakdown will help later with machine learning but also help EDA in exploring age groups in a discrete manner.

Before we actually create these features, however, we need to impute the missing values. One strategy is to use a mean or median age, which would unnecessarily inflate the `YoungAdult` variable, as seen in the second plot below.

```
Child <- sum(ifelse(train$Age > 0 & train$Age < 13,1,0),na.rm=TRUE)
Teen <-  sum(ifelse(train$Age > 12 & train$Age < 20,1,0),na.rm=TRUE)
YoungAdult <- sum(ifelse(train$Age > 19 & train$Age < 36,1,0),na.rm=TRUE)
MiddleAged <- sum(ifelse(train$Age > 35 & train$Age < 56,1,0),na.rm=TRUE)
Elderly <- sum(ifelse(train$Age > 55 & train$Age < 100,1,0),na.rm=TRUE)
discrete_age_vec1 <- c(Child,Teen,YoungAdult,MiddleAged,Elderly)
par(mfrow=c(1,2))
barplot(discrete_age_vec1, col=terrain.colors(5), ylim=c(0,515),
        cex.names=.7, cex.main=0.8, cex.axis=0.8, las=2,
        main="Age Distribution with Missing Values",  space=c(0,0,0,0,0),
        names.arg=c("Child","Teen","Young Adult","Middle Aged","Elderly"))
# adding all 177 NAs to the Young Adult category
YoungAdult <- YoungAdult + sum(is.na(train$Age))
discrete_age_vec2 <- c(Child,Teen,YoungAdult,MiddleAged,Elderly)
barplot(discrete_age_vec2, col=terrain.colors(5), ylim=c(0,515),
        cex.names=.7, cex.main=0.8, cex.axis=0.8, las=2,
        main="Age Distribution with Imputed Averages", space=c(0,0,0,0,0),
        names.arg=c("Child","Teen","Young Adult","Middle Aged","Elderly"))
```



Another strategy would be to **generate random values** given a similar distribution to that which we observed in our traning data, yet one problem with this approach is that it overfits the values we observe, reinforcing patterns that might not necessarily be generalizable.

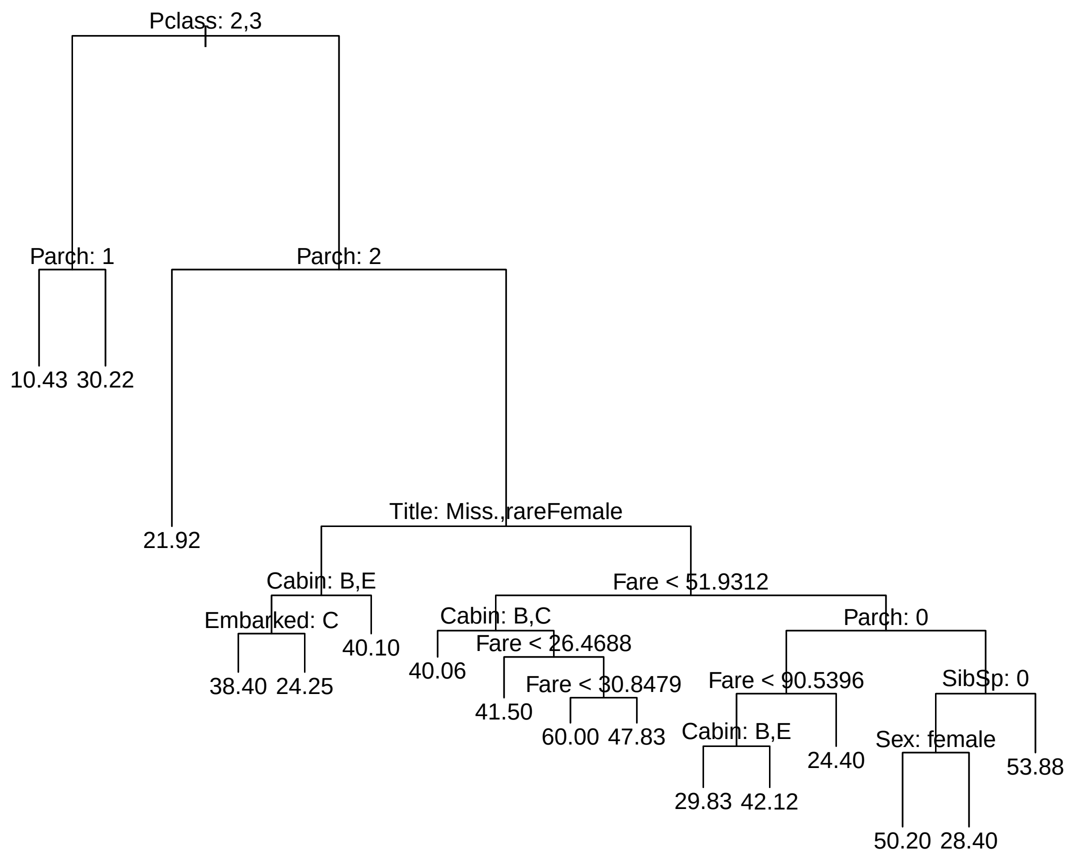
A final and more sophisticated approach would be to use the rest of the information in the training data and **generate predictions** for the ages of those individuals, based on other attributes. Since Decision Tree models are easy to use and accept missing values and unscaled features and so forth, we can easily generate ages without much hassle and modeling work.

First we separate the data into sets with age and without age:

```
'%ni%' <- Negate('%in%')
out_vars <- c("SurvivedNum","FareLog") # we do not need these redundant features
yesAge <- train[!is.na(train$Age), colnames(train) %ni% out_vars] # 714 rows
noAge <- train[is.na(train$Age), colnames(train) %ni% out_vars] # 177 rows we're trying to predict
noAge <- noAge[, colnames(noAge) != "Age"]
```

Now we can use the dataset with ages to train a tree model:

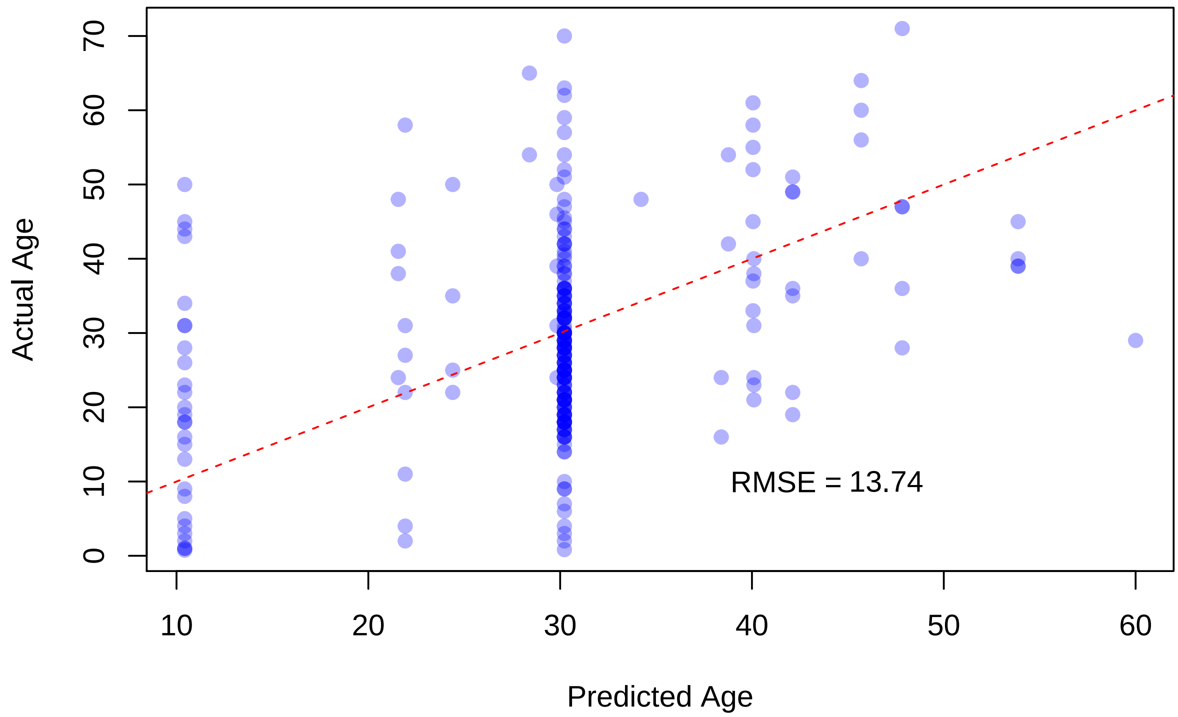
```
library(tree)
library(caTools)
set.seed(1)
# split on outcome
Y_age <- yesAge[, "Age"]
age_bool <- sample.split(Y_age, SplitRatio = 2/3)
age_train <- yesAge[age_bool, ]
age_test <- yesAge[!age_bool, colnames(yesAge) != "Age"]
# fit model
age_mod <- tree(Age~.,data=age_train)
# plot tree
plot(age_mod)
text(age_mod, pretty=0)
```



According to our single tree, the passenger class and number of parents and children are the best predictors for age, followed by title, fare, cabin, and port of embarkation, as can be seen in the numerous nodes. One problem with this single tree approach is that another random starting point would generate an entirely different tree.

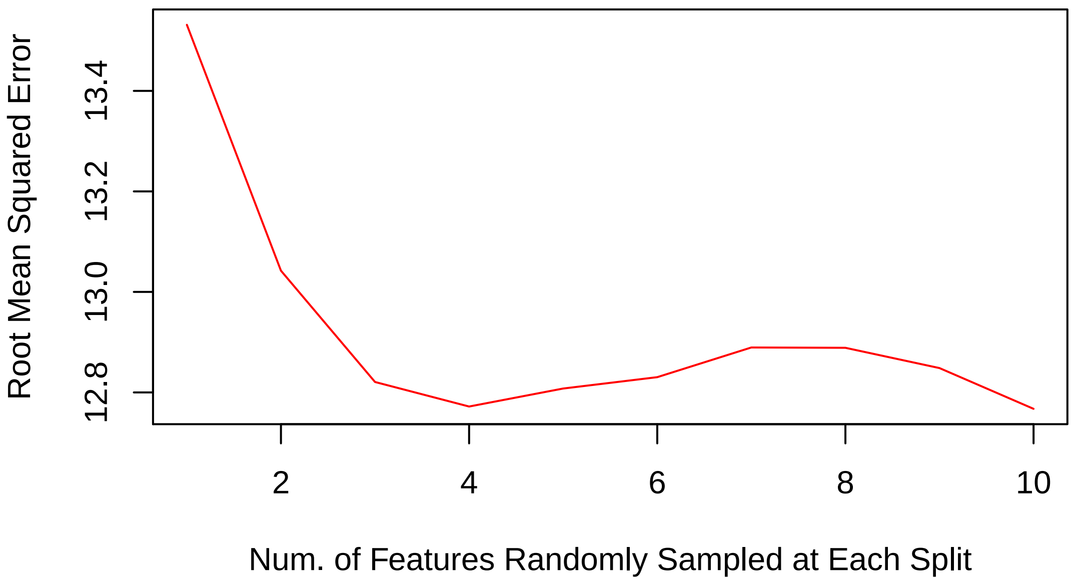
Let's first see how this tree did as far as predicting ages in the test set:

```
y_hat <- predict(age_mod, newdata=age_test)
y_test <- yesAge[!age_bool, "Age"]
test_RMSE <- round(sqrt(mean((y_hat - y_test)^2)),2)
plot(y_hat, y_test,ylab="Actual Age",xlab="Predicted Age",pch=19,col=rgb(0,0,1,0.3))
text(c(42,47),10, c("RMSE = ", test_RMSE))
abline(0,1, col="red",lty=2)
```



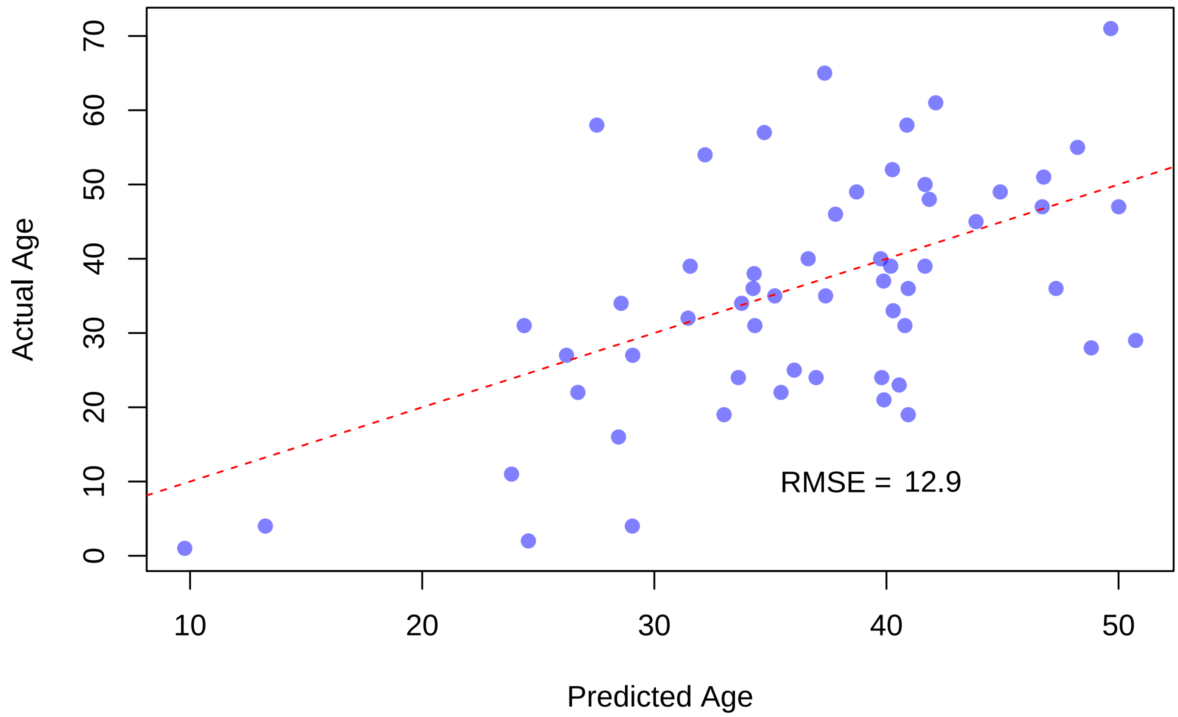
The tree seems to be overpredicting discrete levels of ages such as 30, 10, and 40, and making lots of errors. Let's see if an ensemble model performs better.

```
suppressMessages(library(randomForest))
set.seed(1)
# checking various RMSEs
rf_RMSEs <- vector("numeric", length=10)
for (i in 1:10) {
  rf_age <- randomForest(Age ~., data=age_train, mtry=i, na.action=na.omit)
  rf_yhat <- predict(rf_age, newdata=age_test)
  rf_RMSEs[i] <- sqrt(mean((rf_yhat - y_test)^2,na.rm=TRUE))
}
plot(rf_RMSEs, ylim=range(rf_RMSEs), ylab="Root Mean Squared Error", col="red",
     xlab="Num. of Features Randomly Sampled at Each Split", type="l")
```



Looks like 4 features gets the job done:

```
set.seed(1)
rf_age <- randomForest(Age ~., data=age_train, mtry=4, na.action=na.omit)
rf_yhat <- predict(rf_age, newdata=age_test)
test_RMSE <- round(sqrt(mean((rf_yhat - y_test)^2, na.rm=TRUE)), 2)
plot(rf_yhat, y_test,ylab="Actual Age",xlab="Predicted Age",pch=19,col=rgb(0,0,1,0.5))
text(c(38,42),10, c("RMSE = ", test_RMSE))
abline(0,1, col="red",lty=2)
```



The root mean squared error is a bit better even though there were many missing values in this model.

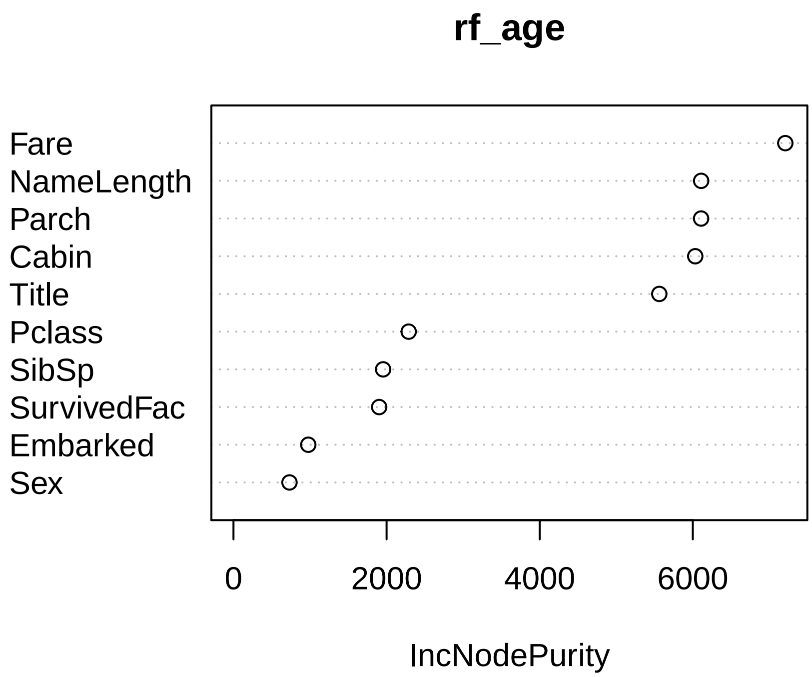
Making predictions on the `noAge` data, imputing values and comparing the new, full `Age` distribution to our original distribution of ages.

```
set.seed(1)
rf_age <- randomForest(Age ~., data=yesAge, mtry=4, na.action=na.omit)
noAge$Age <- predict(rf_age, newdata=noAge)
round(colMeans(is.na(noAge))[colSums(is.na(noAge))>0]*100,2)
```

```
## Cabin   Age
## 89.27 89.27
```

We need to drop `Cabin` from the features given that the random forest model relies on it too much and it has too many missing values.

```
varImpPlot(rf_age)
```

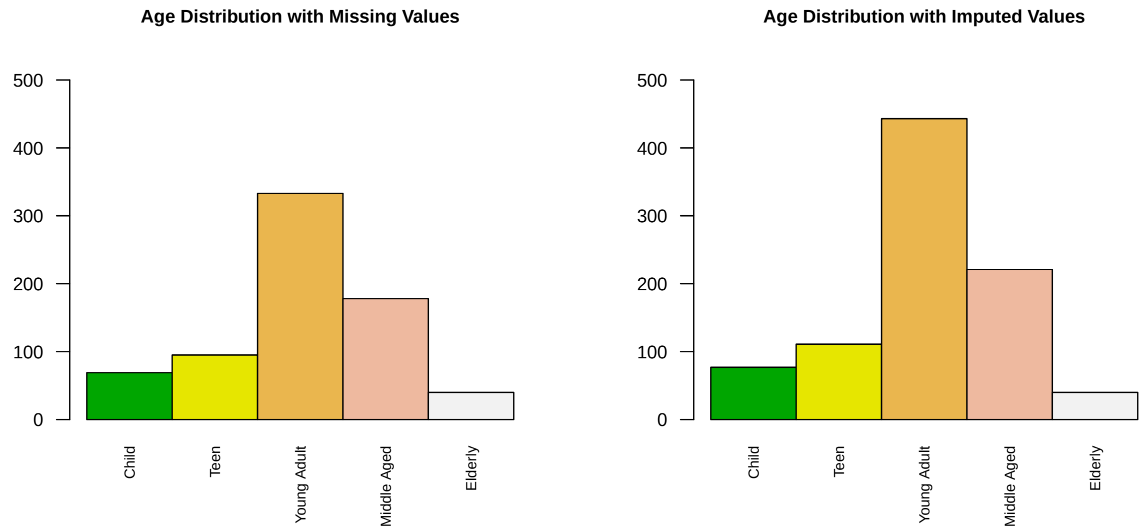



```r
set.seed(1)
yesAge <- yesAge[,colnames(yesAge) != "Cabin"]
noAge <- noAge[,colnames(noAge) != "Cabin"]
rf_age <- randomForest(Age ~., data=yesAge, mtry=4, na.action=na.omit)
# imputing Age predictions
train$Age[is.na(train$Age)] <- round(predict(rf_age, newdata=noAge),0)
sum(is.na(train$Age)) == 0
```

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

Confirming we have no missing values in `Age`, we now plot the new distribution of this variable:

```r
Child <- sum(ifelse(train$Age > 0 & train$Age < 13,1,0))
Teen <-  sum(ifelse(train$Age > 12 & train$Age < 20,1,0))
YoungAdult <- sum(ifelse(train$Age > 19 & train$Age < 36,1,0))
MiddleAged <- sum(ifelse(train$Age > 35 & train$Age < 56,1,0))
Elderly <- sum(ifelse(train$Age > 55 & train$Age < 100,1,0))
discrete_age_vec3 <- c(Child,Teen,YoungAdult,MiddleAged,Elderly)
par(mfrow=c(1,2))
# original values
barplot(discrete_age_vec1, col=terrain.colors(5), ylim=c(0,515),
        cex.names=.7, cex.main=0.8, cex.axis=0.8, las=2,
        main="Age Distribution with Missing Values",  space=c(0,0,0,0,0),
        names.arg=c("Child","Teen","Young Adult","Middle Aged","Elderly"))
# imputed values
barplot(discrete_age_vec3, col=terrain.colors(5), ylim=c(0,515),
        cex.names=.7, cex.main=0.8, cex.axis=0.8, las=2,
        main="Age Distribution with Imputed Values",  space=c(0,0,0,0,0),
        names.arg=c("Child","Teen","Young Adult","Middle Aged","Elderly"))
```



We see a slight improvement from our previous imputation with averages. Treating `Age` as a numerical variable might have shown a greater improvement so we will keep both continuous and ordinal versions.

```
train$AgeFac <- ifelse(train$Age > 0 & train$Age < 13, "Child",
                    ifelse(train$Age > 12 & train$Age < 20, "Teen",
                    ifelse(train$Age > 19 & train$Age < 36, "YoungAdult",
                    ifelse(train$Age > 35 & train$Age < 56, "MiddleAged", "Elderly"))))
str(train)
```

```
## 'data.frame':    891 obs. of  14 variables:
##  $ Pclass     : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 41 54 2 27 14 ...
##  $ SibSp      : Factor w/ 7 levels "0","1","2","3",..: 2 2 1 2 1 1 1 4 1 2 ...
##  $ Parch      : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 2 3 1 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 6 levels "A","B","C","D",..: NA 3 NA 3 NA NA 5 NA NA NA ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
##  $ SurvivedFac: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ SurvivedNum: num  0 1 1 1 0 0 0 0 1 1 ...
##  $ Title      : Factor w/ 5 levels "Miss.","Mr.",..: 2 3 1 3 2 2 2 5 3 3 ...
##  $ NameLength : num  23 51 22 44 24 16 23 30 49 35 ...
##  $ FareLog    : num  2.11 4.28 2.19 3.99 2.2 ...
##  $ AgeFac     : chr  "YoungAdult" "MiddleAged" "YoungAdult" "YoungAdult" ...
```