Titanic Survival Part 1: EDA in R

Marcelo Sanches

July 4, 2019

Contents

- Summary
- Preliminary EDA
- Pre-Processing 1: PassengerId, Survived, Pclass
- Pre-Processing 2: Name, Sex
- Pre-Processing 3: SibSp, Parch
- Pre-Processing 4: Ticket, Fare
- Pre-Processing 5: Cabin, Embarked
- Pre-Processing 6: Age
- Univariate Graphical EDA
- Bivariate Graphical EDA
- Multivariate Graphical EDA
- Conclusion

Summary

Motivation

The **Titanic Survival Project** was born out of my desire not only to participate in a Kaggle competition but also do avoid the common mistake of overfitting the test set by submitting various predictions. In a realistic production environment, the test set would be future data that we would run a single prediction on, not a playground for getting better at prediction outcomes of that particular set of data.

I also wanted to use **R** and **Python** and play to their strengths: R for **data exploration** and **quick visualizations**, and Python for **machine-learning pipelines** and **production code**.

Project Parts

In **Part 1** of the Titanic Survival project I conduct **Exploratory Data Analisys (EDA)** of the Kaggle Titanic train dataset in R, creating an **RMarkdown report** with RStudio and the `knitr` package, with summary tables and visualizations, performing minor pre-processing as needed.

In **Part 2** of the project I perform all the necessary pre-processing steps for Machine Learning models, conduct model evaluation and regularization, and run predictions using Python in a **Jupyter Notebook**. Given a final model, I run a **single pipeline** for **pre-processing and modeling** that emulates a production environment, where the Titanic test set is used as if it were future data never before seen.

Part 1 Sections

In the **Contents** we can see how **Part 1** is divided into several sections. After a **preliminary EDA** we spend most of our time **pre-processing** the data, as usual, and then spend a fair amount of time exploring it through **visualizations**. I tried to organize this exploration into **univariate, bivariate,** and **multivariate** sub-sections, fully aware that the number of possible combinations quickly explodes even with a few attributes, so this exploration is still mostly ad hoc.

Preliminary EDA

First we load the training data and look at its structure, summary, top and bottom rows. 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.

```
## '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 not an attribute of a passenger). The attributes are:

- **Survived**, integer, binary indicator (Survived = 1) and the target outcome or dependent variable we are to predict.
- **Pclass**, integer, an ordinal variable for the passenger class.
- **Name**, Factor w/ 891 levels (one level per passenger).
- **Sex**, Factor with two levels: "female", "male".
- **Age**, numerical, has 177 missing values coded as `NA`.
- **SibSp**, integer, an ordinal variable for the number of siblings or spouses.
- **Parch**, integer, an ordinal variable for the number of parents or children.
- **Ticket**, Factor w/ 681 levels.
- **Fare**, numerical, is in Pounds Sterling, a proxy for wealth or social status.
- **Cabin**, Factor w/ 147 levels, has 687 missing values.
- **Embarked**, Factor w/ 3 levels: "C", "Q", and "S" for the port of embarkation (Cherbourg, Queenstown, and Southhampton), has 2 missing values.

```r
# Preliminary summary
summary(train)
```

```
##   PassengerId       Survived          Pclass     
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000  
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000  
##  Median :446.0   Median :0.0000   Median :3.000  
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309  
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000  
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000  
##
```

```
##                                     Name         Sex           Age       
##  Abbing, Mr. Anthony                  :  1   female:314   Min.   : 0.42  
##  Abbott, Mr. Rossmore Edward          :  1   male  :577   1st Qu.:20.12  
##  Abbott, Mrs. Stanton (Rosa Hunt)     :  1                Median :28.00  
##  Abelson, Mr. Samuel                  :  1                Mean   :29.70  
##  Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00  
##  Adahl, Mr. Mauritz Nils Martin       :  1                Max.   :80.00  
##  (Other)                              :885                NA's   :177    
##      SibSp           Parch             Ticket         Fare       
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00  
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91  
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45  
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20  
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00  
##  Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33  
##                                   (Other) :852                   
##          Cabin     Embarked  
##  B96 B98    :  4   C   :168  
##  C23 C25 C27:  4   Q   : 77  
##  G6         :  4   S   :644  
##  C22 C26    :  3   NA's:  2  
##  D          :  3             
##  (Other)    :186             
##  NA's       :687             
```

This first summary of our data is not very useful and helps us determine how to proceed with data pre-processing, converting appropriate variables into categorical format, cleaning up variables and imputing missing values as needed. I will refrain from commenting on the data until pre-processing is mostly finished.

The large number of missing values in `Cabin` and `Age` will need to be dealt with. The 2 missing values in `Embarked` can be filled in with the most common port of embarkation.

A look at the dataset helps us get a feel for it:

```r
head(train)
```

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

```
tail(train)
```

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

Pre-Processing 1: PassengerId, Survived, Pclass

PassengerId

`PassengerId` is just an index. Since R keeps a row index and we shouldn't use this variable for modeling as it provides no information, we drop it, but first check that is has no duplicates and has stepwise values to ensure data integrity (see "trust but verify" code chunk in the Appendix).

Survived

`Survived` is dropped and `SurvivedFac` (as categorical outcome) and `SurvivedNum` (as a continuous range from 0 to 1, indicating probabilities) are created.

```
# Survived
train$SurvivedFac <- ifelse(train$Survived=="1","yes","no")
train$SurvivedFac <- factor(train$Survived)
train$SurvivedNum <- as.numeric(train$Survived)
train$Survived <- NULL # drop original
```

Pclass

`Pclass` is dropped and `PclassFac` (as categorical) and `PclassNum` (as ordinal) are created. The former is useful for plotting, the latter for machine learning.

```
# Pclass
train$PclassFac <- ifelse(train$Pclass==1, "1st Class", ifelse(train$Pclass==2, "2nd Class", "3rd Class
train$PclassFac <- factor(train$PclassFac)
train$PclassNum <- as.integer(train$Pclass)
train$Pclass <- NULL
```

Pre-Processing 2: Name, Sex

The `Name` attribute is not indicative of a person's survival, yet information can be extracted from it such as titles and name lengths, which might contain some predictive power.

A `Title` attribute can be created by extracting titles with regular expresssions.

Title

```r
# Create Title attribute
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$Title[i] <- unlist(strsplit(unlist(regmatches(x,m))," "))[2]
}
# looking at unique titles
unique(train$Title)
```

```
##  [1] "Mr."       "Mrs."      "Miss."     "Master."   "Don."
##  [6] "Rev."      "Dr."       "Mme."      "Ms."       "Major."
## [11] "Lady."     "Sir."      "Mlle."     "Col."      "Capt."
## [16] "the"       "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.

I will not fix the title **the**, which stands for **the Countess**, since any specific fixes will not be generalizable to any future data (aka the test set) in production. Instead, I am hoping no other specific male titles (such as **the Count**) will pop up in the test data and will use the above rare male titles as baseline for the rare cases, all other rare cases will end up in the rare female bucket.

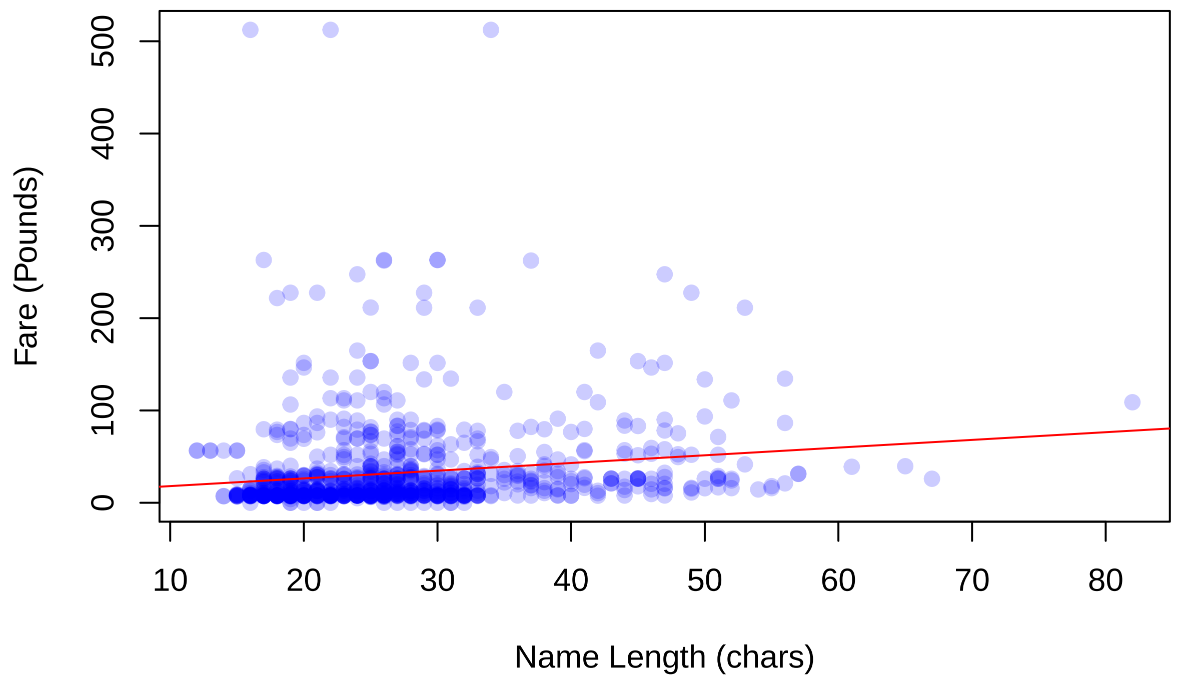
Note that some decisions are simplifications, there is a female doctor (Dr. Alice Leader) yet I assigned 'Dr.' to the rare male title category since at that time most doctors were males.

```r
# Clean up Title
common_titles <- c("Mr.", "Mrs.", "Miss.")
rare_male <- c("Don.","Rev.","Dr.","Major.","Master.", "Sir.","Col.","Capt.","Jonkheer.")
for (i in 1:nrow(train)) {
    train$Title[i] <- ifelse(train$Title[i] %in% common_titles, train$Title[i], # do not replace
                             ifelse(train$Title[i] %in% rare_male, "rareMale", "rareFemale"))
}
train$Title <- factor(train$Title)
# unique titles
unique(train$Title)
```

```
## [1] Mr.        Mrs.       Miss.      rareMale   rareFemale
## Levels: Miss. Mr. Mrs. rareFemale rareMale
```

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. (See Appendix for the `NameLength` code.)

NameLength



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 could drop `Name`, but will do so after some further cleaning as we use this for EDA later.

Sex: GenederFac, IsMale

The variable `Sex` is usually best represented as a binary indicator for a given gender in machine learning, however, for plotting purposes we keep a categorical representation, creating `GenderFac` and `IsMale` wherein Male=1.

```r
train$GenderFac <- train$Sex # factor for plotting
train$IsMale <- ifelse(train$Sex=="male",1,0) # indicator for ML
train$Sex <- NULL # drop original
```

Pre-Processing 3: SibSp, Parch

We conveniently rename `SibSp` and `Parch` to `SiblingSpouse` and `ParentChildren` and create a variable that is a sum of the two: `NumRelatives`.

```r
# Change SibSp and Parch to factor
train$SiblingSpouse <- factor(train$SibSp)
```

```r
train$ParentChildren <- factor(train$Parch)
train$NumRelatives <- train$SibSp + train$Parch
train$NumRelatives <- factor(train$NumRelatives)
train$SibSp <- NULL
train$Parch <- NULL
```

Pre-Processing 4: Ticket, Fare

Ticket

The `Ticket` attribute is somewhat useless as far as extracting information from the ticket number itself. What the ticket number does provide, however, is information on how many tickets were purchased under a given `Fare`, such that we can calculate the **fare per person**, which is what we need since our observational unit (a row) is a person.

Here is a sample of how there are repeated ticket numbers under the same fare:

```r
# EDA into Ticket and Fare
temp_dfm <- train[, colnames(train) %in% c("Name","Ticket","Fare")]
temp_dfm <- temp_dfm[order(temp_dfm["Ticket"]),] # order by Ticket
temp_dfm[1:10,]
```

```
##                                                          Name Ticket  Fare
## 258                                      Cherry, Miss. Gladys 110152 86.50
## 505                                     Maioni, Miss. Roberta 110152 86.50
## 760 Rothes, the Countess. of (Lucy Noel Martha Dyer-Edwards) 110152 86.50
## 263                                         Taussig, Mr. Emil 110413 79.65
## 559                    Taussig, Mrs. Emil (Tillie Mandelbaum) 110413 79.65
## 586                                       Taussig, Miss. Ruth 110413 79.65
## 111                            Porter, Mr. Walter Chamberlain 110465 52.00
## 476                               Clifford, Mr. George Quincy 110465 52.00
## 431                 Bjornstrom-Steffansson, Mr. Mauritz Hakan 110564 26.55
## 367          Warren, Mrs. Frank Manley (Anna Sophia Atkinson) 110813 75.25
```

There are many cases of families with the same last name (such as the Taussig above) which leads us to believe that these are not individual prices but group prices under the same ticket. So we keep `Ticket` only to clean fare.

Fare

We create a `FarePerPerson` attribute and compare it to the `Fare` attribute:

```r
# keep counts of tickets
counts <- aggregate(train$Ticket, by=list(train$Ticket),
                    FUN=function(ticket) sum(!is.na(ticket)))
# function that takes a data frame's fare and ticket counts and apply
divide_fare_count <- function(dfm) {
  fare <- as.numeric(dfm["Fare"])
  # ticket counts
  count_given_ticket <- counts[which(counts[,1] == dfm["Ticket"]), 2]
  result <- round(fare/count_given_ticket,2)
  return(result)
}
# create FarePerPerson
train$FarePerPerson <- apply(X=train, MARGIN=1, FUN=divide_fare_count)

# looking at the temp dataframe of results again
chosen <- c("Name","Ticket","Fare","FarePerPerson")
temp_dfm <- train[, colnames(train) %in% chosen]
temp_dfm <- temp_dfm[order(temp_dfm["Ticket"]),] # order by Ticket
temp_dfm[1:10,]
```

```
##                                                         Name Ticket  Fare
## 258                                     Cherry, Miss. Gladys 110152 86.50
## 505                                    Maioni, Miss. Roberta 110152 86.50
## 760 Rothes, the Countess. of (Lucy Noel Martha Dyer-Edwards) 110152 86.50
## 263                                        Taussig, Mr. Emil 110413 79.65
## 559                   Taussig, Mrs. Emil (Tillie Mandelbaum) 110413 79.65
## 586                                      Taussig, Miss. Ruth 110413 79.65
## 111                           Porter, Mr. Walter Chamberlain 110465 52.00
## 476                              Clifford, Mr. George Quincy 110465 52.00
## 431                Bjornstrom-Steffansson, Mr. Mauritz Hakan 110564 26.55
## 367         Warren, Mrs. Frank Manley (Anna Sophia Atkinson) 110813 75.25
##     FarePerPerson
## 258         28.83
## 505         28.83
## 760         28.83
## 263         26.55
## 559         26.55
## 586         26.55
## 111         26.00
## 476         26.00
## 431         26.55
## 367         75.25
```

We can now drop `Fare`, `Name`, and `Ticket`, but we can keep the ticket counts as its own attribute `TicketCount`:

```r
# create TicketCount
train$TicketCount <- apply(X=train, MARGIN=1, FUN=function(dfm) counts[which(counts[,1] == dfm["Ticket"
# drop Fare, Name, and Ticket
'%ni%' <- Negate('%in%')
not_chosen <- c("Fare","Name","Ticket")
train <- train[,colnames(train) %ni% not_chosen]
```

Since `FarePerPerson` has a skewed distribution (as we shall see in the Graphical EDA section) we create a `FarePerPErsonLog` variable which will help with linear models.

```r
# create FareLog
train$FarePerPersonLog <- log(train$FarePerPerson+1)
```

```r
names(train)
```

```
##  [1] "Age"            "Cabin"          "Embarked"
##  [4] "SurvivedFac"    "SurvivedNum"    "PclassFac"
##  [7] "PclassNum"      "Title"          "NameLength"
## [10] "GenderFac"      "IsMale"         "SiblingSpouse"
## [13] "ParentChildren" "NumRelatives"   "FarePerPerson"
## [16] "TicketCount"      "FarePerPersonLog"
```

Pre-Processing 5: Cabin, Embarked

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.

```
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.

Embarked

We substitute the letters for port names and impute the two missing cases with the majority class.

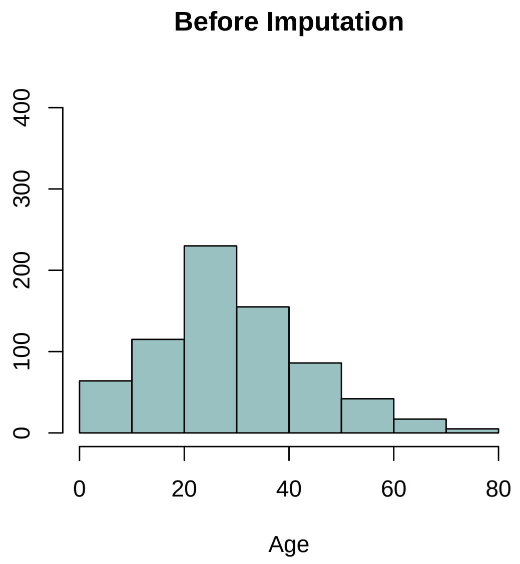
```
train$Embarked <- ifelse(train$Embarked=="C","Cherbourg",
                         ifelse(train$Embarked=="Q","Queensland","Southhampton"))
train$Embarked[is.na(train$Embarked)] <- "Southhampton"
train$Embarked <- factor(train$Embarked) # re-factoring
```

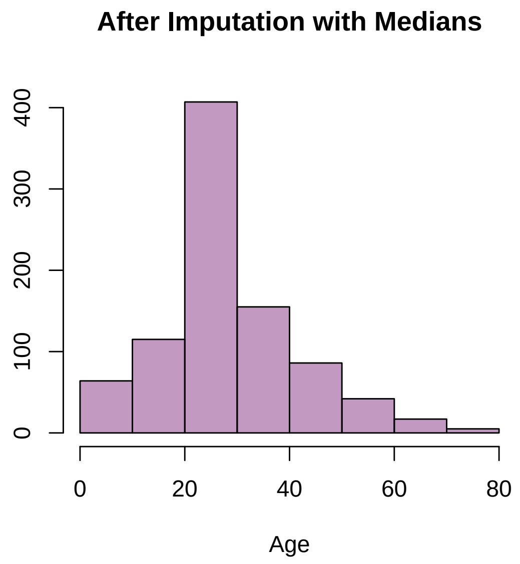
Pre-Processing 6: Age

Imputing Missing Values

The `Age` variable had to be considered at the end of pre-processing since we will be doing some pre-modeling (modeling during data pre-processing) to impute missing values and needed other variables to be relatively clean before this pre-modeling stage.

It is helpful to visualize the distribution of ages before and after a certain imputing strategy to see the effect it has on the data. The common practice of imputing with measures of center such as the mean or the median (in our case there wouldn't be much of a difference as the distribution is approximately normal) distorts the distribution. To show this effectively, the y axes must agree:





Imputing medians amounts to deciding that when we do not know an age, we will classify this person as a young adult.

A better 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 **predict ages** for those individuals, based on other attributes. Since Decision Tree models take missing and unscaled values, and work with categorical features, we can quickly predict ages with minimal modeling.

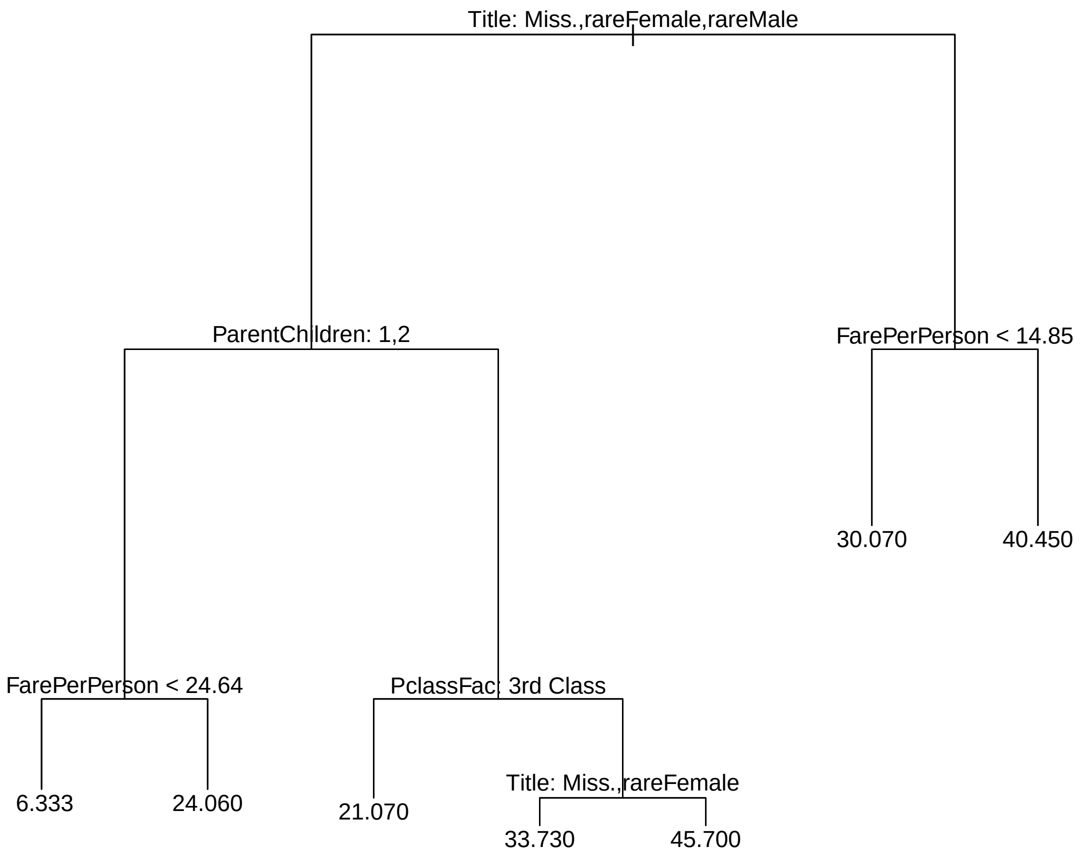
First we select features for modeling since we have many redundant features (such as the logged variables), and separate the data into sets with age and without age. These features were chosen after a bit of trial and error and plotting of the variable importance score generated by the random forest (see below), which allowed me to simplify the model by removing attributes that weren't helping the model. In short, the model was too complex and therefore there was too much variance.

```r
# choose features for modeling
chosen <- c("Age","SurvivedFac","PclassFac","Title","NameLength",
            "SiblingSpouse","ParentChildren","FarePerPerson","TicketCount")
yesAge <- train[!is.na(train$Age), colnames(train) %in% chosen] # with ages <- to train and evaluate mo
noAge <- train[is.na(train$Age), colnames(train) %in% chosen] # without ages <- to predict
# drop the outcome since it only has missing values
noAge$Age <- NULL
```

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

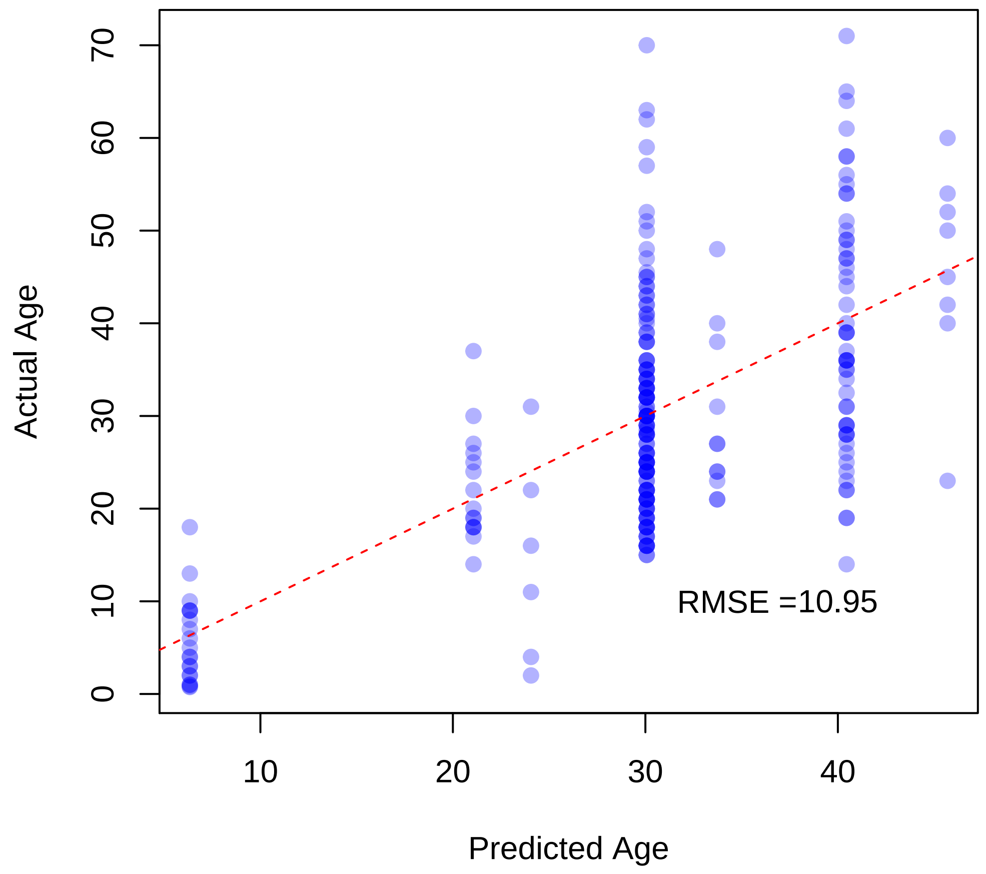
```r
library(tree)
library(caTools)
set.seed(1)
# split on outcome
Y_age <- yesAge[, "Age"]
age_bool <- sample.split(Y_age, SplitRatio = 2/3)
age_train <- yesAge[age_bool, ]
age_test <- yesAge[!age_bool, colnames(yesAge) != "Age"]
# fit model
```

```r
age_mod <- tree(Age~.,data=age_train)
# plot tree
plot(age_mod)
text(age_mod, pretty=0)
```



One problem with this single tree approach is that another random starting point would generate an entirely different tree. Let's how this single tree did as far as predicting ages in the test set:

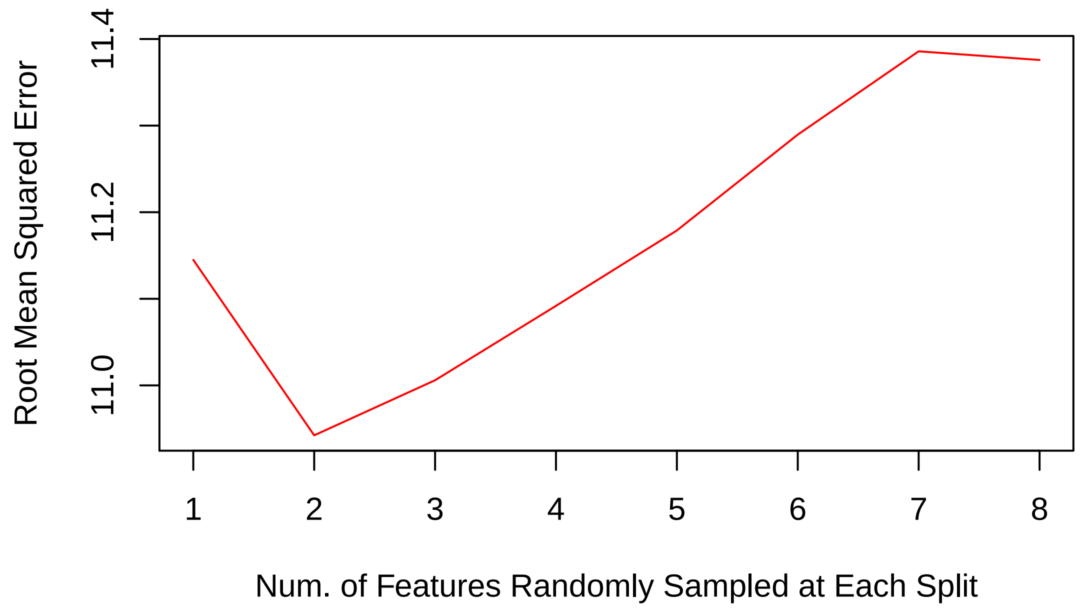
```r
y_hat <- predict(age_mod, newdata=age_test)
y_test <- yesAge[!age_bool, "Age"]
test_RMSE <- round(sqrt(mean((y_hat - y_test)^2)),2)
plot(y_hat, y_test,ylab="Actual Age",xlab="Predicted Age",pch=19,col=rgb(0,0,1,0.3))
text(c(35,40),10, c("RMSE = ", test_RMSE))
abline(0,1, col="red",lty=2)
```



The tree seems to be overpredicting specific ages like 30 and 40 and making lots of errors because of this. Let's see if an ensemble model like random forest performs better.

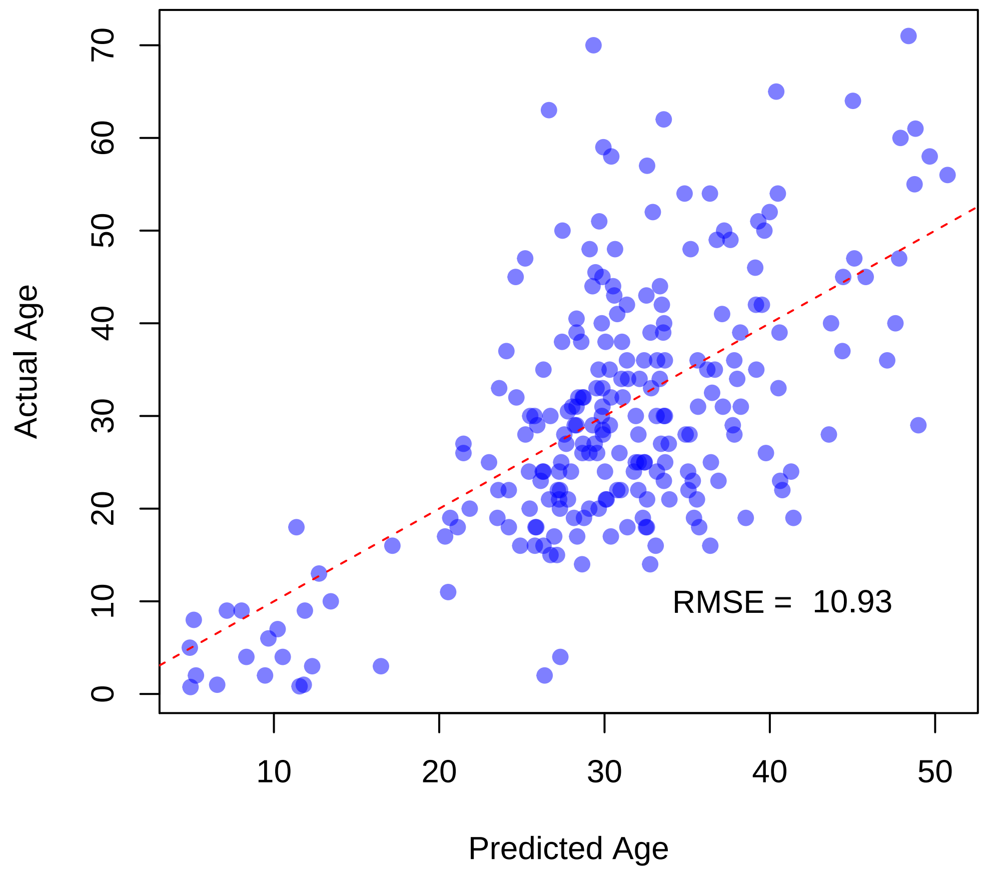
```
suppressMessages(library(randomForest))
# split on outcome
set.seed(1)
Y_age <- yesAge[, "Age"]
age_bool <- sample.split(Y_age, SplitRatio = 2/3)
age_train <- yesAge[age_bool, ]
age_test <- yesAge[!age_bool, colnames(yesAge) != "Age"]
y_test <- yesAge[!age_bool, "Age"]

# checking various RMSEs
rf_RMSEs <- vector("numeric", length=8)
for (i in 1:8) {
  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 the best RMSE is when we use 2 feastures to be randomly sampled at each split.

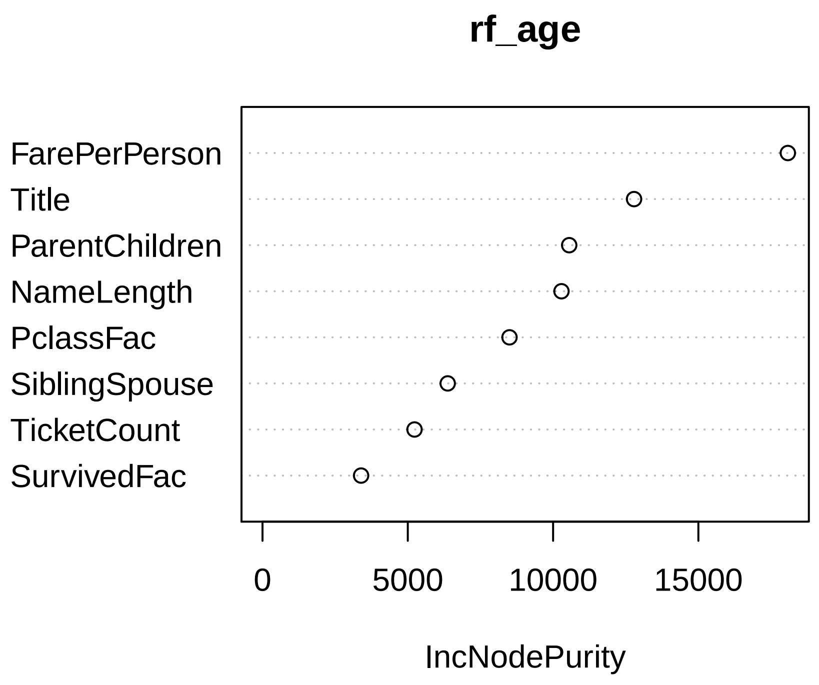
```
rf_age <- randomForest(Age ~., data=age_train, mtry=2, na.action=na.omit)
rf_yhat <- predict(rf_age, newdata=age_test)
test_RMSE <- round(sqrt(mean((rf_yhat - y_test)^2, na.rm=TRUE)), 2)
plot(rf_yhat, y_test,ylab="Actual Age",xlab="Predicted Age",pch=19,col=rgb(0,0,1,0.5))
text(c(38,45),10, c("RMSE = ", test_RMSE))
abline(0,1, col="red",lty=2)
```



The model seems to make lots of mistakes still but predictions seem more disperse and the RMSE is basically the same. I believe the random forest model will generalize better than a single tree (we might have gotten lucky with the RMSE) so I make predictions for the `noAge` data with this last ensemble model, imputing values and comparing the new, full `Age` distribution to our original distribution of ages.

For the record, this is how I determined which variables to remove from the overly complex first models I built:

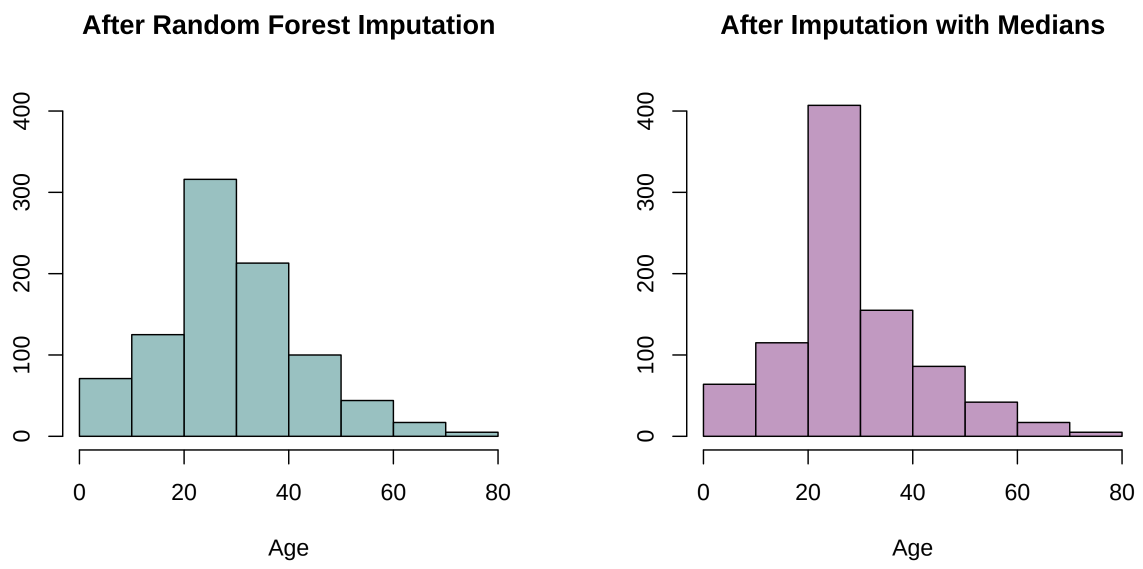
```
varImpPlot(rf_age)
```



```r
set.seed(1)
rf_age <- randomForest(Age ~., data=yesAge, mtry=2, 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:



We see that the random forest model performed a more sensible imputation than the imputation with medians, as the distribution more closely resembles that of the original `Age` variable.

Age Categories

We can also create an `AgeFac` variable that bins ages into the following groups: `Child` $(0-12)$ `Teen` $(13-19)$, `YoungAdult` $(20-35)$, `MiddleAged` $(36-55)$, and `Elderly` $(56-80)$.

```
# create AgeFac for Age Categories
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$Age < 56, "MiddleAged", "Elderly"))))
train$AgeFac <- factor(train$AgeFac)
train$AgeNum <- as.integer(train$Age)
train$Age <- NULL
```

Summary after pre-processing

First we reorder a bit the variables.

```
new_order <- c("Cabin","Embarked","PclassNum","PclassFac","NameLength","Title","IsMale",
               "GenderFac","SiblingSpouse","ParentChildren","NumRelatives","TicketCount",
               "FarePerPerson","FarePerPersonLog","AgeNum","AgeFac","SurvivedNum","SurvivedFac")
train <- train[,new_order]
head(train)
```

```
##   Cabin     Embarked PclassNum PclassFac NameLength Title IsMale GenderFac
## 1  <NA> Southhampton         3 3rd Class         23   Mr.      1      male
## 2     C    Cherbourg         1 1st Class         51  Mrs.      0    female
## 3  <NA> Southhampton         3 3rd Class         22 Miss.      0    female
## 4     C Southhampton         1 1st Class         44  Mrs.      0    female
## 5  <NA> Southhampton         3 3rd Class         24   Mr.      1      male
## 6  <NA>   Queensland         3 3rd Class         16   Mr.      1      male
##   SiblingSpouse ParentChildren NumRelatives TicketCount FarePerPerson
## 1             1              0            1           1          7.25
## 2             1              0            1           1         71.28
## 3             0              0            0           1          7.92
## 4             1              0            1           2         26.55
## 5             0              0            0           1          8.05
## 6             0              0            0           1          8.46
##   FarePerPersonLog AgeNum     AgeFac SurvivedNum SurvivedFac
## 1         2.110213     22 YoungAdult           0           0
## 2         4.280547     38 MiddleAged           1           1
## 3         2.188296     26 YoungAdult           1           1
## 4         3.316003     35 YoungAdult           1           1
## 5         2.202765     35 YoungAdult           0           0
## 6         2.247072     30 YoungAdult           0           0
```

```
# Summary after pre-processing
summary(train)
```

```
##   Cabin             Embarked     PclassNum          PclassFac
##  A   : 15   Cherbourg   :168   Min.   :1.000   1st Class:216
##  B   : 47   Queensland  : 77   1st Qu.:2.000   2nd Class:184
##  C   : 59   Southhampton:646   Median :3.000   3rd Class:491
##  D   : 33                      Mean   :2.309
##  E   : 32                      3rd Qu.:3.000
##  F   : 18                      Max.   :3.000
##  NA's:687
##    NameLength           Title         IsMale        GenderFac
##  Min.   :12.00   Miss.     :182   Min.   :0.0000   female:314
##  1st Qu.:20.00   Mr.       :517   1st Qu.:0.0000   male  :577
##  Median :25.00   Mrs.      :125   Median :1.0000
##  Mean   :26.97   rareFemale:  6   Mean   :0.6476
##  3rd Qu.:30.00   rareMale  : 61   3rd Qu.:1.0000
##  Max.   :82.00                    Max.   :1.0000
##
##  SiblingSpouse ParentChildren  NumRelatives  TicketCount
##  0:608         0:678          0      :537   Min.   :1.000
##  1:209         1:118          1      :161   1st Qu.:1.000
##  2: 28         2: 80          2      :102   Median :1.000
##  3: 16         3:  5          3      : 29   Mean   :1.788
##  4: 18         4:  4          5      : 22   3rd Qu.:2.000
##  5:  5         5:  5          4      : 15   Max.   :7.000
##  8:  7         6:  1          (Other): 25
##  FarePerPerson     FarePerPersonLog     AgeNum             AgeFac
##  Min.   :  0.000   Min.   :0.000    Min.   : 0.00   Child     : 76
##  1st Qu.:  7.765   1st Qu.:2.171    1st Qu.:21.00   Elderly   : 39
##  Median :  8.850   Median :2.287    Median :29.00   MiddleAged:210
##  Mean   : 17.789   Mean   :2.600    Mean   :29.68   Teen      :104
##  3rd Qu.: 24.290   3rd Qu.:3.229    3rd Qu.:37.00   YoungAdult:462
##  Max.   :221.780   Max.   :5.406    Max.   :80.00
##
##   SurvivedNum     SurvivedFac
##  Min.   :0.0000   0:549
##  1st Qu.:0.0000   1:342
##  Median :0.0000
##  Mean   :0.3838
##  3rd Qu.:1.0000
##  Max.   :1.0000
##
```

Class representation in `Cabin` is trouble free although there are a lot of NAs. Class representation in general is not a problem, except perhaps for the rareFemale level in `Title` which we might drop in Part 2 of the project. There are more males than females yet as we shall see, females survived a lot more. The distribution of the number of relative variables is skewed, as well as `FarePerPerson`. A good 38% of the people in the training data survived, so it should not be hard to predict and we do not need to implement SMOTE or other method of balancing classes.

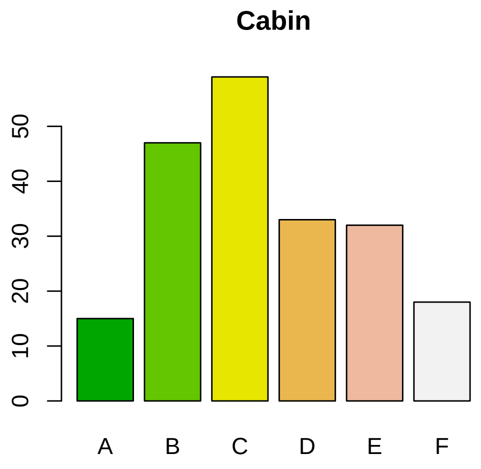
Univariate Graphical EDA

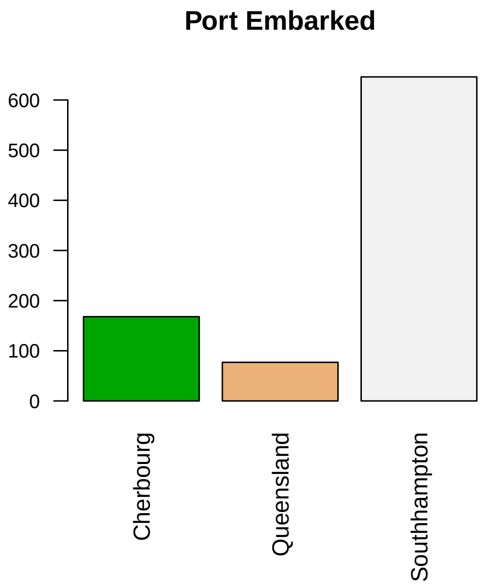
Now that we have the data in a basic shape for graphical EDA, we can try understanding the underlying distributions and associations of this training set better, remembering that this is just a sample so our findings are not necessarily representative of the population (one hopes that the creators of the Titanic competition in Kaggle used proper random sampling techniques in splitting their train and test samples).

```r
names(train)
```

```
##  [1] "Cabin"          "Embarked"         "PclassNum"
##  [4] "PclassFac"      "NameLength"       "Title"
##  [7] "IsMale"         "GenderFac"        "SiblingSpouse"
## [10] "ParentChildren" "NumRelatives"     "TicketCount"
## [13] "FarePerPerson"  "FarePerPersonLog" "AgeNum"
## [16] "AgeFac"         "SurvivedNum"      "SurvivedFac"
```

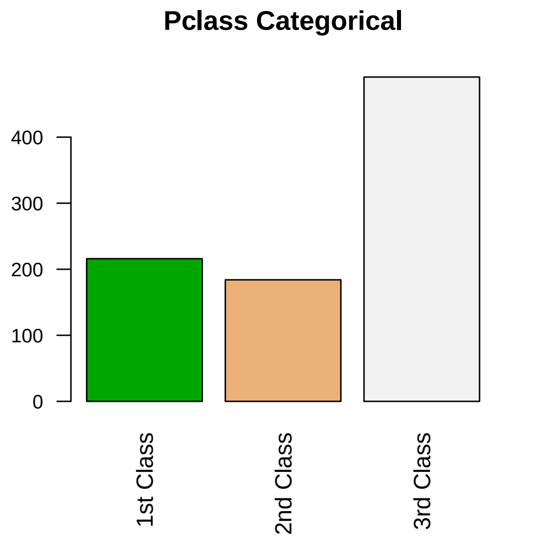
Cabin, Embarked

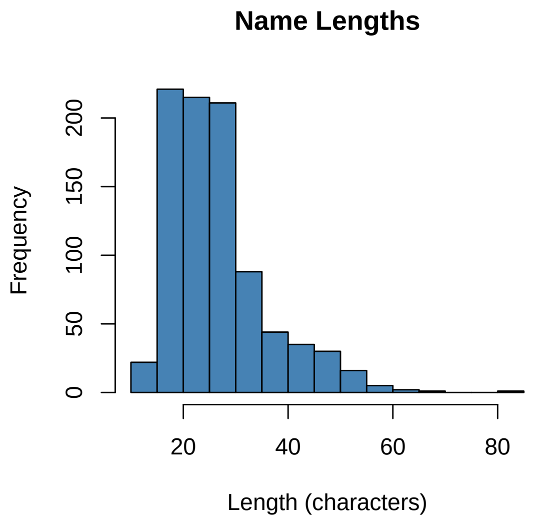




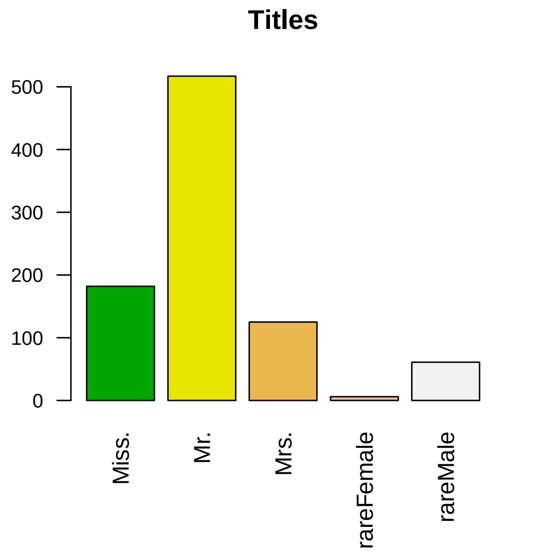
Most people were in Cabin C and yet the distribution is not too skewed, while a vast majority embarked in Southhampton.

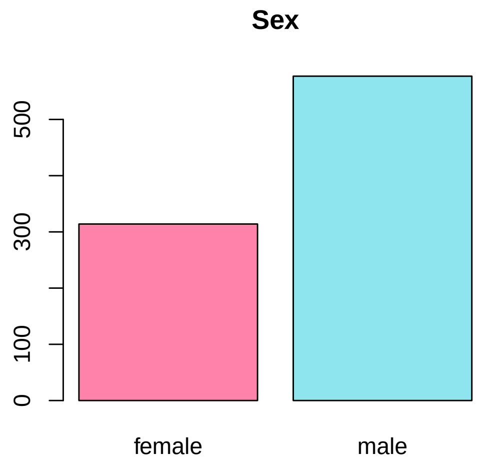
PclassFac, NameLength





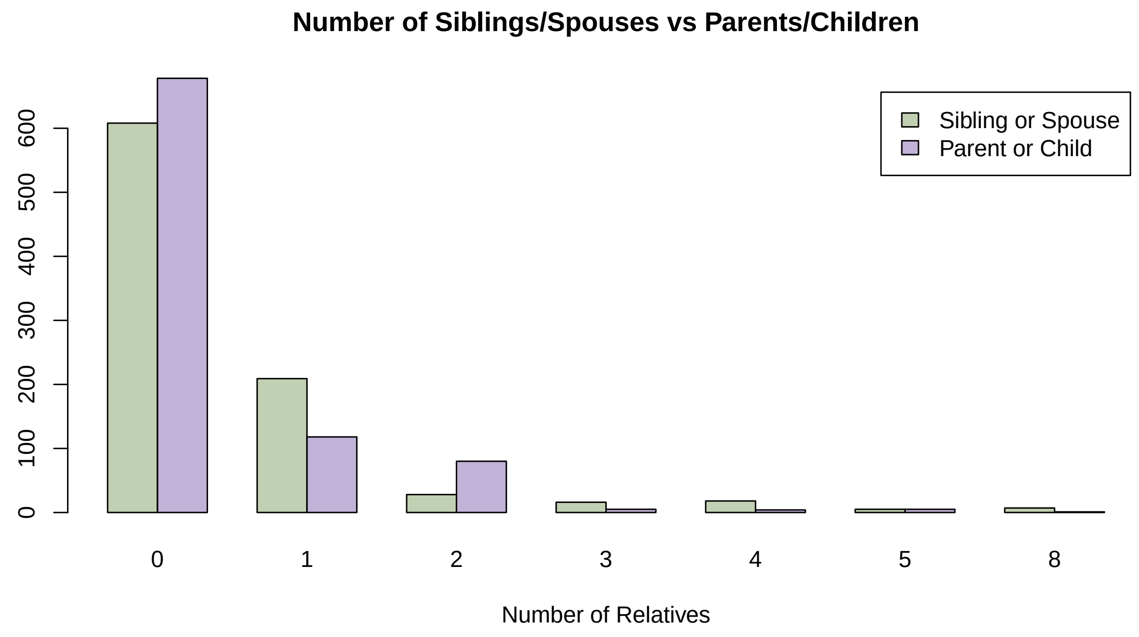
Title, GenderFac





As noted, rareFemale is under represented. There is a class imbalance in the male and female proportions as well but it is not so sever as to warrant any special treatment.

SiblingSpouse, ParentChildren



NumbRelatives, TicketCount

FarePerPerson, FarePerPersonLog

AgeNum, AgeFac

SurvivedNum, SurvivedFac