Titanic

Jordan Ziskin

5/20/2021

Let's start by reading in the data and looking at its dimensions and summary.

```
# Read in data
train <- read.csv("data/train.csv")
test <- read.csv("data/test.csv")

# View dimensions
dim(train)
```

```
## [1] 891  12
```

```
dim(test)
```

```
## [1] 418  11
```

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

```
head(test)
```

```
##   PassengerId Pclass                                          Name    Sex  Age
## 1         892      3                              Kelly, Mr. James   male 34.5
## 2         893      3              Wilkes, Mrs. James (Ellen Needs) female 47.0
## 3         894      2                     Myles, Mr. Thomas Francis   male 62.0
## 4         895      3                              Wirz, Mr. Albert   male 27.0
## 5         896      3 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female 22.0
## 6         897      3                    Svensson, Mr. Johan Cervin   male 14.0
##   SibSp Parch  Ticket    Fare Cabin Embarked
## 1     0     0  330911  7.8292              Q
## 2     1     0  363272  7.0000              S
## 3     0     0  240276  9.6875              Q
## 4     0     0  315154  8.6625              S
## 5     1     1 3101298 12.2875              S
## 6     0     0    7538  9.2250              S
```

```
# Assign ID to variables and remove from dataset
id.train <- train$PassengerId
id.test <- test$PassengerId
train <- train[,-1]
test <- test[,-1]
```

Our training dataset consists of 891 observations, and 12 columns. Of the 12 columns, 1 is the PassengerID, 1 is the response variable "Survived", and the remaining 10 are predictors. The testing dataset has 418 observations with 11 columns. The missing column from the testing dataset is, of course, the response variable "Survived" which we will be predicting here. The ID column was removed from the training and testing datasets and stored as vectors. Before we look at the variables more closely, it is important to notice that we have missing values in the age column labeled as NA, but we also have missing values in other columns, such as the cabin column, that are blank. Let's replace these values with NA so we don't forget to later.

```
# Replace empty spaces with NA
train <- as.data.frame(apply(train, 2, function(x) gsub("^$|^ $", NA, x)))
test <- as.data.frame(apply(test, 2, function(x) gsub("^$|^ $", NA, x)))

# Maintain numeric columns
for (col in c("Pclass", "Age", "SibSp", "Parch", "Fare")){
    train[,col] <- as.numeric(train[,col])
    test[,col] <- as.numeric(test[,col])
}
```

Let's look at each column individually and see what we can learn.

```
# View survival table
table(train$Survived)
```

```
##
##   0   1
## 549 342
```

```
cat(paste0("Only ", round(table(train$Survived)[2]/sum(table(train$Survived))*100,2),
           "% of passengers in the training set survived."))
```

```
## Only 38.38% of passengers in the training set survived.
```

```
# View Pclass table
table(train$Pclass)
```

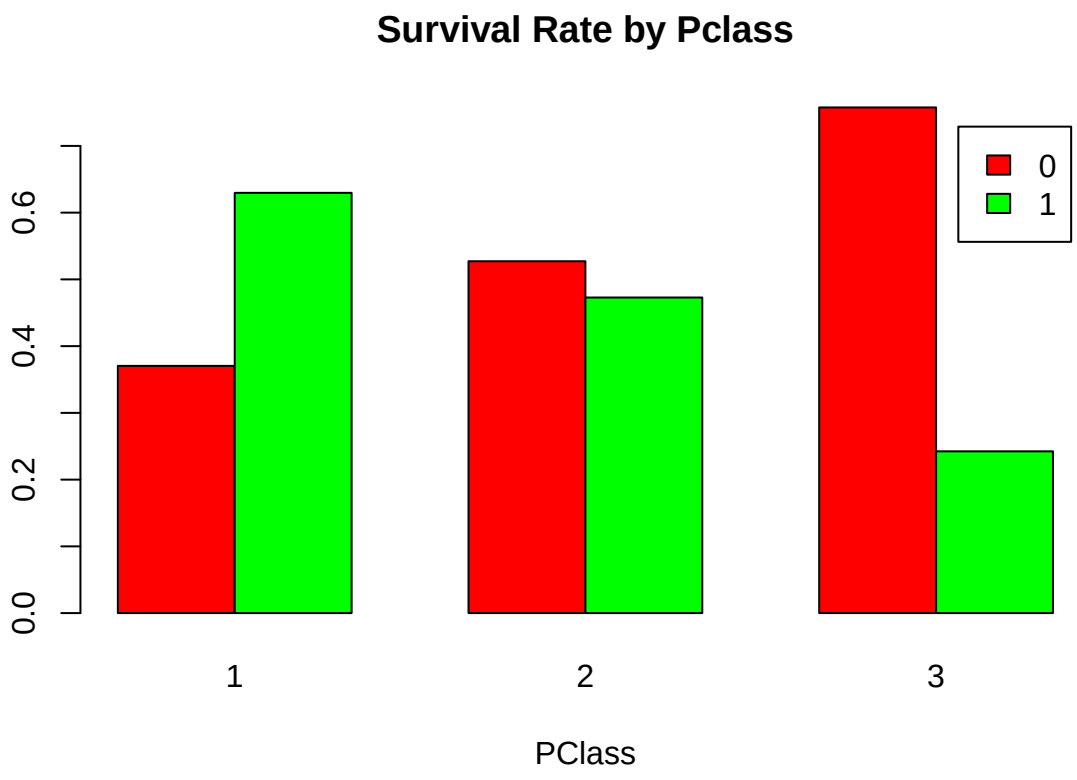
```
##
##   1   2   3
## 216 184 491
```

```
table(test$Pclass)
```

```
##
##   1   2   3
## 107  93 218
```

The number of passengers in Pclass 1 and 2 are approximately the same in both datasets. Additionally, in both datasets the number of passengers in Pclass 3 is double the amount in Pclass 1 or 2.

```
# View survival plot
counts <- table(train$Survived, train$Pclass)
barplot(prop.table(counts,2), main="Survival Rate by Pclass",
  xlab="PClass", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```



We can clearly see that Pclass is indirectly correlated with survival. For that reason, keeping Pclass as a continuous variable seems best. Next we'll look at Sex.

```
# View Sex table
table(train$Sex)
```

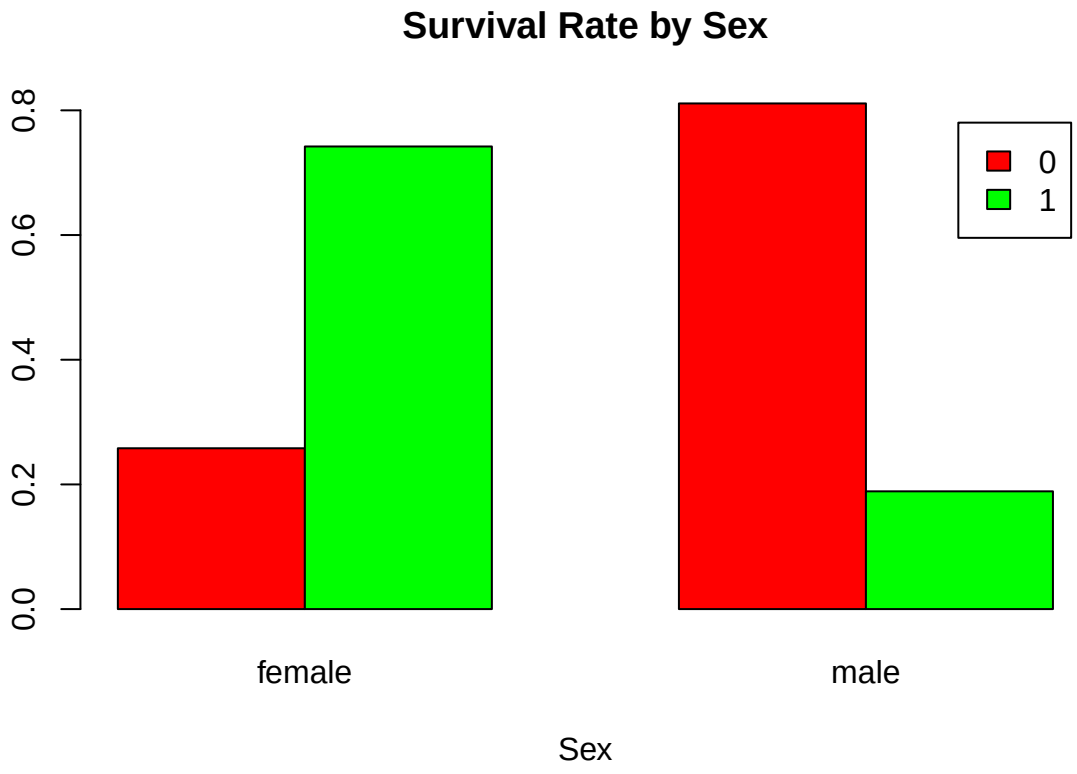
```
##
## female   male
##    314    577
```

```
table(test$Sex)
```

```
##
## female   male
##    152    266
```

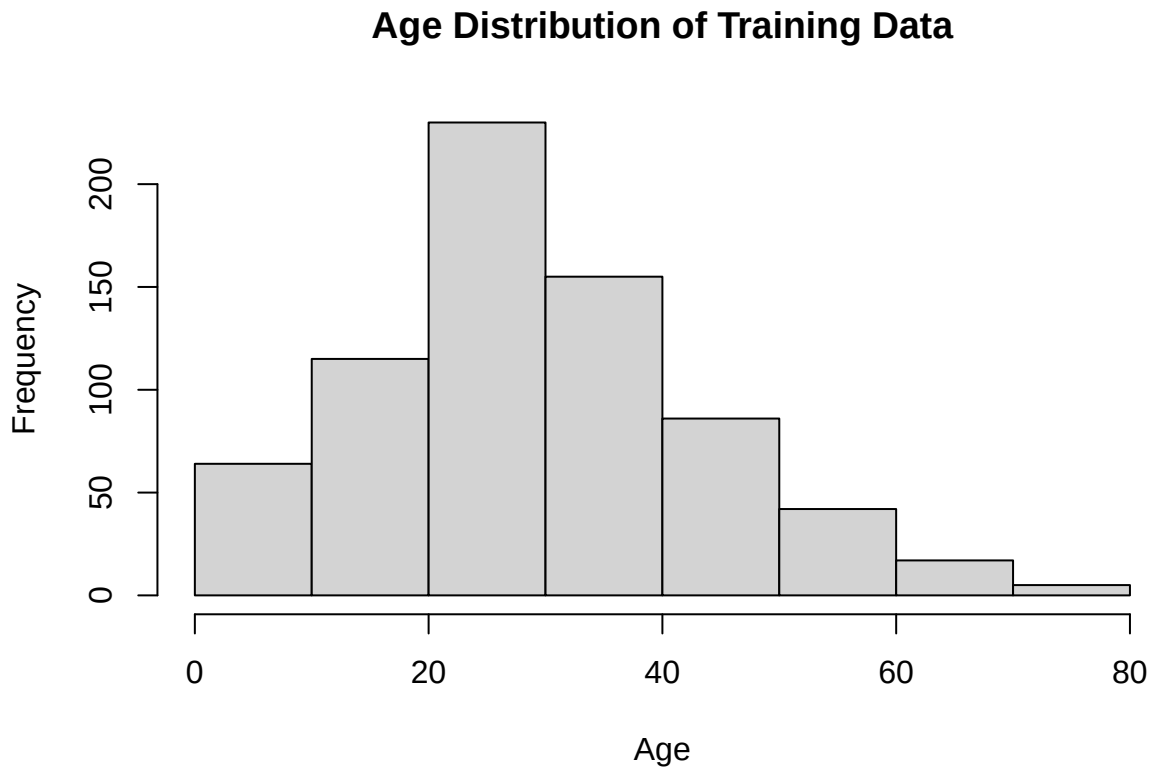
Just as we saw with Pclass, the proportion of males in the training and testing datasets are proportional with the number males being just under double the number of females.

```
# View survival plot
counts <- table(train$Survived, train$Sex)
barplot(prop.table(counts,2), main="Survival Rate by Sex",
  xlab="Sex", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```

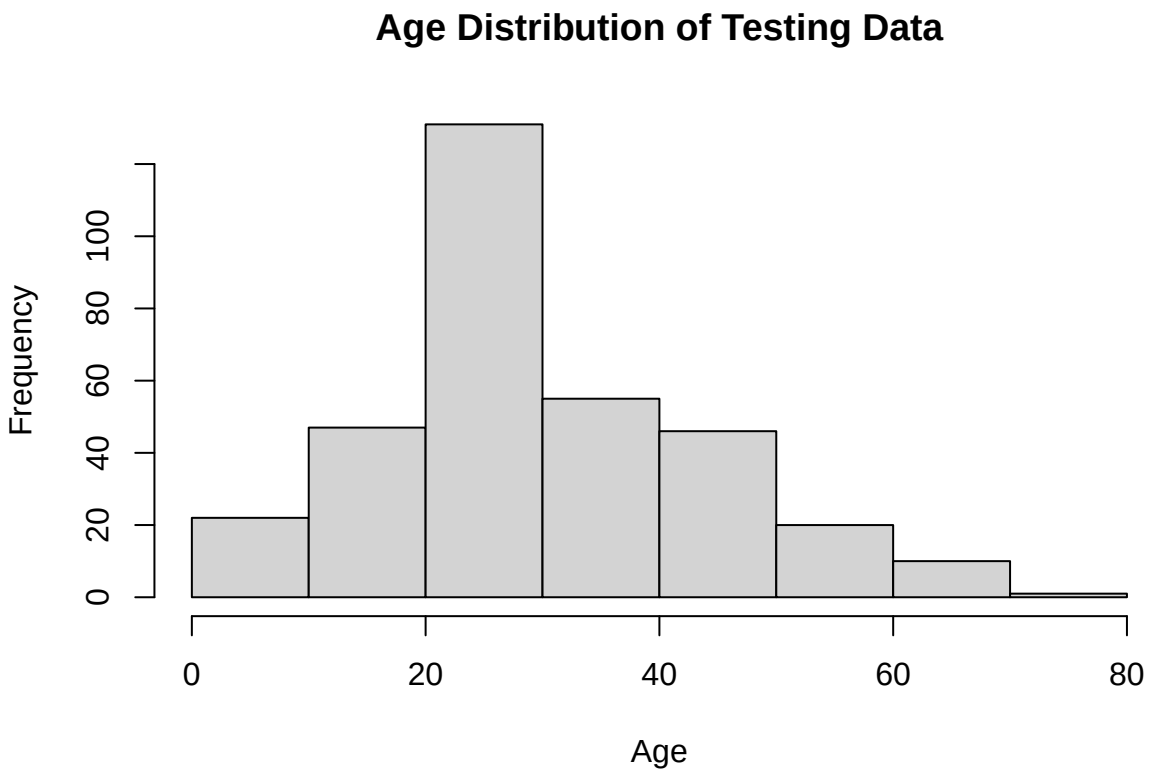


As expected, the survival rate of females is significantly larger than the survival rate of males. This suggests that sex is going to be a very strong indicator in determining survival.

```
# View histogram of ages
hist(train$Age, main="Age Distribution of Training Data", xlab="Age")
```

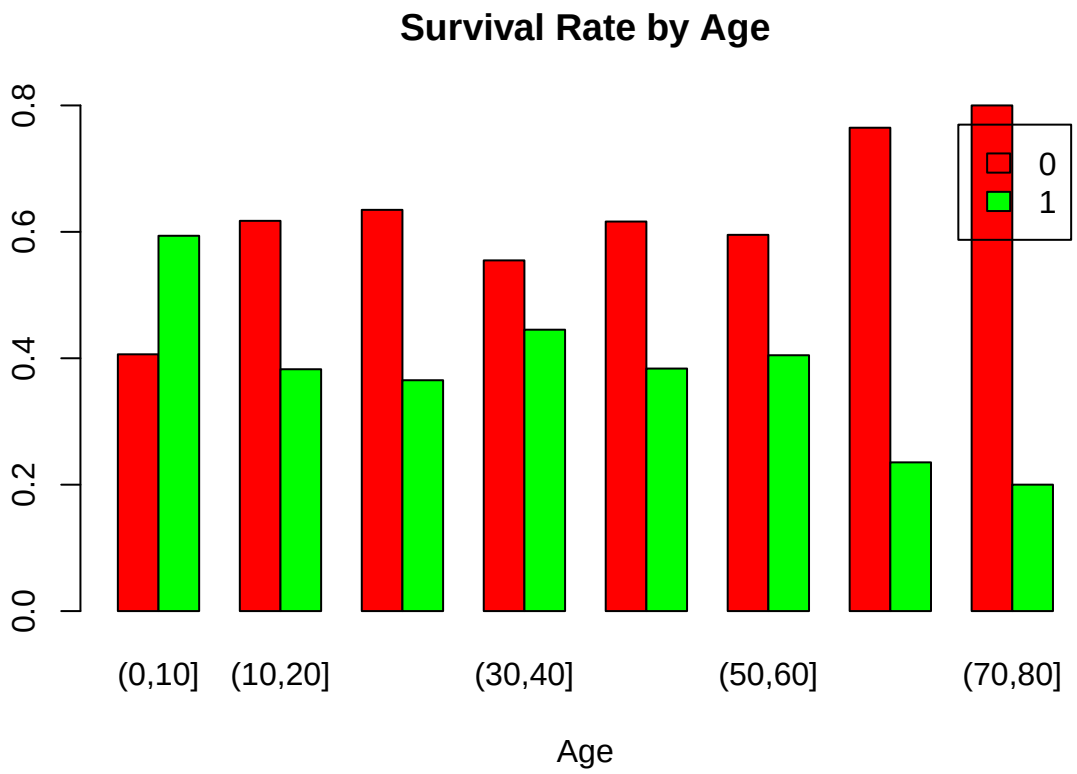



```
hist(test$Age, main="Age Distribution of Testing Data", xlab="Age")
```



We can see the distributions of ages on the titanic are relatively normal with the peak reached at 20 - 30 years old. The distributions between sets is consistent with there being a slightly higher peak in the 20 - 30 year old in the testing set.

```
# View survival plot
counts <- table(train$Survived, cut(train$Age, breaks=seq(0,80, by=10)))
barplot(prop.table(counts,2), main="Survival Rate by Age",
  xlab="Age", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```



Looking at the survival rate by age, we can see that as age increase, the survival rate tends to decrease. While age may not be the strongest indicator, I expect it will still be helpful when combined with the other predictors. Unfortunately, we have a lot of missing values in both the data

```
# Print missing values
cat(paste("There are", sum(is.na(train$Age)),
          "missing values in the training dataset."), "\n")
```

```
## There are 177 missing values in the training dataset.
```

```
cat(paste("There are", sum(is.na(test$Age)), "missing values in the testing dataset."))
```

```
## There are 86 missing values in the testing dataset.
```

We will use machine learning models to best predict these missing values a little later. Next we'll look at the variables Sibsp and Parch. The distinction between these two categories seem somewhat arbitrary. Why would the numbers of siblings and spouse be together in one category while parents and children be together in another category. It may be helpful to combine these variables into one variable called FamSize for family size. Let's look more closely at this combined family size variable.

```
# View family size table
table(train$SibSp+train$Parch)
```

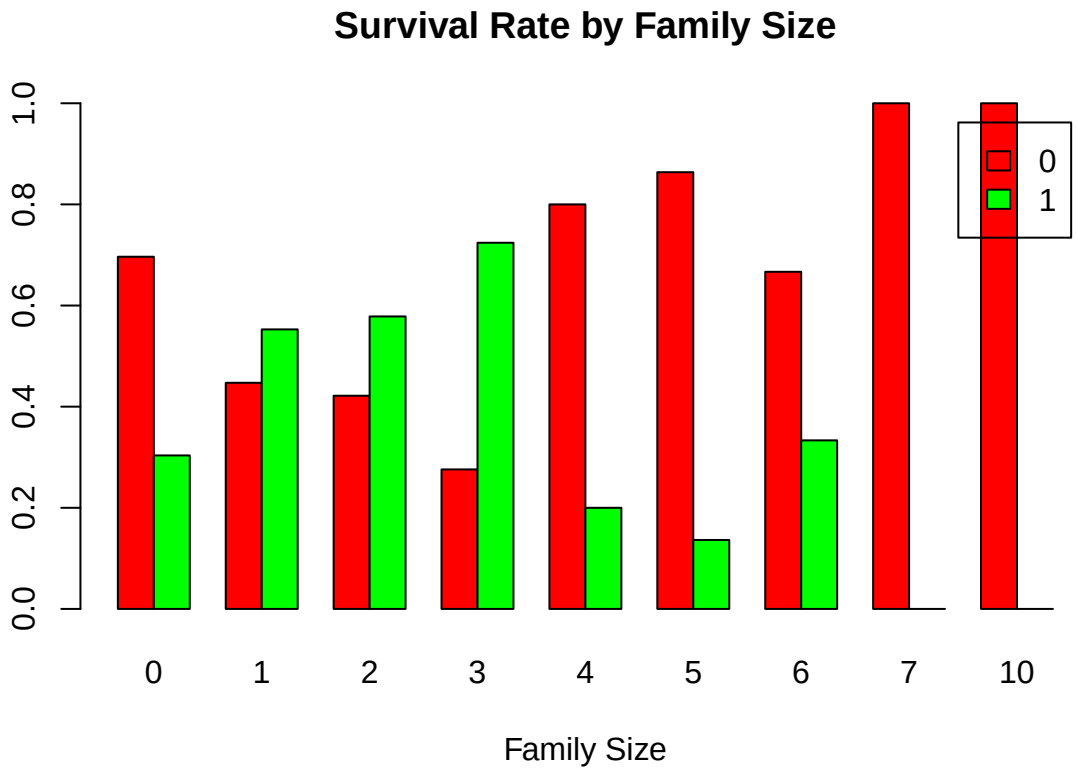
```
##
##   0   1   2   3   4   5   6   7  10
## 537 161 102  29  15  22  12   6   7
```

```
table(test$SibSp+test$Parch)
```

```
##
##   0   1   2   3   4   5   6   7  10
## 253  74  57  14   7   3   4   2   4
```

Family size appears to be proportional between training and testing datasets. Most passengers have no family members aboard and very few passengers had 3 or more family members onboard.

```
# View survival plot
counts <- table(train$Survived, train$SibSp+train$Parch)
barplot(prop.table(counts,2), main="Survival Rate by Family Size",
  xlab="Family Size", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```



There seems to be a somewhat quadratic relationship between survival rate and family size with very large and very small family sizes incurring the greatest number of deaths. For this reason it may be more helpful to make FamSize a categorical variable with 3 categories, "Solo" (FamSize=0), "Nuclear" (FamSize=1-3) and "Large" (FamSize > 3).

```
# Add FamSize to training and testing datasets
train.fs <- train$SibSp + train$Parch
test.fs <- test$SibSp + test$Parch
train$FamSize <- ifelse(train.fs == 0, "Solo", ifelse(0 < train.fs & train.fs < 4,
                                                      "Nuclear", "Large"))
```

```
test$FamSize <- ifelse(test.fs == 0, "Solo", ifelse(0 < test.fs & test.fs < 4,
                                                    "Nuclear", "Large"))

# Remove SipSp and Parch from training and testing datasets
train <- train[,-which(names(train) %in% c("SibSp", "Parch"))]
test <- test[,-which(names(test) %in% c("SibSp", "Parch"))]
```

Next we'll look at Embarked.

```
# View embarked table
table(train$Embarked)
```

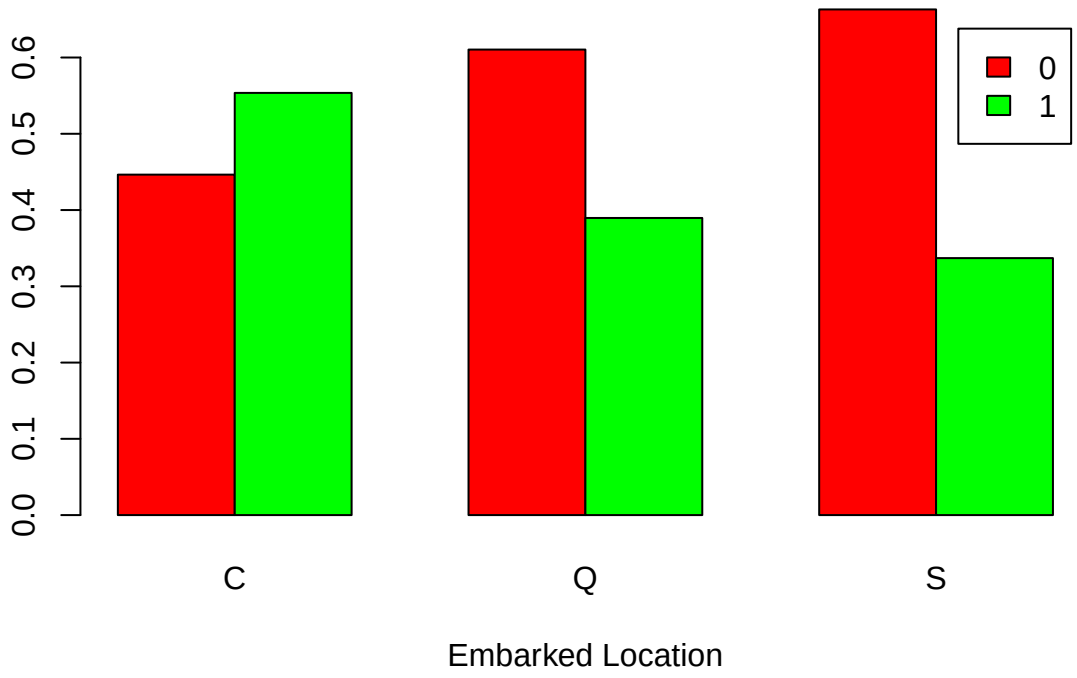
```
##
##   C   Q   S
## 168  77 644
```

```
table(test$Embarked)
```

```
##
##   C   Q   S
## 102  46 270
```

We see that while the distribution is relatively proportional, the training set seems to have a larger proportion of "S"'s than the testing set.

```
# View survival plot
counts <- table(train$Survived, train$Embarked)
barplot(prop.table(counts,2), main="Survival Rate by Embarked Location",
  xlab="Embarked Location", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```



Passengers who embarked from location C seems to have a net positive survival rate, while those who embarked from locations Q and S have a net negative survival rate. Let's look at the Cabin column.

```
# View unique cabin values
sort(unique(train$Cabin))
```

```
##   [1] "A10"            "A14"            "A16"            "A19"
##   [5] "A20"            "A23"            "A24"            "A26"
##   [9] "A31"            "A32"            "A34"            "A36"
##  [13] "A5"             "A6"             "A7"             "B101"
##  [17] "B102"           "B18"            "B19"            "B20"
##  [21] "B22"            "B28"            "B3"             "B30"
##  [25] "B35"            "B37"            "B38"            "B39"
##  [29] "B4"             "B41"            "B42"            "B49"
##  [33] "B5"             "B50"            "B51 B53 B55"    "B57 B59 B63 B66"
##  [37] "B58 B60"        "B69"            "B71"            "B73"
##  [41] "B77"            "B78"            "B79"            "B80"
##  [45] "B82 B84"        "B86"            "B94"            "B96 B98"
##  [49] "C101"           "C103"           "C104"           "C106"
##  [53] "C110"           "C111"           "C118"           "C123"
##  [57] "C124"           "C125"           "C126"           "C128"
##  [61] "C148"           "C2"             "C22 C26"        "C23 C25 C27"
##  [65] "C30"            "C32"            "C45"            "C46"
##  [69] "C47"            "C49"            "C50"            "C52"
##  [73] "C54"            "C62 C64"        "C65"            "C68"
##  [77] "C7"             "C70"            "C78"            "C82"
```

```
##  [81] "C83"             "C85"             "C86"             "C87"
##  [85] "C90"             "C91"             "C92"             "C93"
##  [89] "C95"             "C99"             "D"               "D10 D12"
##  [93] "D11"             "D15"             "D17"             "D19"
##  [97] "D20"             "D21"             "D26"             "D28"
## [101] "D30"             "D33"             "D35"             "D36"
## [105] "D37"             "D45"             "D46"             "D47"
## [109] "D48"             "D49"             "D50"             "D56"
## [113] "D6"              "D7"              "D9"              "E10"
## [117] "E101"            "E12"             "E121"            "E17"
## [121] "E24"             "E25"             "E31"             "E33"
## [125] "E34"             "E36"             "E38"             "E40"
## [129] "E44"             "E46"             "E49"             "E50"
## [133] "E58"             "E63"             "E67"             "E68"
## [137] "E77"             "E8"              "F E69"           "F G63"
## [141] "F G73"           "F2"              "F33"             "F38"
## [145] "F4"              "G6"              "T"
```

```
sort(unique(test$Cabin))
```

```
##  [1] "A11"             "A18"             "A21"             "A29"
##  [5] "A34"             "A9"              "B10"             "B11"
##  [9] "B24"             "B26"             "B36"             "B41"
## [13] "B45"             "B51 B53 B55"     "B52 B54 B56"     "B57 B59 B63 B66"
## [17] "B58 B60"         "B61"             "B69"             "B71"
## [21] "B78"             "C101"            "C105"            "C106"
## [25] "C116"            "C130"            "C132"            "C22 C26"
## [29] "C23 C25 C27"     "C28"             "C31"             "C32"
## [33] "C39"             "C46"             "C51"             "C53"
## [37] "C54"             "C55 C57"         "C6"              "C62 C64"
## [41] "C7"              "C78"             "C80"             "C85"
## [45] "C86"             "C89"             "C97"             "D"
## [49] "D10 D12"         "D15"             "D19"             "D21"
## [53] "D22"             "D28"             "D30"             "D34"
## [57] "D37"             "D38"             "D40"             "D43"
## [61] "E31"             "E34"             "E39 E41"         "E45"
## [65] "E46"             "E50"             "E52"             "E60"
## [69] "F"               "F E46"           "F E57"           "F G63"
## [73] "F2"              "F33"             "F4"              "G6"
```

```
# Print missing values
cat(paste("There are", sum(is.na(train$Cabin)),
          "missing values in the training dataset."), "\n")
```

```
## There are 687 missing values in the training dataset.
```

```
cat(paste("There are", sum(is.na(test$Cabin)),
          "missing values in the testing dataset."))
```

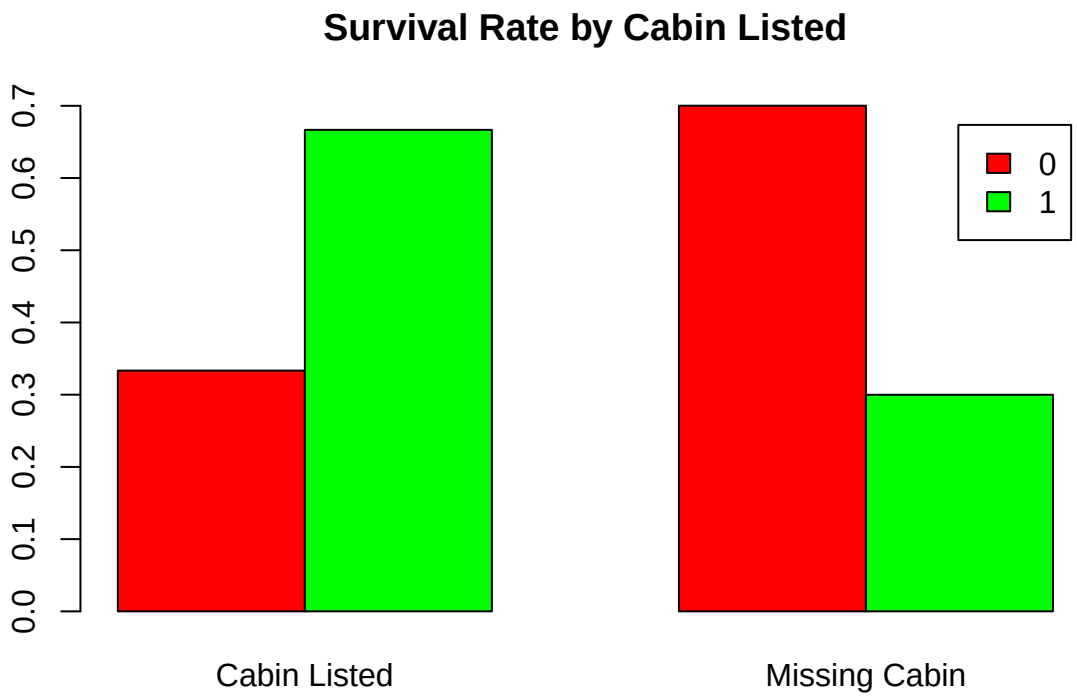
```
## There are 327 missing values in the testing dataset.
```

We notice a few things looking at this data.

1. Most people do not have a cabin listed.
2. Cabin names are typically a letter A-G, followed by a number
3. Some entries have more than 1 cabin listed.
4. Some entries have a letter listed and then a cabin with another letter like “F G73”, “F E57”, and “F E69”
5. In the training dataset we have one cabin listed as “T”

How we deal with these findings is going to take some judgment. Because the vast majority of cabins are missing, we have to concede that it is very unlikely that we will have enough information to predict the cabin of the missing entries. So with that said, we should first consider if there is any difference between the entries for which cabin is named and cabin isn’t named. If there is no difference, it may be worth throwing out the column entirely.

```
counts <- table(train$Survived, is.na(train$Cabin))
colnames(counts) <- c("Cabin Listed", "Missing Cabin")

barplot(prop.table(counts,2), main="Survival Rate by Cabin Listed",
        col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```



We can see that for the entries for which the cabin is listed, the survival rate was significantly higher. This could be due to survivor bias; those who survived the titanic were able to live to tell their cabin number. Or perhaps there is another reason. Regardless, it seems there is merit to keeping this variable. Let’s try to break cabin down even more by using the first letter of the cabin number. Above, we noticed that there were a handful of entries that had the letter F followed by a space and then another non-F cabin like “F G73”, “F E57”, and “F E69.” Below we will list these entries and then we’ll remove the F and the space

```
# Show entries with multiple characters
train[grepl("[A-Z] +", train$Cabin),]
```

```
##     Survived Pclass                                          Name    Sex Age
## 76         0      3                       Moen, Mr. Sigurd Hansen   male  25
## 129        1      3                             Peter, Miss. Anna female  NA
## 700        0      3   Humblen, Mr. Adolf Mathias Nicolai Olsen   male  42
## 716        0      3 Soholt, Mr. Peter Andreas Lauritz Andersen   male  19
##     Ticket    Fare Cabin Embarked FamSize
## 76  348123  7.6500 F G73        S    Solo
## 129   2668 22.3583 F E69        C Nuclear
## 700 348121  7.6500 F G63        S    Solo
## 716 348124  7.6500 F G73        S    Solo
```

```
test[grepl("[A-Z] +", test$Cabin),]
```

```
##     Pclass                           Name  Sex Age Ticket   Fare Cabin Embarked
## 58       3 Abelseth, Mr. Olaus Jorgensen male  25 348122 7.6500 F G63        S
## 289      3       Mardirosian, Mr. Sarkis male  NA   2655 7.2292 F E46        C
## 322      3         Krekorian, Mr. Neshan male  25   2654 7.2292 F E57        C
##     FamSize
## 58     Solo
## 289    Solo
## 322    Solo
```

```
# Remove first two characters
remove_first_2_chars <- function(x){
    sub('..', '', x)
}

# Remove F and space characters from multi-lettered cabin entries
train[grepl("[A-Z] +", train$Cabin),
      which(names(train)=="Cabin")] <- sapply(train[grepl("[A-Z] +", train$Cabin),
                                                    which(names(train)=="Cabin")],
                                              remove_first_2_chars)
test[grepl("[A-Z] +", test$Cabin),
     which(names(test)=="Cabin")] <- sapply(test[grepl("[A-Z] +", test$Cabin),
                                                   which(names(test)=="Cabin")],
                                             remove_first_2_chars)

# View same rows
train[c(76,129,700,716),]
```

```
##     Survived Pclass                                          Name    Sex Age
## 76         0      3                       Moen, Mr. Sigurd Hansen   male  25
## 129        1      3                             Peter, Miss. Anna female  NA
## 700        0      3   Humblen, Mr. Adolf Mathias Nicolai Olsen   male  42
## 716        0      3 Soholt, Mr. Peter Andreas Lauritz Andersen   male  19
##     Ticket    Fare Cabin Embarked FamSize
## 76  348123  7.6500   G73        S    Solo
## 129   2668 22.3583   E69        C Nuclear
## 700 348121  7.6500   G63        S    Solo
## 716 348124  7.6500   G73        S    Solo
```

```
test[c(58,289,322),]
```

```
##     Pclass                          Name  Sex Age Ticket   Fare Cabin Embarked
## 58       3 Abelseth, Mr. Olaus Jorgensen male  25 348122 7.6500   G63        S
## 289      3       Mardirosian, Mr. Sarkis male  NA   2655 7.2292   E46        C
## 322      3         Krekorian, Mr. Neshan male  25   2654 7.2292   E57        C
##     FamSize
## 58     Solo
## 289    Solo
## 322    Solo
```

Additionally, let's replace the cabin marked "T" with NA.

```
# Show T entry
train[grepl("T", train$Cabin),]
```

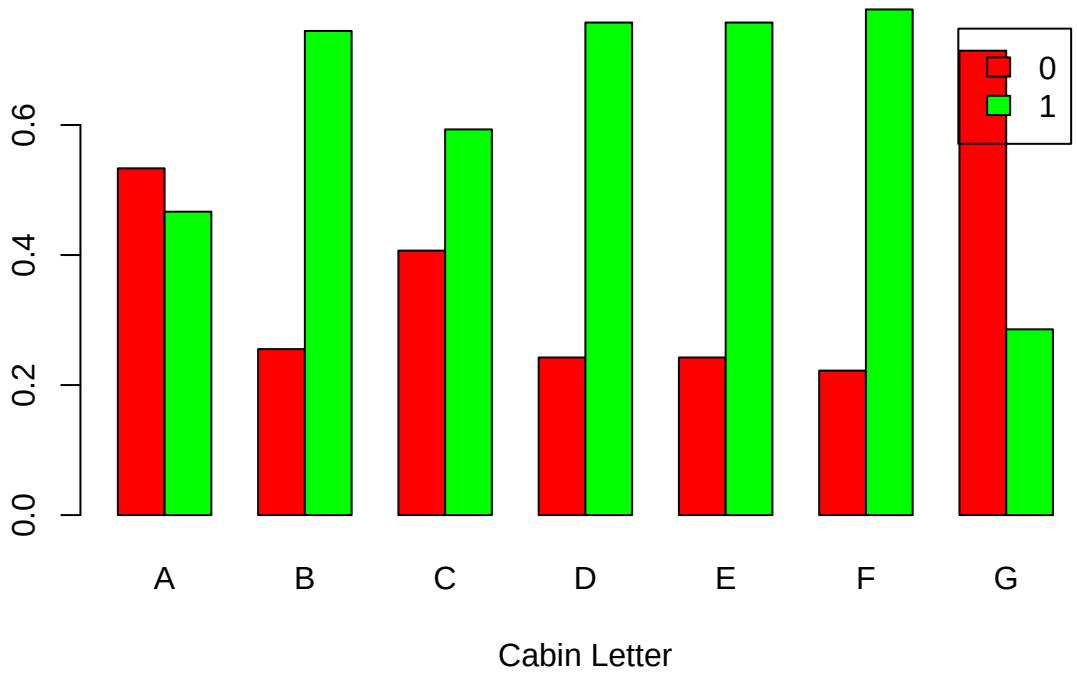
```
##     Survived Pclass                          Name  Sex Age Ticket Fare Cabin
## 340        0      1 Blackwell, Mr. Stephen Weart male  45 113784 35.5     T
##     Embarked FamSize
## 340        S    Solo
```

```
# Replace with NA
train[grepl("T", train$Cabin),"Cabin"] <- NA

# View same row
train[340,]
```

```
##     Survived Pclass                          Name  Sex Age Ticket Fare Cabin
## 340        0      1 Blackwell, Mr. Stephen Weart male  45 113784 35.5  <NA>
##     Embarked FamSize
## 340        S    Solo
```

Now that every cabin entry is either NA or begins with a letter A-G, we will create a new row called CabinLetter which takes the first letter of the cabin.

```
# Create CabinLetter column
train$CabinLetter <- sapply(train$Cabin, substr,1,1)
test$CabinLetter <- sapply(test$Cabin, substr,1,1)

# View survival plot
counts <- table(train$Survived, train$CabinLetter)
barplot(prop.table(counts,2), main="Survival Rate by Cabin Letter",
  xlab="Cabin Letter", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```




```
# Replace NAs with "Unknown"
train[is.na(train$CabinLetter),"CabinLetter"] <- "Unknown"
test[is.na(test$CabinLetter),"CabinLetter"] <- "Unknown"
```

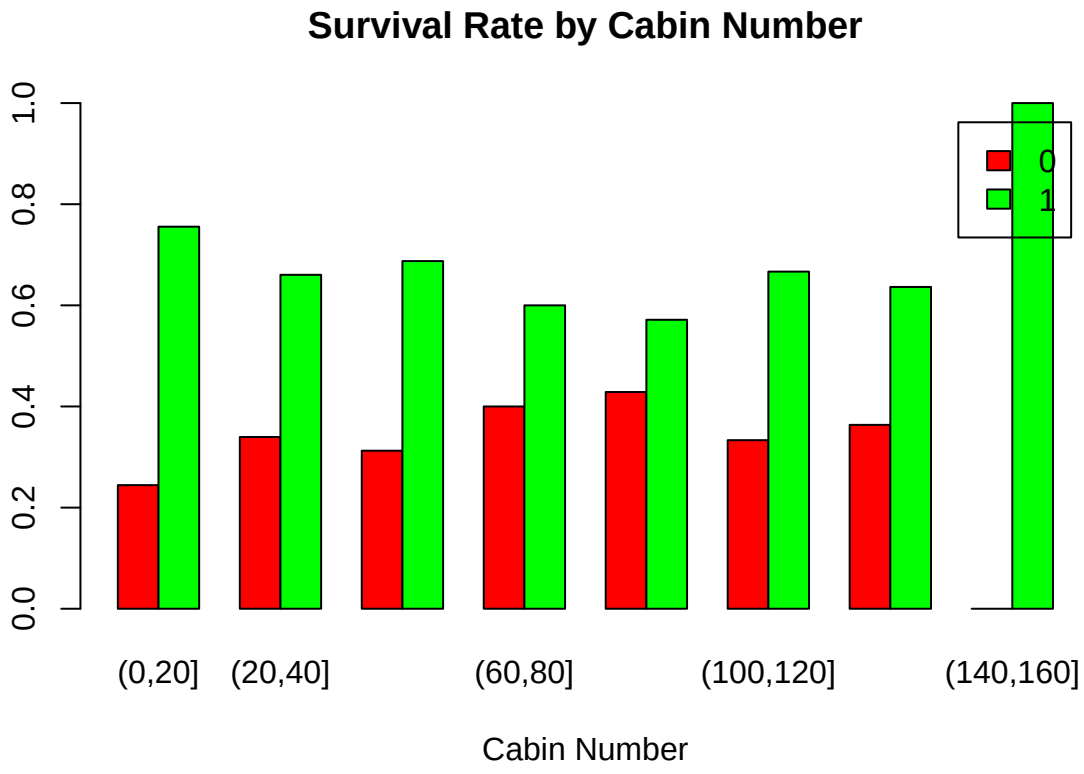
Because only certain cabin letters, B-F, produce a net positive survival rate, it is helpful to have these separate categories as opposed to the previous version of cabin being listed vs. unlisted. In the CabinLetter column, we have replaced all NAs with "Unknown." Before we move on to the next category, we should look for any relationship between cabin number and survival. Perhaps passengers with rooms towards the middle or one particular side of the ship had a lower great of survival than everyone else.

```
# Create CabinNumber column
train$CabinNumber <- as.numeric(gsub( " .*$", "", sub( ".","",train$Cabin)))

# View unique cabin numbers
sort(unique(train$CabinNumber))
```

```
##  [1]   2   3   4   5   6   7   8   9  10  11  12  14  15  16  17  18  19  20  21
## [20]  22  23  24  25  26  28  30  31  32  33  34  35  36  37  38  39  40  41  42
## [39]  44  45  46  47  48  49  50  51  52  54  56  57  58  62  63  65  67  68  69
## [58]  70  71  73  77  78  79  80  82  83  85  86  87  90  91  92  93  94  95  96
## [77]  99 101 102 103 104 106 110 111 118 121 123 124 125 126 128 148
```

```
# View survival plot
counts <- table(train$Survived, cut(train$CabinNumber, breaks=seq(0,160, by=20)))
barplot(prop.table(counts,2), main="Survival Rate by Cabin Number",
```

```
  xlab="Cabin Number", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```

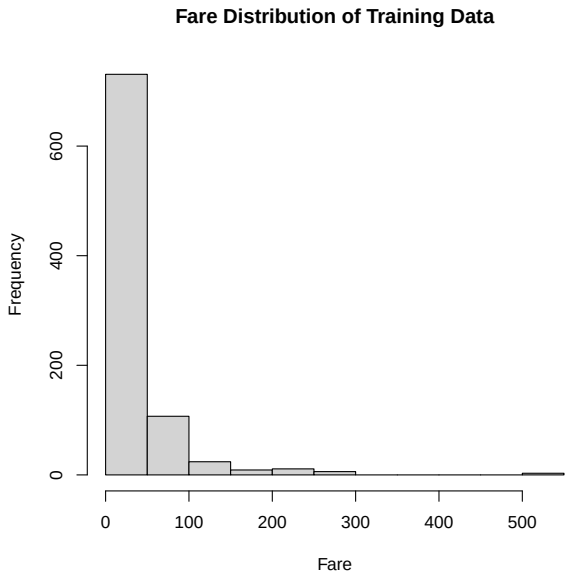


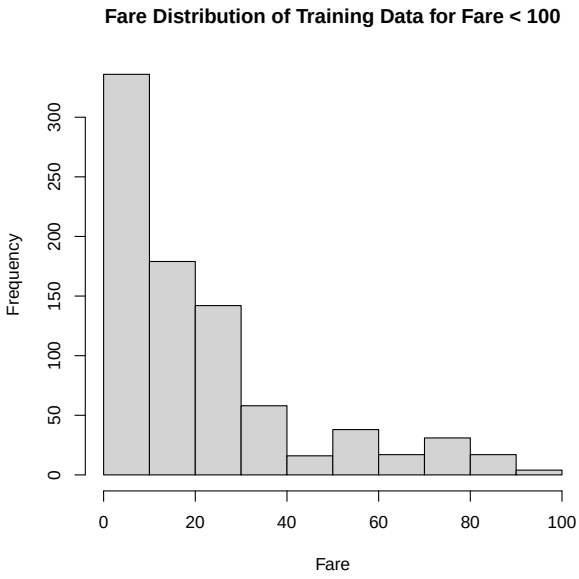
There doesn't seem to be any strong relationship between cabin number and survival rate. We'll remove this variable as well as Cabin.

```
# Remove Cabin and CabinNumber columns
train <- train[,!names(train) %in% c("Cabin","CabinNumber")]
test <- test[,!names(test) %in% c("Cabin")]
```

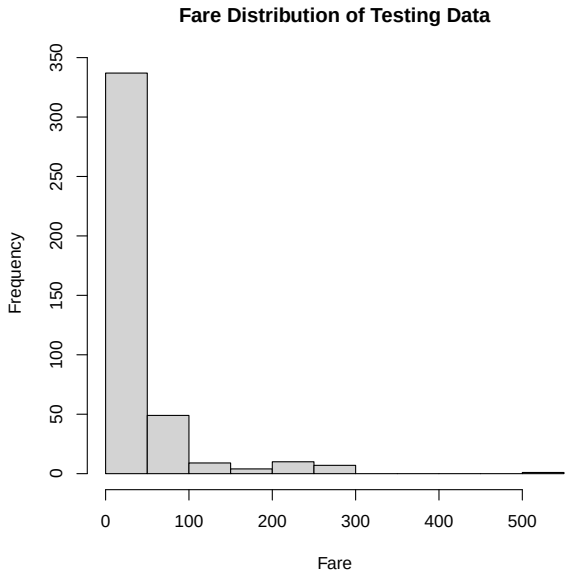
Let's see if there is any correlation between fare and survival rate.

```
# View histogram of fares
my_par = par(mfrow=c(1,2))
hist(train$Fare, main="Fare Distribution of Training Data", xlab="Fare")
hist(train$Fare[train$Fare<100], main="Fare Distribution of Training Data for Fare < 100",
     xlab="Fare")
```





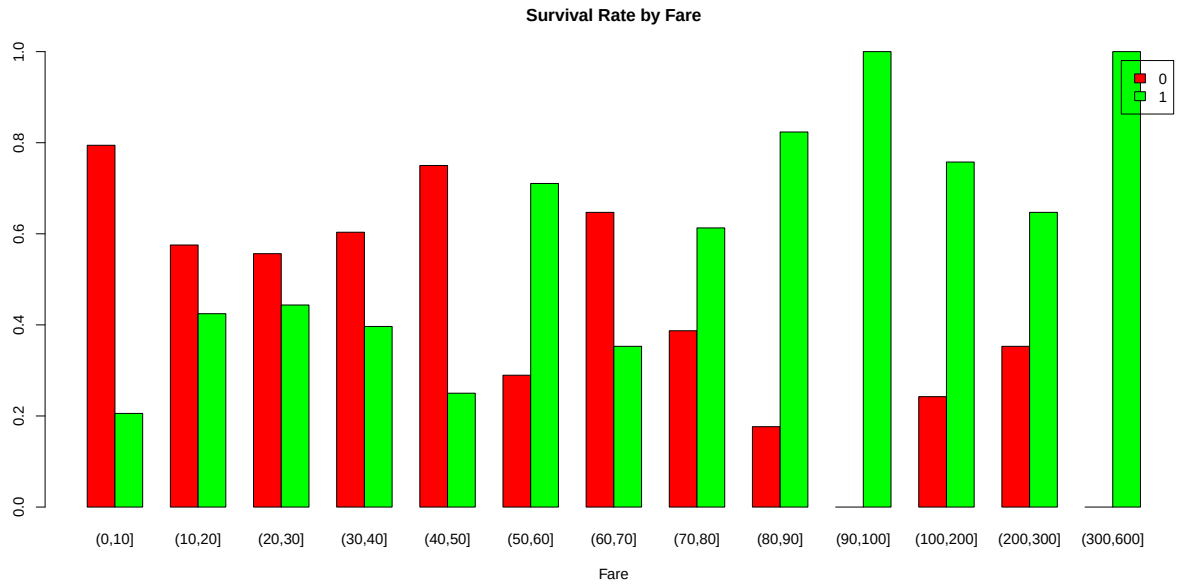

```
 # View histogram of fares for fares < 100
hist(test$Fare, main="Fare Distribution of Testing Data", xlab="Fare")
hist(test$Fare[test$Fare<100], main="Fare Distribution of Testing Data for Fare < 100",
     xlab="Fare")
```




```
par(my_par)
```

We can see the distributions of fares are skewed sharply to the right with the vast majority of fares being less than 20 dollars. The distributions between sets is consistent. There also appears to be a few outleir fares in both datasets that are over 500.

```
# View survival plot
counts <- table(train$Survived, cut(train$Fare, breaks=c(seq(0,100, by=10), 200, 300, 600)))
barplot(prop.table(counts,2), main="Survival Rate by Fare",
  xlab="Fare", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```



As expected, we can see a clear relationship between fare and survival rate. Next we'll see if there is anything we can find from the ticket column.

```
# View unique tickets
sort(unique(train$Ticket))
```

```
##   [1] "110152"            "110413"            "110465"
##   [4] "110564"            "110813"            "111240"
##   [7] "111320"            "111361"            "111369"
##  [10] "111426"            "111427"            "111428"
##  [13] "112050"            "112052"            "112053"
##  [16] "112058"            "112059"            "112277"
##  [19] "112379"            "113028"            "113043"
##  [22] "113050"            "113051"            "113055"
##  [25] "113056"            "113059"            "113501"
##  [28] "113503"            "113505"            "113509"
##  [31] "113510"            "113514"            "113572"
##  [34] "113760"            "113767"            "113773"
##  [37] "113776"            "113781"            "113783"
##  [40] "113784"            "113786"            "113787"
##  [43] "113788"            "113789"            "113792"
##  [46] "113794"            "113796"            "113798"
##  [49] "113800"            "113803"            "113804"
##  [52] "113806"            "113807"            "11668"
##  [55] "11751"             "11752"             "11753"
##  [58] "11755"             "11765"             "11767"
##  [61] "11769"             "11771"             "11774"
```

```
##  [64] "11813"            "11967"            "12233"
##  [67] "12460"            "12749"            "13049"
##  [70] "13213"            "13214"            "13502"
##  [73] "13507"            "13509"            "13567"
##  [76] "13568"            "14311"            "14312"
##  [79] "14313"            "14973"            "1601"
##  [82] "16966"            "16988"            "17421"
##  [85] "17453"            "17463"            "17464"
##  [88] "17465"            "17466"            "17474"
##  [91] "17764"            "19877"            "19928"
##  [94] "19943"            "19947"            "19950"
##  [97] "19952"            "19972"            "19988"
## [100] "19996"            "2003"             "211536"
## [103] "21440"            "218629"           "219533"
## [106] "220367"           "220845"           "2223"
## [109] "223596"           "226593"           "226875"
## [112] "228414"           "229236"           "230080"
## [115] "230136"           "230433"           "230434"
## [118] "231919"           "231945"           "233639"
## [121] "233866"           "234360"           "234604"
## [124] "234686"           "234818"           "236171"
## [127] "236852"           "236853"           "237442"
## [130] "237565"           "237668"           "237671"
## [133] "237736"           "237789"           "237798"
## [136] "239853"           "239854"           "239855"
## [139] "239856"           "239865"           "240929"
## [142] "24160"            "243847"           "243880"
## [145] "244252"           "244270"           "244278"
## [148] "244310"           "244358"           "244361"
## [151] "244367"           "244373"           "248698"
## [154] "248706"           "248723"           "248727"
## [157] "248731"           "248733"           "248738"
## [160] "248740"           "248747"           "250643"
## [163] "250644"           "250646"           "250647"
## [166] "250648"           "250649"           "250651"
## [169] "250652"           "250653"           "250655"
## [172] "2620"             "2623"             "2624"
## [175] "2625"             "2626"             "2627"
## [178] "2628"             "2629"             "2631"
## [181] "26360"            "2641"             "2647"
## [184] "2648"             "2649"             "2650"
## [187] "2651"             "2653"             "2659"
## [190] "2661"             "2662"             "2663"
## [193] "2664"             "2665"             "2666"
## [196] "2667"             "2668"             "2669"
## [199] "26707"            "2671"             "2672"
## [202] "2674"             "2677"             "2678"
## [205] "2680"             "2683"             "2685"
## [208] "2686"             "2687"             "2689"
## [211] "2690"             "2691"             "2693"
## [214] "2694"             "2695"             "2697"
## [217] "2699"             "2700"             "27042"
## [220] "27267"            "27849"            "28134"
## [223] "28206"            "28213"            "28220"
```

```
## [226] "28228"            "28403"            "28424"
## [229] "28425"            "28551"            "28664"
## [232] "28665"            "29011"            "2908"
## [235] "29103"            "29104"            "29105"
## [238] "29106"            "29108"            "2926"
## [241] "29750"            "29751"            "3101264"
## [244] "3101265"          "3101267"          "3101276"
## [247] "3101277"          "3101278"          "3101281"
## [250] "3101295"          "3101296"          "3101298"
## [253] "31027"            "31028"            "312991"
## [256] "312992"           "312993"           "31418"
## [259] "315037"           "315082"           "315084"
## [262] "315086"           "315088"           "315089"
## [265] "315090"           "315093"           "315094"
## [268] "315096"           "315097"           "315098"
## [271] "315151"           "315153"           "323592"
## [274] "323951"           "324669"           "330877"
## [277] "330909"           "330919"           "330923"
## [280] "330931"           "330932"           "330935"
## [283] "330958"           "330959"           "330979"
## [286] "330980"           "334912"           "335097"
## [289] "335677"           "33638"            "336439"
## [292] "3411"             "341826"           "34218"
## [295] "342826"           "343095"           "343120"
## [298] "343275"           "343276"           "345364"
## [301] "345572"           "345763"           "345764"
## [304] "345765"           "345767"           "345769"
## [307] "345770"           "345773"           "345774"
## [310] "345777"           "345778"           "345779"
## [313] "345780"           "345781"           "345783"
## [316] "3460"             "347054"           "347060"
## [319] "347061"           "347062"           "347063"
## [322] "347064"           "347067"           "347068"
## [325] "347069"           "347071"           "347073"
## [328] "347074"           "347076"           "347077"
## [331] "347078"           "347080"           "347081"
## [334] "347082"           "347083"           "347085"
## [337] "347087"           "347088"           "347089"
## [340] "3474"             "347464"           "347466"
## [343] "347468"           "347470"           "347742"
## [346] "347743"           "348121"           "348123"
## [349] "348124"           "349201"           "349203"
## [352] "349204"           "349205"           "349206"
## [355] "349207"           "349208"           "349209"
## [358] "349210"           "349212"           "349213"
## [361] "349214"           "349215"           "349216"
## [364] "349217"           "349218"           "349219"
## [367] "349221"           "349222"           "349223"
## [370] "349224"           "349225"           "349227"
## [373] "349228"           "349231"           "349233"
## [376] "349234"           "349236"           "349237"
## [379] "349239"           "349240"           "349241"
## [382] "349242"           "349243"           "349244"
## [385] "349245"           "349246"           "349247"
```

```
## [388] "349248"           "349249"           "349251"
## [391] "349252"           "349253"           "349254"
## [394] "349256"           "349257"           "349909"
## [397] "349910"           "349912"           "350025"
## [400] "350026"           "350029"           "350034"
## [403] "350035"           "350036"           "350042"
## [406] "350043"           "350046"           "350047"
## [409] "350048"           "350050"           "350052"
## [412] "350060"           "350404"           "350406"
## [415] "350407"           "350417"           "35273"
## [418] "35281"            "35851"            "35852"
## [421] "358585"           "36209"            "362316"
## [424] "363291"           "363294"           "363592"
## [427] "364498"           "364499"           "364500"
## [430] "364506"           "364511"           "364512"
## [433] "364516"           "364846"           "364848"
## [436] "364849"           "364850"           "364851"
## [439] "365222"           "365226"           "36568"
## [442] "367226"           "367228"           "367229"
## [445] "367230"           "367231"           "367232"
## [448] "367655"           "368323"           "36864"
## [451] "36865"            "36866"            "368703"
## [454] "36928"            "36947"            "36963"
## [457] "36967"            "36973"            "370129"
## [460] "370365"           "370369"           "370370"
## [463] "370371"           "370372"           "370373"
## [466] "370375"           "370376"           "370377"
## [469] "371060"           "371110"           "371362"
## [472] "372622"           "373450"           "374746"
## [475] "374887"           "374910"           "376564"
## [478] "376566"           "382649"           "382651"
## [481] "382652"           "383121"           "384461"
## [484] "386525"           "392091"           "392092"
## [487] "392096"           "394140"           "4133"
## [490] "4134"             "4135"             "4136"
## [493] "4137"             "4138"             "4579"
## [496] "54636"            "5727"             "65303"
## [499] "65304"            "65306"            "6563"
## [502] "693"              "695"              "7267"
## [505] "7534"             "7540"             "7545"
## [508] "7546"             "7552"             "7553"
## [511] "7598"             "8471"             "8475"
## [514] "9234"             "A./5. 2152"       "A./5. 3235"
## [517] "A.5. 11206"       "A.5. 18509"       "A/4 45380"
## [520] "A/4 48871"        "A/4. 20589"       "A/4. 34244"
## [523] "A/4. 39886"       "A/5 21171"        "A/5 21172"
## [526] "A/5 21173"        "A/5 21174"        "A/5 2466"
## [529] "A/5 2817"         "A/5 3536"         "A/5 3540"
## [532] "A/5 3594"         "A/5 3902"         "A/5. 10482"
## [535] "A/5. 13032"       "A/5. 2151"        "A/5. 3336"
## [538] "A/5. 3337"        "A/5. 851"         "A/S 2816"
## [541] "A4. 54510"        "C 17369"          "C 4001"
## [544] "C 7075"           "C 7076"           "C 7077"
## [547] "C.A. 17248"       "C.A. 18723"       "C.A. 2315"
```

```
## [550] "C.A. 24579"         "C.A. 24580"         "C.A. 2673"
## [553] "C.A. 29178"         "C.A. 29395"         "C.A. 29566"
## [556] "C.A. 31026"         "C.A. 31921"         "C.A. 33111"
## [559] "C.A. 33112"         "C.A. 33595"         "C.A. 34260"
## [562] "C.A. 34651"         "C.A. 37671"         "C.A. 5547"
## [565] "C.A. 6212"          "C.A./SOTON 34068"   "CA 2144"
## [568] "CA. 2314"           "CA. 2343"           "F.C. 12750"
## [571] "F.C.C. 13528"       "F.C.C. 13529"       "F.C.C. 13531"
## [574] "Fa 265302"          "LINE"               "P/PP 3381"
## [577] "PC 17318"           "PC 17473"           "PC 17474"
## [580] "PC 17475"           "PC 17476"           "PC 17477"
## [583] "PC 17482"           "PC 17483"           "PC 17485"
## [586] "PC 17558"           "PC 17569"           "PC 17572"
## [589] "PC 17582"           "PC 17585"           "PC 17590"
## [592] "PC 17592"           "PC 17593"           "PC 17595"
## [595] "PC 17596"           "PC 17597"           "PC 17599"
## [598] "PC 17600"           "PC 17601"           "PC 17603"
## [601] "PC 17604"           "PC 17605"           "PC 17608"
## [604] "PC 17609"           "PC 17610"           "PC 17611"
## [607] "PC 17612"           "PC 17754"           "PC 17755"
## [610] "PC 17756"           "PC 17757"           "PC 17758"
## [613] "PC 17759"           "PC 17760"           "PC 17761"
## [616] "PP 4348"            "PP 9549"            "S.C./A.4. 23567"
## [619] "S.C./PARIS 2079"    "S.O./P.P. 3"        "S.O./P.P. 751"
## [622] "S.O.C. 14879"       "S.O.P. 1166"        "S.P. 3464"
## [625] "S.W./PP 752"        "SC 1748"            "SC/AH 29037"
## [628] "SC/AH 3085"         "SC/AH Basle 541"    "SC/Paris 2123"
## [631] "SC/PARIS 2131"      "SC/PARIS 2133"      "SC/PARIS 2146"
## [634] "SC/PARIS 2149"      "SC/Paris 2163"      "SC/PARIS 2167"
## [637] "SCO/W 1585"         "SO/C 14885"         "SOTON/O.Q. 3101305"
## [640] "SOTON/O.Q. 3101306" "SOTON/O.Q. 3101307" "SOTON/O.Q. 3101310"
## [643] "SOTON/O.Q. 3101311" "SOTON/O.Q. 3101312" "SOTON/O.Q. 392078"
## [646] "SOTON/O.Q. 392087"  "SOTON/O2 3101272"   "SOTON/O2 3101287"
## [649] "SOTON/OQ 3101316"   "SOTON/OQ 3101317"   "SOTON/OQ 392076"
## [652] "SOTON/OQ 392082"    "SOTON/OQ 392086"    "SOTON/OQ 392089"
## [655] "SOTON/OQ 392090"    "STON/O 2. 3101269"  "STON/O 2. 3101273"
## [658] "STON/O 2. 3101274"  "STON/O 2. 3101275"  "STON/O 2. 3101280"
## [661] "STON/O 2. 3101285"  "STON/O 2. 3101286"  "STON/O 2. 3101288"
## [664] "STON/O 2. 3101289"  "STON/O 2. 3101292"  "STON/O 2. 3101293"
## [667] "STON/O 2. 3101294"  "STON/O2. 3101271"   "STON/O2. 3101279"
## [670] "STON/O2. 3101282"   "STON/O2. 3101283"   "STON/O2. 3101290"
## [673] "SW/PP 751"          "W./C. 14258"        "W./C. 14263"
## [676] "W./C. 6607"         "W./C. 6608"         "W./C. 6609"
## [679] "W.E.P. 5734"        "W/C 14208"          "WE/P 5735"
```

```
sort(unique(test$Ticket))
```

```
##   [1] "110469"           "110489"           "110813"
##   [4] "111163"           "112051"           "112058"
##   [7] "112377"           "112378"           "112901"
##  [10] "113038"           "113044"           "113054"
##  [13] "113059"           "113503"           "113509"
##  [16] "113773"           "113778"           "113780"
##  [19] "113781"           "113790"           "113791"
```

```
##  [22] "113795"            "113796"            "113801"
##  [25] "11753"             "11765"             "11767"
##  [28] "11769"             "11770"             "11778"
##  [31] "11813"             "1222"              "12749"
##  [34] "13050"             "13236"             "13508"
##  [37] "13567"             "13695"             "13905"
##  [40] "1601"              "16966"             "17463"
##  [43] "17464"             "17475"             "17765"
##  [46] "17770"             "19877"             "19924"
##  [49] "19928"             "19950"             "2003"
##  [52] "211535"            "21228"             "21332"
##  [55] "220844"            "220845"            "226875"
##  [58] "228414"            "230136"            "233478"
##  [61] "233734"            "235509"            "236853"
##  [64] "236854"            "237216"            "237249"
##  [67] "237393"            "237670"            "237734"
##  [70] "237735"            "237789"            "239059"
##  [73] "240261"            "240276"            "24065"
##  [76] "24160"             "242963"            "244346"
##  [79] "244358"            "244360"            "244368"
##  [82] "248659"            "248726"            "248734"
##  [85] "248738"            "248744"            "248746"
##  [88] "250650"            "250651"            "2543"
##  [91] "2621"              "2622"              "2625"
##  [94] "26360"             "2650"              "2652"
##  [97] "2653"              "2654"              "2655"
## [100] "2656"              "2657"              "2658"
## [103] "2660"              "2661"              "2662"
## [106] "2668"              "2670"              "26707"
## [109] "2673"              "2675"              "2676"
## [112] "2678"              "2679"              "2680"
## [115] "2681"              "2682"              "2684"
## [118] "2688"              "2689"              "2692"
## [121] "2696"              "2698"              "28004"
## [124] "28034"             "28133"             "28220"
## [127] "28221"             "28404"             "28664"
## [130] "28666"             "29103"             "29105"
## [133] "29107"             "2926"              "29750"
## [136] "3101266"           "3101295"           "3101297"
## [139] "3101298"           "315083"            "315085"
## [142] "315087"            "315091"            "315092"
## [145] "315095"            "315152"            "315153"
## [148] "315154"            "32302"             "329944"
## [151] "330844"            "330910"            "330911"
## [154] "330920"            "330924"            "330963"
## [157] "330968"            "330971"            "330972"
## [160] "334914"            "334915"            "335432"
## [163] "33638"             "3410"              "342441"
## [166] "342684"            "342712"            "343271"
## [169] "345498"            "345501"            "345572"
## [172] "345763"            "345768"            "345771"
## [175] "345775"            "3470"              "347065"
## [178] "347066"            "347070"            "347072"
## [181] "347075"            "347077"            "347079"
```

```
## [184] "347080"           "347086"           "347090"
## [187] "347091"           "347465"           "347467"
## [190] "347469"           "347471"           "348122"
## [193] "348125"           "349202"           "349211"
## [196] "349220"           "349226"           "349229"
## [199] "349230"           "349232"           "349235"
## [202] "349238"           "349250"           "349255"
## [205] "349256"           "349909"           "349910"
## [208] "349911"           "350026"           "350033"
## [211] "350045"           "350053"           "350054"
## [214] "350403"           "350405"           "350408"
## [217] "350409"           "350410"           "350416"
## [220] "359306"           "359309"           "363272"
## [223] "363611"           "364498"           "364856"
## [226] "364858"           "364859"           "365235"
## [229] "365237"           "36568"            "366713"
## [232] "367226"           "367227"           "368364"
## [235] "368402"           "368573"           "368702"
## [238] "368783"           "36928"            "3701"
## [241] "370129"           "370368"           "370371"
## [244] "370374"           "371109"           "371362"
## [247] "376563"           "376566"           "382650"
## [250] "382652"           "382653"           "383123"
## [253] "383162"           "386525"           "392091"
## [256] "392095"           "4133"             "65305"
## [259] "680"              "694"              "7266"
## [262] "7538"             "7548"             "7935"
## [265] "9232"             "A. 2. 39186"      "A./5. 3338"
## [268] "A.5. 3236"        "A/4 31416"        "A/4 48871"
## [271] "A/4 48873"        "A/5 1478"         "A/5 21175"
## [274] "A/5. 3337"        "A/5. 851"         "AQ/3. 30631"
## [277] "AQ/4 3130"        "C 17368"          "C 4001"
## [280] "C.A. 15185"       "C.A. 2315"        "C.A. 2673"
## [283] "C.A. 30769"       "C.A. 31029"       "C.A. 31030"
## [286] "C.A. 33112"       "C.A. 33595"       "C.A. 34050"
## [289] "C.A. 34644"       "C.A. 34651"       "C.A. 37671"
## [292] "C.A. 42795"       "C.A. 49867"       "C.A. 6212"
## [295] "CA 2144"          "CA 31352"         "CA. 2343"
## [298] "F.C. 12750"       "F.C. 12998"       "F.C.C. 13528"
## [301] "F.C.C. 13534"     "F.C.C. 13540"     "LP 1588"
## [304] "PC 17483"         "PC 17531"         "PC 17558"
## [307] "PC 17562"         "PC 17569"         "PC 17580"
## [310] "PC 17585"         "PC 17591"         "PC 17592"
## [313] "PC 17594"         "PC 17597"         "PC 17598"
## [316] "PC 17599"         "PC 17603"         "PC 17606"
## [319] "PC 17607"         "PC 17608"         "PC 17613"
## [322] "PC 17755"         "PC 17756"         "PC 17757"
## [325] "PC 17758"         "PC 17759"         "PC 17760"
## [328] "PC 17761"         "PP 9549"          "S.C./PARIS 2079"
## [331] "S.O./P.P. 2"      "S.O./P.P. 251"    "S.O./P.P. 752"
## [334] "S.O.C. 14879"     "SC 14888"         "SC/A.3 2861"
## [337] "SC/A4 23568"      "SC/AH 29037"      "SC/AH 3085"
## [340] "SC/Paris 2123"    "SC/PARIS 2147"    "SC/PARIS 2148"
## [343] "SC/PARIS 2159"    "SC/PARIS 2166"    "SC/PARIS 2167"
```

```
## [346] "SC/PARIS 2168"      "SOTON/O.Q. 3101262" "SOTON/O.Q. 3101263"
## [349] "SOTON/O.Q. 3101308" "SOTON/O.Q. 3101309" "SOTON/O.Q. 3101314"
## [352] "SOTON/O.Q. 3101315" "SOTON/O2 3101284"   "SOTON/OQ 392083"
## [355] "STON/O 2. 3101268"  "STON/O 2. 3101291"  "STON/O2. 3101270"
## [358] "STON/OQ. 369943"    "W./C. 14260"        "W./C. 14266"
## [361] "W./C. 6607"         "W./C. 6608"         "W.E.P. 5734"
```

We have so many tickets to choose from, it's hard to know how to use this data. My first inclination is to do what we did with Cabin and see if the starting letter has any affect on the survival rate.

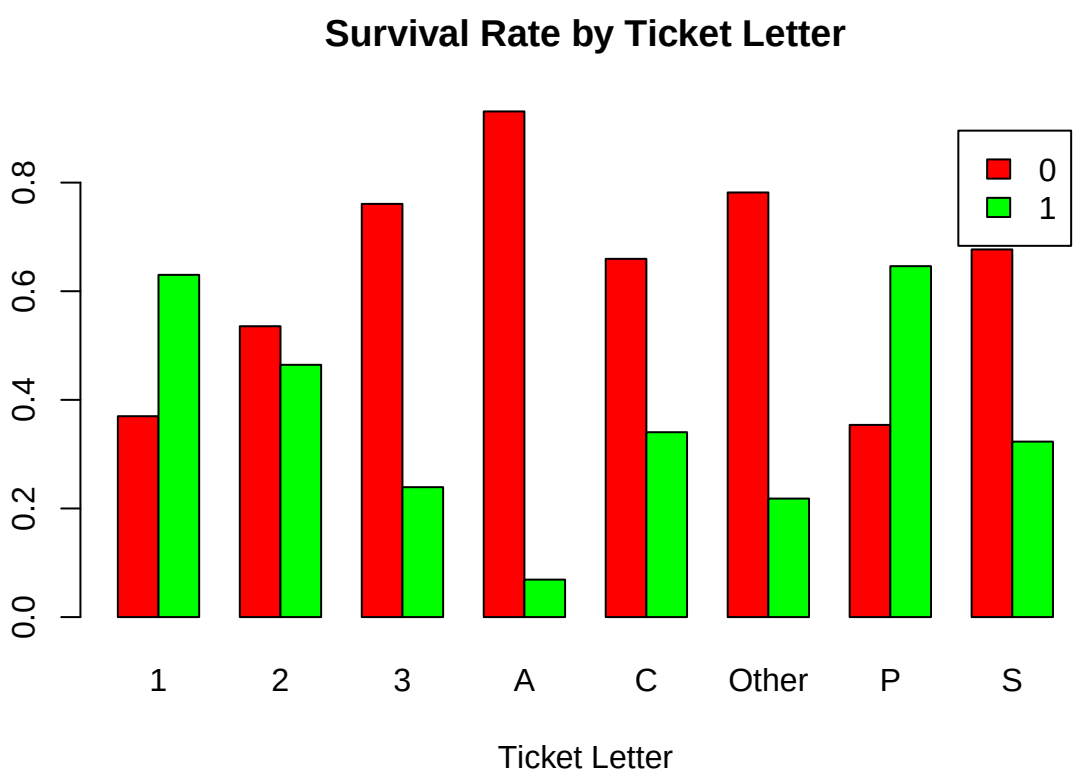
```
# View tables of TicketLetter
table(train$Survived, sapply(train$Ticket, substr,1,1))
```

```
##
##       1   2   3   4   5   6   7   8   9   A   C   F   L   P   S   W
##   0  54  98 229   8   3   5   8   2   0  27  31   3   3  23  44  11
##   1  92  85  72   2   0   1   1   0   1   2  16   4   1  42  21   2
```

```
table(sapply(test$Ticket, substr,1,1))
```

```
##
##   1   2   3   4   6   7   9   A   C   F   L   P   S   W
##  64  95 128   1   3   4   1  13  30   6   1  33  33   6
```

```
# Only use categories such than n > 25 for training
ticket.cats <- c("1","2","3","A","C","P","S")
train$TicketLetter <- ifelse(sapply(train$Ticket, substr,1,1) %in% ticket.cats,
                             sapply(train$Ticket, substr,1,1), "Other")
test$TicketLetter <- ifelse(sapply(test$Ticket, substr,1,1) %in% ticket.cats,
                            sapply(test$Ticket, substr,1,1), "Other")

# View Survival Plot
counts <- table(train$Survived, train$TicketLetter)
barplot(prop.table(counts,2), main="Survival Rate by Ticket Letter",
  xlab="Ticket Letter", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```




```
# Remove Ticket column
train <- train[,!names(train) %in% c("Ticket")]
test <- test[,!names(test) %in% c("Ticket")]
```

The tickets whose starting letter frequency is less than 25 in the training set were placed together in an "other" group. We then plotted survival rate by ticket letter and found that some letter indicated survival more frequently than others. We must be careful with this category as it could be too strongly correlated with the Pclass or Cabin variables, as they could all be affected by the underlying factor of passenger wealth. We will address this later once we've finished analyzing the remaining predictor, name. While names themselves probably don't tell us too much, the title of the passenger may give us information about their survival rate or perhaps their age.

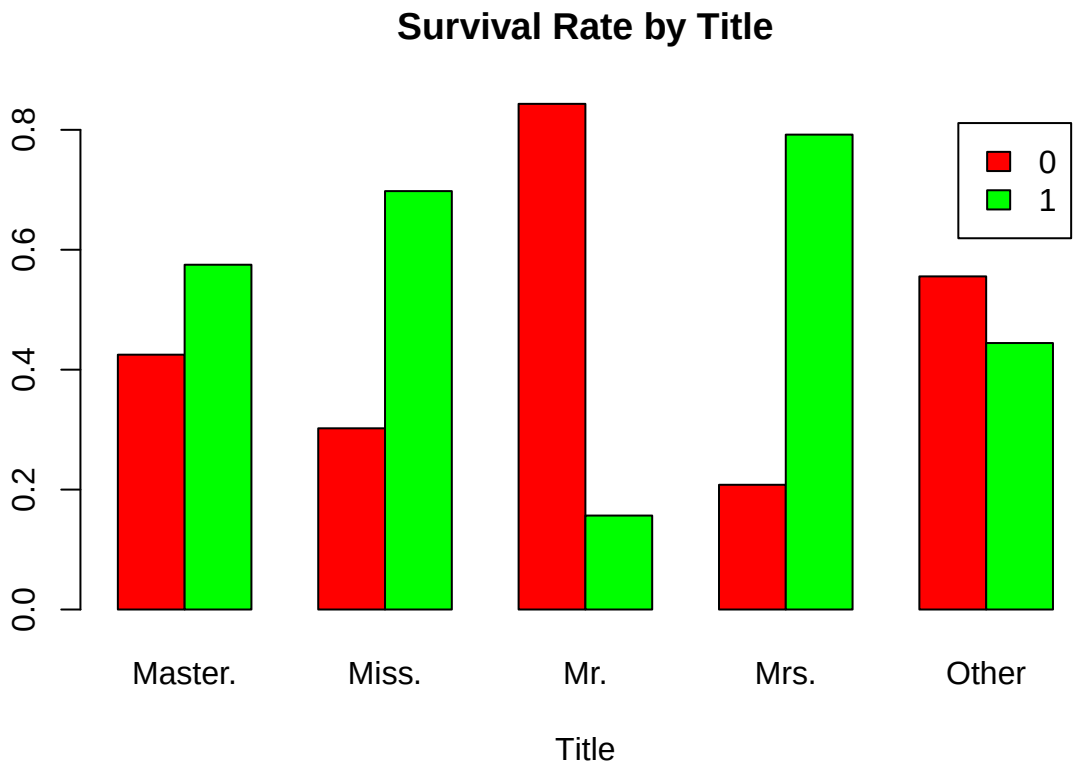
```
# Get Titles from the Name Column
train.title <- sapply(strsplit(sapply(strsplit(train$Name, ","),"[[",2)," "),"[[",2)
test.title <- sapply(strsplit(sapply(strsplit(test$Name, ","),"[[",2)," "),"[[",2)

# View Counts of each title
table(train.title)
```

```
## train.title
##     Capt.      Col.      Don.       Dr. Jonkheer.     Lady.    Major.   Master.
##         1         2         1         7         1         1         2        40
##     Miss.     Mlle.      Mme.       Mr.      Mrs.       Ms.      Rev.      Sir.
##       182         2         1       517       125         1         6         1
##       the
##         1
```

```
table(test.title)
```

```
## test.title
##    Col.   Dona.     Dr. Master.   Miss.     Mr.    Mrs.     Ms.    Rev.
##       2       1       1      21      78     240      72       1       2
```

```
# Only use the titles with n > 25 in training set
most_common_titles <- c("Master.", "Miss.", "Mr.", "Mrs.")
train.title <- ifelse(train.title %in% most_common_titles, train.title, "Other")
test.title <- ifelse(test.title %in% most_common_titles, test.title, "Other")

# Survival Plot
counts <- table(train$Survived, train.title)
barplot(prop.table(counts,2), main="Survival Rate by Title",
  xlab="Title", col=c("red","green"),
  legend = rownames(counts), beside=TRUE)
```




```
# Age by Title Plot
boxplot(train$Age~train.title, main="Age Distribution by Title (Training Set)",
  xlab="Title", ylab="Age")
```


```
boxplot(test$Age~test.title, main="Age Distribution by Title (Testing Set)",
  xlab="Title", ylab="Age")
```


```
# Add Title to Dataframes and Remove Name
train$Title <- train.title
test$Title <- test.title
train <- train[,-which(names(train)=="Name")]
test <- test[,-which(names(test)=="Name")]
```

The relationship between title and age will likely help us fill in NA values for age. The plot of age by title show that the title of "Master." is given to young men under the age of 16 and "Miss." is given to young women until they marry. The distribution of ages for "Mr." "Mrs." and other titles are much higher. The relationship between survival and title seems to be the same relationship between survival and age. It's likely that using the title column will be too highly correlated to age and won't actually help our model. For this reason, my plan is to use title to help impute the missing values of age. Before we get to imputing missing values, it's important the we first check for collinearity between variables.

```
# Assign predictors to vectors
continuous_predictors <- c("Pclass", "Age", "Fare")
binary_categorical_predictors <- "Sex"
non_binary_categorical_predictors <- c("Embarked", "FamSize", "CabinLetter",
                                       "TicketLetter", "Title")
categorical_predictors <- c(binary_categorical_predictors,
                            non_binary_categorical_predictors)

# Factorize categorical data
train[sapply(train, is.character)] <- lapply(train[sapply(train, is.character)],
                                             as.factor)
test[sapply(test, is.character)] <- lapply(test[sapply(test, is.character)],
```

```
                                          as.factor)

# Check for colinearity between the continuous predictor variables.
vif(train[,which(names(train) %in% continuous_predictors)])
```

```
##   Pclass      Age     Fare
## 1.688894 1.181104 1.472237
```

```
# Check for correlation of categorical predictor terms
for (i in 1:(length(categorical_predictors)-1)){
    for (j in (i+1):length(categorical_predictors)){
        cat(paste0("Categorical Variables: ", categorical_predictors[i], " and ",
                   categorical_predictors[j]), "\n")
        tabl <- table(train[,categorical_predictors[i]],
                      train[,categorical_predictors[j]])
        chsq <- suppressWarnings(chisq.test(table(tabl)))
        cat(paste("P-value:", chsq$p.value), "\n")
        cat(paste("Cramer's V:",
                  sqrt(chsq$statistic / (sum(tabl)*(max(dim(tabl))-1)))), "\n\n")
    }
}
```

```
## Categorical Variables: Sex and Embarked
## P-value: 1
## Cramer's V: 0
##
## Categorical Variables: Sex and FamSize
## P-value: 1
## Cramer's V: 0
##
## Categorical Variables: Sex and CabinLetter
## P-value: 0.99995682816108
## Cramer's V: 0.0155080701733078
##
## Categorical Variables: Sex and TicketLetter
## P-value: 0.999999584396819
## Cramer's V: 0.0118444842419784
##
## Categorical Variables: Sex and Title
## P-value: 0.493624491073462
## Cramer's V: 0.0389249472080761
##
## Categorical Variables: Embarked and FamSize
## P-value: 1
## Cramer's V: 0
##
## Categorical Variables: Embarked and CabinLetter
## P-value: 0.382051661502864
## Cramer's V: 0.0507060862523494
##
## Categorical Variables: Embarked and TicketLetter
## P-value: 0.380224560186033
## Cramer's V: 0.0523945852661805
```

```
##
## Categorical Variables: Embarked and Title
## P-value: 0.999998397950152
## Cramer's V: 0.0156115193065287
##
## Categorical Variables: FamSize and CabinLetter
## P-value: 0.679029057090415
## Cramer's V: 0.0438634463306513
##
## Categorical Variables: FamSize and TicketLetter
## P-value: 0.999999994197422
## Cramer's V: 0.016346941287125
##
## Categorical Variables: FamSize and Title
## P-value: 0.999884295643391
## Cramer's V: 0.0202860206483395
##
## Categorical Variables: CabinLetter and TicketLetter
## P-value: 8.2631671636194e-43
## Cramer's V: 0.203558805433517
##
## Categorical Variables: CabinLetter and Title
## P-value: 0.0562232094410743
## Cramer's V: 0.0674794133061179
##
## Categorical Variables: TicketLetter and Title
## P-value: 0.696776146303107
## Cramer's V: 0.0566274657152917
```

Of the continuous variables, the VIF values show that while there is a correlation between our continuous predictors, their correlation is relatively low. Similarly, most of our categorical variables are not strongly correlated. The only exception to this is CabinLetter and TicketLetter which have a Cramer's V of 0.2. That said this is somewhat expected as the Cabin Letter and Ticket Letter were likely related. However, because we were missing so many Cabins entries, I think it is okay to include both. Now we're finally ready to start imputing missing values. Let's start by listing all the missing values we have.

```
for (col in names(train)){
    if (sum(is.na(train[,col])) > 0){
        cat(paste("There's", sum(is.na(train[,col])), "missing values in the", col,
                  "column of the training dataset."), "\n")
    }
}
```

```
## There's 177 missing values in the Age column of the training dataset.
## There's 2 missing values in the Embarked column of the training dataset.
```

```
for (col in names(test)){
    if (sum(is.na(test[,col])) > 0){
        cat(paste("There's", sum(is.na(test[,col])), "missing values in the", col,
                  "column of the testing dataset."), "\n")
    }
}
```

```
## There's 86 missing values in the Age column of the testing dataset.
## There's 1 missing values in the Fare column of the testing dataset.
```

For the Embarked and Fare missing values, we are going to use simple imputation methods as we are only dealing with 3 observations.

```
# View observations where embarked is missing
train[is.na(train$Embarked),]
```

```
##     Survived Pclass    Sex Age Fare Embarked FamSize CabinLetter TicketLetter
## 62         1      1 female  38   80     <NA>    Solo           B            1
## 830        1      1 female  62   80     <NA>    Solo           B            1
##     Title
## 62  Miss.
## 830  Mrs.
```

```
# View table embarked with and without accounting for fare
table(train$Embarked)
```

```
##
##   C   Q   S
## 168  77 644
```

```
table(train[train$Fare>70 & train$TicketLetter=="1","Embarked"])
```

```
##
##  C  Q  S
## 16  2 27
```

```
# Replace missing embarked with "S"
train[is.na(train$Embarked),"Embarked"] <- "S"
```

```
# View observations where fare is missing
test[is.na(test$Fare),]
```

```
##     Pclass  Sex  Age Fare Embarked FamSize CabinLetter TicketLetter Title
## 153      3 male 60.5   NA        S    Solo     Unknown            3   Mr.
```

```
# Replace missing fare with median fare for all Embarked=S and TicketLetter=3.
(fare.med <- median(test[test$Embarked == "S" & test$TicketLetter=="3","Fare"],
                    na.rm=TRUE))
```

```
## [1] 8.05
```

```
test[is.na(test$Fare),"Fare"] <- fare.med
```

For age, imputation is going to be a lot more challenging. Approximately 20% of the values for age are missing from the training and testing dataset. To simply put the mean or median age value would surely hurt our model. Instead, we are going to predict the age of our missing values using a neural network and MICE with different tuning parameters. We will use cross validation to determine which imputation model is best. Lets start by spliting our datasets into training (where age is known) and NA (where age isn't known) datasets. We will use two new NA sets to allow us to predict the missing age values for the original training and testing datasets.

```
# New training data. Combination of all entries (training and testing) where age is known. Survived col
age.train <- rbind(train[!is.na(train$Age),-which(names(train)=="Survived")],
                   test[!is.na(test$Age),])

# Dataframes of original traning and testing in which age is unknown. Survived and Age col removed.
age.NA.train <- train[is.na(train$Age),-which(names(train) %in% c("Survived", "Age"))]
age.NA.test <- test[is.na(test$Age),-which(names(train)=="Age")]

# View heads
head(age.train)
```

```
##   Pclass    Sex Age    Fare Embarked FamSize CabinLetter TicketLetter Title
## 1      3   male  22  7.2500        S Nuclear     Unknown            A   Mr.
## 2      1 female  38 71.2833        C Nuclear           C            P  Mrs.
## 3      3 female  26  7.9250        S    Solo     Unknown            S Miss.
## 4      1 female  35 53.1000        S Nuclear           C            1  Mrs.
## 5      3   male  35  8.0500        S    Solo     Unknown            3   Mr.
## 7      1   male  54 51.8625        S    Solo           E            1   Mr.
```

```
head(age.NA.train)
```

```
##    Pclass    Sex    Fare Embarked FamSize CabinLetter TicketLetter Title
## 6       3   male  8.4583        Q    Solo     Unknown            3   Mr.
## 18      2   male 13.0000        S    Solo     Unknown            2   Mr.
## 20      3 female  7.2250        C    Solo     Unknown            2  Mrs.
## 27      3   male  7.2250        C    Solo     Unknown            2   Mr.
## 29      3 female  7.8792        Q    Solo     Unknown            3 Miss.
## 30      3   male  7.8958        S    Solo     Unknown            3   Mr.
```

```
head(age.NA.test)
```

```
##    Pclass    Sex Age Embarked FamSize CabinLetter TicketLetter Title
## 11      3   male  NA        S    Solo     Unknown            3   Mr.
## 23      1 female  NA        S    Solo     Unknown            P  Mrs.
## 30      3   male  NA        C Nuclear     Unknown            2   Mr.
## 34      3 female  NA        S Nuclear     Unknown        Other  Mrs.
## 37      3 female  NA        S    Solo     Unknown            3 Miss.
## 40      3   male  NA        S    Solo     Unknown            1   Mr.
```

```
# View dimensions of training set
dim(age.train)
```

```
## [1] 1046    9
```

```
dim(age.NA.train)
```

```
## [1] 177   8
```

```
dim(age.NA.test)
```

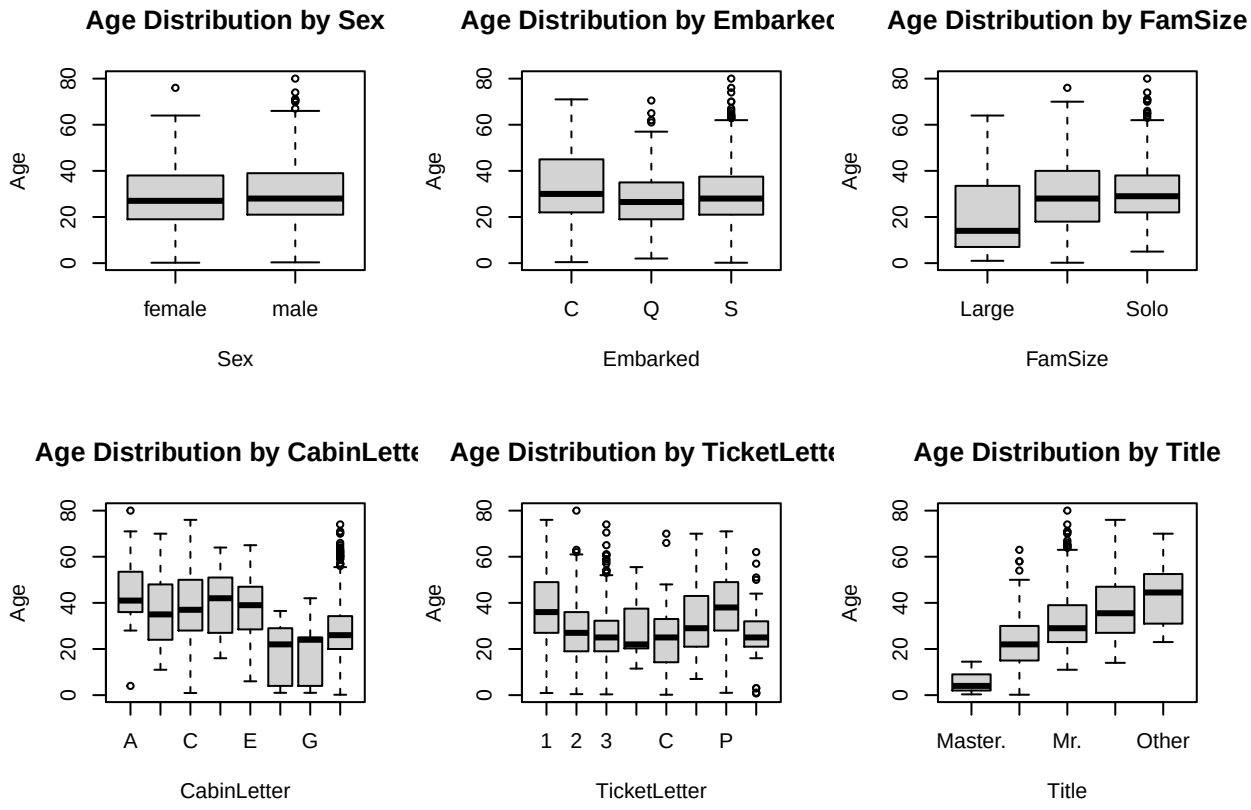
```
## [1] 86  8
```

Next we're going to view which variables are most closely associated with Age and therefore be a good predictors for the missing age values

```
# Correlation of continuous predictors
cor(age.train[!is.na(age.train$Age),which(names(age.train) %in% continuous_predictors)])
```

```
##             Pclass        Age       Fare
## Pclass  1.0000000 -0.4081062 -0.5654071
## Age    -0.4081062  1.0000000  0.1772857
## Fare   -0.5654071  0.1772857  1.0000000
```

```
# Boxplots of Age vs. Categorical Variables
my_par = par(mfrow=c(2,3))
for (col in c(categorical_predictors)){
    boxplot(age.train$Age~age.train[,which(names(age.train)==col)],
            main=paste("Age Distribution by",col),
            xlab=col, ylab="Age")
}
```



Of our predictor variables, Age seems most strongly correlated with Title, Pclass, and CabinLetter. The other variables do not appear to provide any insight and therefore may not be helpful in predicting the age. As a result we will leave them out of our imputation models.

```
# Subset Data
imputation_vars <- c("Age", "Title", "Pclass")
age.train <- age.train[,which(names(age.train) %in% imputation_vars)]
age.NA.train <- age.NA.train[,which(names(age.NA.train) %in% imputation_vars)]
age.NA.test <- age.NA.test[,which(names(age.NA.test) %in% imputation_vars)]


# This for loop takes all the categorical predictors and the larges category columns and puts them in a
# The vector extra_cols will be used to remove excess columns created by dummy_cols
extra_cols <- c("Title")
for (col in extra_cols){
    tbl <- table(age.train[,col])
    largest_cat <- names(tbl)[which(tbl == max(tbl))]
    extra_col <- paste0(col,"_",largest_cat)
    extra_cols <- c(extra_cols, extra_col)
}

# Store age subsets for later
age.train.no.dummies <- age.train

# Create dummy cols for age.train for neural network
age.train <- dummy_cols(age.train)
```

```
## Registered S3 methods overwritten by 'tibble':
##   method     from
##   format.tbl pillar
##   print.tbl  pillar
```

```
age.train <- age.train[,-which(names(age.train) %in% extra_cols)]
age.NA.train <- dummy_cols(age.NA.train)
age.NA.train <- age.NA.train[,-which(names(age.NA.train) %in% extra_cols)]
age.NA.test <- dummy_cols(age.NA.test)
age.NA.test <- age.NA.test[,-which(names(age.NA.test) %in% extra_cols)]
```

The following two functions will be used to run 4-fold cross validation on the age.train set with 5 repetitions to find the best imputation method. A comparison of the test errors will be used to determine which process is best for imputing the data.

```
nnProcedure <- function(df, folds=2, rep=1, threshold=0.01, stepmax=10000, nodes=1,
                        nnrep=1){
    # Create Result Variable
    result <- NULL

    # Loop through rep times
    for (i in 1:rep){

        # Shuffle df
        df <- df[sample(nrow(df)),]

        # Split data into folds
        fold_n <- nrow(df)/folds
        for (i in 1:folds){
```

```
            lower_bound <- floor(fold_n*(i-1)+1)
            upper_bound <- floor(fold_n*(i))
            testing.data <- df[lower_bound:upper_bound,]
            training.data <- df[-(lower_bound:upper_bound),]


            # Neural Network
            nnRes <- neuralnet(Age~.,training.data,hidden=nodes,linear.output=TRUE,
                               threshold=threshold,stepmax=stepmax, rep=nnrep)

            # Loop through all neural networks that converge
            for (n in 1:length(nnRes$net.result)){
                if (!is.null(nnRes$net.result[[n]])){
                    # Train Error
                    train_pred <- nnRes$net.result[[n]][,1]
                    train_actual <- training.data$Age
                    train_MSE <- sum((train_pred - train_actual)^2)/nrow(training.data)

                    # Test Error
                    test_pred <- predict(nnRes,testing.data)[,1]
                    test_actual <- testing.data$Age
                    test_MSE <- sum((test_pred - test_actual)^2)/nrow(testing.data)

                    # Create a result vector with variables and errors for plotting
                    result_vec <- c(nodes, train_MSE, test_MSE)
                    result <- rbind(result, result_vec)
                }
            }
        }
    }
    # Return dataframe of result
    result <- as.data.frame(result)
    names(result) <- c("Nodes", "Train MSE", "Test MSE")
    return (result)
}
```

```
lrProcedure <- function(df, folds=2, rep=1){
    # Create Result Variable
    result <- NULL

    # Loop through rep times
    for (i in 1:rep){

        # Shuffle df
        df <- df[sample(nrow(df)),]

        # Split data into folds
        fold_n <- nrow(df)/folds
        for (i in 1:folds){
            lower_bound <- floor(fold_n*(i-1)+1)
            upper_bound <- floor(fold_n*(i))
            testing.data <- df[lower_bound:upper_bound,]
            training.data <- df[-(lower_bound:upper_bound),]
```

```
            # Linear Regression
            reg <- lm(Age~., training.data)

            # Train Error
            train_pred <- reg$fitted.values
            train_actual <- training.data$Age
            train_MSE <- sum((train_pred - train_actual)^2)/nrow(training.data)

            # Test Error
            test_pred <- predict(reg,testing.data)
            test_actual <- testing.data$Age
            test_MSE <- sum((test_pred - test_actual)^2)/nrow(testing.data)

            # Create a result vector with variables and errors for plotting
            result_vec <- c(train_MSE, test_MSE)
            result <- rbind(result, result_vec)
        }
    }
    # Return dataframe of result
    result <- as.data.frame(result)
    names(result) <- c("Train MSE", "Test MSE")
    return (result)
}
```

```
MeanMedProcedure <- function(df, folds=2, rep=1){
    # Create Result Variable
    result <- NULL

    # Loop through rep times
    for (i in 1:rep){

        # Shuffle df
        df <- df[sample(nrow(df)),]

        # Split data into folds
        fold_n <- nrow(df)/folds
        for (i in 1:folds){
            lower_bound <- floor(fold_n*(i-1)+1)
            upper_bound <- floor(fold_n*(i))
            testing.data <- df[lower_bound:upper_bound,]
            training.data <- df[-(lower_bound:upper_bound),]

            # Train Error
            mean_pred <- mean(training.data$Age)
            median_pred <- median(training.data$Age)
            train_actual <- training.data$Age
            mean_train_MSE <- sum((mean_pred - train_actual)^2)/nrow(training.data)
            median_train_MSE <- sum((median_pred - train_actual)^2)/nrow(training.data)

            # Test Error
            test_actual <- testing.data$Age
            mean_test_MSE <- sum((mean_pred - test_actual)^2)/nrow(testing.data)
            median_test_MSE <- sum((median_pred - test_actual)^2)/nrow(testing.data)
```

```
            # Create a result vector with variables and errors for plotting
            result_vec <- c(mean_train_MSE, mean_test_MSE, median_train_MSE,
                            median_test_MSE)
            result <- rbind(result, result_vec)
        }
    }
    # Return dataframe of result
    result <- as.data.frame(result)
    names(result) <- c("Mean Train MSE", "Mean Test MSE", "Median Train MSE",
                       "Median Test MSE")
    return (result)
}
```

```
MeanMedTitleProcedure <- function(df, folds=2, rep=1){
    # Create Result Variable
    result <- NULL

    # Loop through rep times
    for (i in 1:rep){

        # Shuffle df
        df <- df[sample(nrow(df)),]

        # Split data into folds
        fold_n <- nrow(df)/folds
        for (i in 1:folds){
            lower_bound <- floor(fold_n*(i-1)+1)
            upper_bound <- floor(fold_n*(i))
            testing.data <- df[lower_bound:upper_bound,]
            training.data <- df[-(lower_bound:upper_bound),]

            # Get means/medians of each title
            mean_pred <- numeric()
            median_pred <- numeric()
            for (title in sort(unique(training.data$Title))){
                rows <- training.data[training.data$Title == title,
                                      which(names(training.data)=="Age")]
                mean_pred <- c(mean_pred, mean(rows))
                median_pred <- c(median_pred, median(rows))
            }
            pred_df = data.frame(means = mean_pred, medians = median_pred, row.names = sort(unique(trai

            # Train Error
            mean_train_pred <- apply(training.data, 1, function (x) pred_df[x["Title"],
                                                                            "means"])
            median_train_pred <- apply(training.data, 1, function (x) pred_df[x["Title"],
                                                                              "medians"])
            train_actual <- training.data$Age
            mean_train_MSE <- sum((mean_train_pred - train_actual)^2)/nrow(training.data)
            median_train_MSE <- sum((median_train_pred - train_actual)^2)/nrow(training.data)

            # Test Error
            mean_test_pred <- apply(testing.data, 1, function (x) pred_df[x["Title"],
```

```
                                                                    "means"])
            median_test_pred <- apply(testing.data, 1, function (x) pred_df[x["Title"],
                                                                    "medians"])
            test_actual <- testing.data$Age
            mean_test_MSE <- sum((mean_test_pred - test_actual)^2)/nrow(testing.data)
            median_test_MSE <- sum((median_test_pred - test_actual)^2)/nrow(testing.data)

            # Create a result vector with variables and errors for plotting
            result_vec <- c(mean_train_MSE, mean_test_MSE, median_train_MSE,
                            median_test_MSE)
            result <- rbind(result, result_vec)
        }
    }
    # Return dataframe of result
    result <- as.data.frame(result)
    names(result) <- c("Mean Train MSE", "Mean Test MSE", "Median Train MSE",
                       "Median Test MSE")
    return (result)
}
```

```
MeanMedTitleClassProcedure <- function(df, folds=2, rep=1){
    # Create Result Variable
    result <- NULL

    # Loop through rep times
    for (i in 1:rep){

        # Shuffle df
        df <- df[sample(nrow(df)),]

        # Split data into folds
        fold_n <- nrow(df)/folds
        for (i in 1:folds){
            lower_bound <- floor(fold_n*(i-1)+1)
            upper_bound <- floor(fold_n*(i))
            testing.data <- df[lower_bound:upper_bound,]
            training.data <- df[-(lower_bound:upper_bound),]

            # Get means/medians of each title
            titles <- character()
            pclasses <- numeric()
            mean_pred <- numeric()
            median_pred <- numeric()
            for (title in sort(unique(training.data$Title))){
              for (pclass in sort(unique(training.data$Pclass))){
                rows <- training.data[training.data$Title==title & training.data$Pclass==pclass,
                                      which(names(training.data)=="Age")]
                titles <- c(titles, title)
                pclasses <- c(pclasses, pclass)
                mean_pred <- c(mean_pred, mean(rows))
                median_pred <- c(median_pred, median(rows))
              }
            }
```

```
            mean_pred_matrix = matrix(mean_pred, nrow=length(unique(training.data$Pclass)),
                                      dimnames=list(sort(unique(training.data$Pclass)),
                                                    sort(unique(training.data$Title))))
            median_pred_matrix = matrix(median_pred,
                                        nrow=length(unique(training.data$Pclass)),
                                        dimnames=list(sort(unique(training.data$Pclass)),
                                                      sort(unique(training.data$Title))))

            # Train Error
            mean_train_pred <- apply(training.data, 1,
                                     function (x) mean_pred_matrix[x["Pclass"],
                                                                   x["Title"]])
            median_train_pred <- apply(training.data, 1,
                                       function (x) median_pred_matrix[x["Pclass"],
                                                                       x["Title"]])
            train_actual <- training.data$Age
            mean_train_MSE <- sum((mean_train_pred - train_actual)^2)/nrow(training.data)
            median_train_MSE <- sum((median_train_pred - train_actual)^2)/nrow(training.data)

            # Test Error
            mean_test_pred <- apply(testing.data, 1,
                                    function (x) mean_pred_matrix[x["Pclass"],
                                                                  x["Title"]])
            median_test_pred <- apply(testing.data, 1,
                                      function (x) median_pred_matrix[x["Pclass"],
                                                                      x["Title"]])
            test_actual <- testing.data$Age
            mean_test_MSE <- sum((mean_test_pred - test_actual)^2)/nrow(testing.data)
            median_test_MSE <- sum((median_test_pred - test_actual)^2)/nrow(testing.data)

            # Create a result vector with variables and errors for plotting
            result_vec <- c(mean_train_MSE, mean_test_MSE, median_train_MSE, median_test_MSE)
            result <- rbind(result, result_vec)
        }
    }
    # Return dataframe of result
    result <- as.data.frame(result)
    names(result) <- c("Mean Train MSE", "Mean Test MSE", "Median Train MSE",
                       "Median Test MSE")
    return (result)
}
```

```
# Mean/Median
set.seed(100)
mm <- MeanMedProcedure(age.train.no.dummies, folds=4, rep=5)
```

```
# Mean/Median by Title
set.seed(100)
mmByTitle <- MeanMedTitleProcedure(age.train.no.dummies, folds=4, rep=5)
```

```
# Mean/Median by Title and Pclass
set.seed(100)
mmByTitleClass <- MeanMedTitleClassProcedure(age.train.no.dummies, folds=4, rep=5)
```

```
# Linear regression
set.seed(100)
lr <- lrProcedure(age.train, folds=4, rep=5)
```

```
# Neural network with 1 hidden layer of 3 nodes
set.seed(100)
nn.3 <- nnProcedure(age.train, folds=4, rep=5, threshold=0.03, stepmax=150000, nodes=3)
```

```
## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.

## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.

## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.

## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
```

```
# Neural network with 1 hidden layer of 4 nodes
set.seed(100)
nn.4 <- nnProcedure(age.train, folds=4, rep=5, threshold=0.03, stepmax=150000, nodes=4)
```

```
# Neural network with 1 hidden layer of 5 nodes
set.seed(100)
nn.5 <- nnProcedure(age.train, folds=4, rep=5, threshold=0.03, stepmax=150000, nodes=5)
```

```
## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.

## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.

## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
```

```
# Neural network with 1 hidden layer of 6 nodes
set.seed(100)
nn.6 <- nnProcedure(age.train, folds=4, rep=5, threshold=0.03, stepmax=150000, nodes=6)
```

```
## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
```

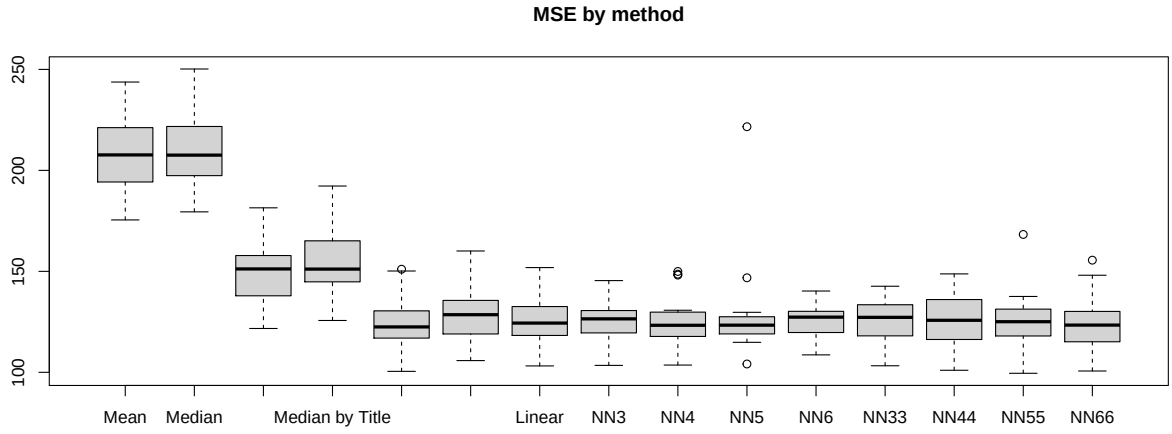
```
# Neural network with 2 hidden layers of 3 nodes each
set.seed(100)
nn.33 <- nnProcedure(age.train, folds=4, rep=5, threshold=0.03, stepmax=150000,
                     nodes=c(3,3))
```

```
# Neural network with 2 hidden layers of 4 nodes each
set.seed(100)
nn.44 <- nnProcedure(age.train, folds=4, rep=5, threshold=0.03, stepmax=150000,
                     nodes=c(4,4))
```

```
# Neural network with 2 hidden layers of 5 nodes each
set.seed(100)
nn.55 <- nnProcedure(age.train, folds=4, rep=5, threshold=0.03, stepmax=150000,
                     nodes=c(5,5))
```

```
# Neural network with 2 hidden layers of 6 nodes each
set.seed(100)
nn.66 <- nnProcedure(age.train, folds=4, rep=5, threshold=0.03, stepmax=150000,
                     nodes=c(6,6))
```

```
# Compare MSE
boxplot(mm[,2], mm[,4], mmByTitle[,2], mmByTitle[,4], mmByTitleClass[,2],
        mmByTitleClass[,4],
        lr[,2], nn.3[,3], nn.4[,3], nn.5[,3], nn.6[,3],nn.33[,4], nn.44[,4],
        nn.55[,4], nn.66[,4],
        names=c("Mean","Median", "Mean by Title", "Median by Title",
                "Mean by Title and Class",
                "Median by Title and Class", "Linear", "NN3","NN4","NN5","NN6",
                "NN33","NN44","NN55","NN66"),
        main="MSE by method")
```



Above I used 4 procedures for imputation. First, I replaced the missing values with the mean and then the median age of all passengers. Then, I replaced the age using the mean and then the median age of each title. Then I tried the same procedure, but I used both title and pclass. Next I tried linear regression. And lastly, I tried training a neural network. The neural networks, linear regression, and mean and median by class and title all produced test errors in the same range. Based on these results, I'm going to go with mean by pclass and title.

```
# Create matrix of mean age by title and pclass
titles <- character()
pclasses <- numeric()
mean_pred <- numeric()

for (title in sort(unique(age.train.no.dummies$Title))){
  for (pclass in sort(unique(age.train.no.dummies$Pclass))){
    rows <- age.train.no.dummies[age.train.no.dummies$Title==title & age.train.no.dummies$Pclass==pclass,
                         which(names(age.train.no.dummies)=="Age")]
    titles <- c(titles, title)
    pclasses <- c(pclasses, pclass)
    mean_pred <- c(mean_pred, mean(rows))
    }
```

```
}
mean_pred_matrix = matrix(mean_pred, nrow=length(unique(age.train.no.dummies$Pclass)),
                          dimnames=list(sort(unique(age.train.no.dummies$Pclass)),
                                        sort(unique(age.train.no.dummies$Title))))

# Impute missing data
train[is.na(train$Age), which(names(train)=="Age")] <- apply(train[is.na(train$Age),],
                                                             1, function (x) mean_pred_matrix[x["Pclass
                                                                                              x["Title"]
test[is.na(test$Age), which(names(test)=="Age")] <- apply(test[is.na(test$Age),],
                                                          1, function (x) mean_pred_matrix[x["Pclass"],
                                                                                           x["Title"]])

# Show missing data is filled
paste("There are", sum(is.na(train$Age)),
      "missing values in the age column of the training set")
```

```
## [1] "There are 0 missing values in the age column of the training set"
```

```
paste("There are", sum(is.na(test$Age)),
      "missing values in the age column of the testing set")
```

```
## [1] "There are 1 missing values in the age column of the testing set"
```

Unfortunately there is still one missing value left in the testing set. This is because there are no available ages for passengers of pclass 3 and title "Other". We will replace this one missing value with the mean age of all female and "Other" titled passengers.

```
# View entry with missing data
train[train$Pclass==3 & train$Title=="Other",]
```

```
##  [1] Survived     Pclass       Sex          Age          Fare
##  [6] Embarked     FamSize      CabinLetter  TicketLetter Title
## <0 rows> (or 0-length row.names)
```

```
test[test$Pclass==3 & test$Title=="Other",]
```

```
##    Pclass    Sex Age Fare Embarked FamSize CabinLetter TicketLetter Title
## 89      3 female NaN 7.75        Q    Solo     Unknown            3 Other
```

```
# Replace missing value with mean age of all female and "Other" titled passengers
full_data <- rbind(train[,-which(names(train)=="Survived")],test)
test[test$Pclass==3 & test$Title=="Other", "Age"] <- mean(full_data[full_data$Sex=="female" & full_data$
                                                          na.rm = TRUE)

# Show missing data is filled
paste("There are", sum(sapply(test, is.na)), "missing values in the training set")
```

```
## [1] "There are 0 missing values in the training set"
```

```
paste("There are", sum(sapply(train, is.na)), "missing values in the testing set")
```

```
## [1] "There are 0 missing values in the testing set"
```

Before we begin our classification methods, we're going to write some functions to assist with our analysis. The assess.prediction function does exactly what the name suggests, it helps up assess the accuracy of our predictions made by a classification method. The function takes the true values and the predicted values and prints out the sensitivity, specificity, precision, false discovery, false positive rate, and a confusion matrix. The assess.crossval.prediction function is similar to the assess prediction function, except it will be used to record all of the previously listed results while performing cross validation in the crossValClassifier function. Lastly, the crossValClassifier function performs cross validation for a given method, kfolds, and repetitions. It returns a matrix of model stats calculated by the access.crossval.prediction function which can be used for plotting. The number of rows in the matrix will equal the number of folds times the number of repetitions. This function works for the following classifying methods: logistic regression, LDA, QDA, KNN, Naive Bayes, and Random Forest.

```
assess.crossval.prediction <- function(truth,predicted, Tval=1, Fval=0) {
    # Calculate Stats
    Accuracy <- signif(sum(truth==predicted)*100/ length(truth),3)
    Error <- signif(sum(truth!=predicted)*100/ length(truth),3)
    TP <- sum(truth==Tval & predicted==Tval)
    TN <- sum(truth==Fval & predicted==Fval)
    FP <- sum(truth==Fval & predicted==Tval)
    FN <- sum(truth==Tval & predicted==Fval)
    P <- TP+FN # total number of positives in the truth data
    N <- FP+TN # total number of negatives
    sens <- signif(100*TP/P,3)
    spec <- signif(100*TN/N,3)
    PPV <- signif(100*TP/(TP+FP),3)
    FDR <- 1-PPV

    # Create Result Vector
    result <- c(Accuracy, Error, TP, TN, FP, FN, sens, spec, PPV, FDR)
    names(result) <- c("Accuracy", "Error", "TP", "TN", "FP", "FN", "Sensitivity",
                       "Specificity", "PPV", "FDR")
    return(result)
}

# Function returns matrix of classifying stats fold * rep times
crossValClassifier <- function(df, folds, method, rep=1, k=1, Tval=1, Fval=0,
                               response="Y", thres=0.5){
    # df is a dataframe (with response variable as response)
    # folds is the number of folds to perform cross validation on
    # method is the classifying method
    # rep is the number of repetitions
    # k is used for KNN classifier

    # Create Result Variable
    result <- NULL

    # Loop through repetitions
    for (r in 1:rep){
```

```
    # Shuffle df
    df <- df[sample(nrow(df)),]

    # Split data into folds
    fold_n <- nrow(df)/folds
    for (i in 1:folds){
        lower_bound <- floor(fold_n*(i-1)+1)
        upper_bound <- floor(fold_n*(i))
        testing.data <- df[lower_bound:upper_bound,]
        training.data <- df[-(lower_bound:upper_bound),]

        # Run classifying method
        # Logistic Regression
        if (method == "Logistic"){
            glm.fit <- glm(as.formula(paste0(response,"~.")),data=training.data,family=binomial)
            glm.probabilities <- predict(glm.fit,newdata=testing.data[,-which(names(testing.data)==
                                         type="response")
            predictions <- as.factor(ifelse(glm.probabilities > thres, Tval, Fval))
            result <- rbind(result, assess.crossval.prediction(testing.data[,which(names(testing.da
                                                               predictions))
        }

        # LDA
        if (method == "LDA"){
            lda.fit <- lda(as.formula(paste0(response,"~.")),data=training.data)
            lda.probabilities <- predict(lda.fit,newdata=testing.data[,-which(names(testing.data)==
            predictions <- as.factor(ifelse(lda.probabilities > thres, Tval, Fval))
            result <- rbind(result, assess.crossval.prediction(testing.data[,which(names(testing.da
                                                               predictions))
        }

        # QDA
        if (method == "QDA"){
            qda.fit <- qda(as.formula(paste0(response,"~.")),data=training.data)
            qda.probabilities <- predict(qda.fit,newdata=testing.data[,-which(names(df)==response)]
            predictions <- as.factor(ifelse(qda.probabilities > thres, Tval, Fval))
            result <- rbind(result, assess.crossval.prediction(testing.data[,which(names(testing.da
                                                               predictions))
        }

        # KNN
        if (method == "KNN"){
            predictions <- knn(train=training.data[,-which(names(training.data)==response)],
                               test=testing.data[,-which(names(testing.data)==response)],
                               cl=training.data[,which(names(training.data)==response)], k=k)
            result <- rbind(result, assess.crossval.prediction(testing.data[,which(names(testing.da
                                                               predictions,
                                                               Tval=Tval, Fval=Fval))
        }

        # Naive Bayes
        if (method == "Naive"){
            nb.fit <- naiveBayes(as.formula(paste0(response,"~.")),data=training.data)
```

```
            predictions <- predict(nb.fit,newdata=testing.data[,-which(names(df)==response)],
                                   type="class")
            result <- rbind(result, assess.crossval.prediction(testing.data[,which(names(testing.da
                                                               predictions,
                                                               Tval=Tval, Fval=Fval))
        }

        # Random Forest
        if (method == "Random Forest"){
            rf.fit <- randomForest(training.data[,-which(names(training.data)==response)],
                                   training.data[,which(names(training.data)==response)])
            predictions <- predict(rf.fit,newdata=testing.data[,-which(names(testing.data)==respons
            result <- rbind(result, c(assess.crossval.prediction(testing.data[,which(names(testing.
                                      "OOB Error"=100*mean(rf.fit$err.rate[,1])))
        }
    }
}

    return(result)
}
```

Below we will run 2-fold cross validation repeated 25 times to see which method best predicts survival on the training data. We use 2-fold cross validation because the testing set is roughly half of the training set. The classifying methods inclue logistic regression, linear discriminant analysis, quadratic discriminant analysis (on two different subsets of the predictors), knn (using k = 1,5,10, and 25), naive bayes, and random forest.

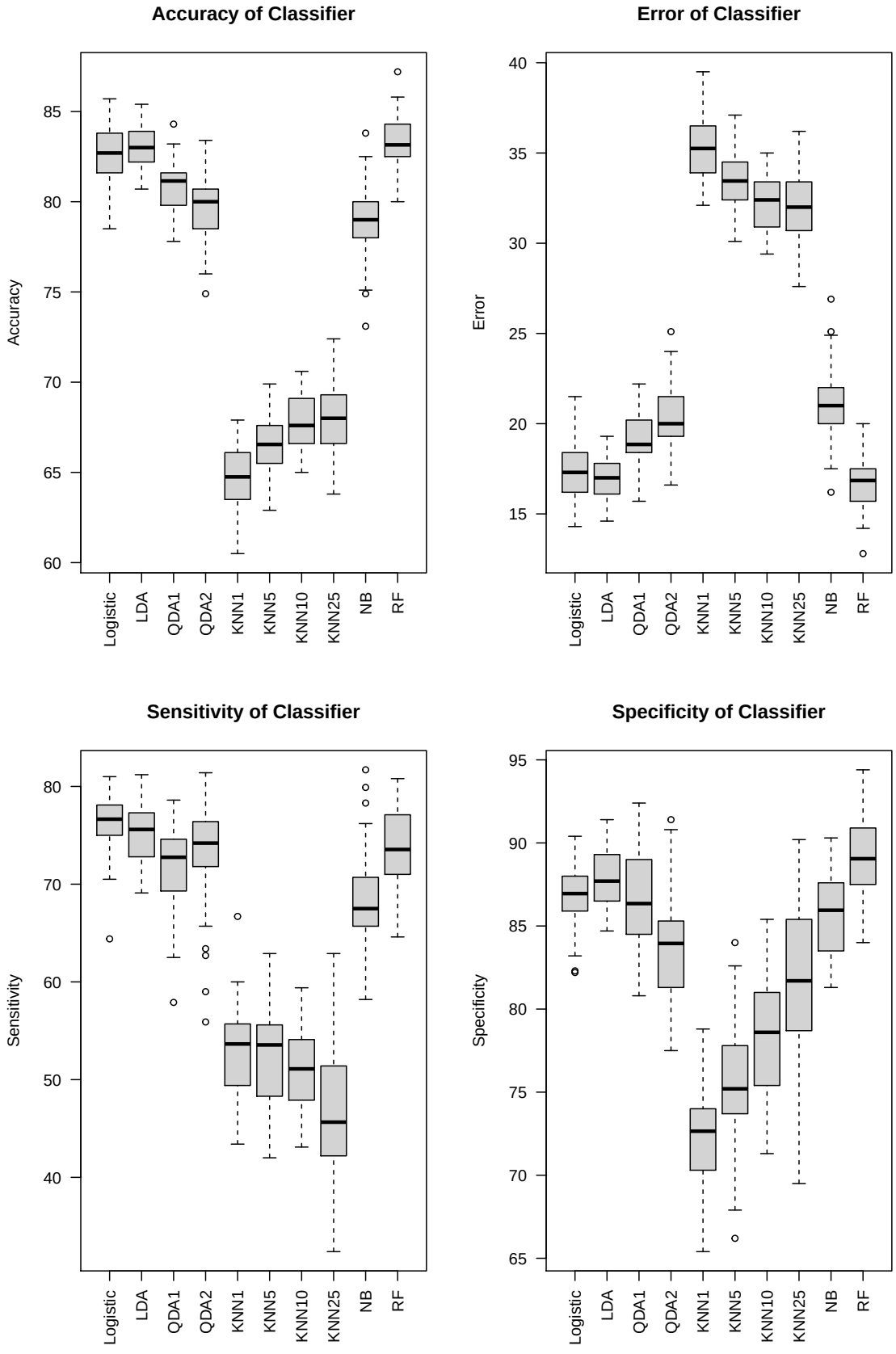
```
set.seed(100)

# Run Cross Validation for different classifying methods
cv.logistic <- crossValClassifier(df=train, folds=2, method="Logistic", rep=25,
                                  response="Survived")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
cv.lda <- crossValClassifier(df=train, folds=2, method="LDA", rep=25, response="Survived")
cv.qda1 <- crossValClassifier(df=train[,c("Survived", "Age", "Fare", "Pclass", "Title", "Embarked", "Fa
                              folds=2, method="QDA", rep=25, response="Survived")
cv.qda2 <- crossValClassifier(df=train[,c("Survived", "Age", "Fare", "Pclass", "Sex", "Embarked", "FamS
                              folds=2, method="QDA", rep=25, response="Survived")
cv.k1 <- crossValClassifier(train[,c("Survived", continuous_predictors)], folds=2,
                            method="KNN", rep=25, k=1, response="Survived")
cv.k5 <- crossValClassifier(train[,c("Survived", continuous_predictors)], folds=2,
                            method="KNN", rep=25, k=5, response="Survived")
cv.k10 <- crossValClassifier(train[,c("Survived", continuous_predictors)], folds=2,
                             method="KNN", rep=25, k=10, response="Survived")
cv.k25 <- crossValClassifier(train[,c("Survived", continuous_predictors)], folds=2,
                             method="KNN", rep=25, k=25, response="Survived")
cv.naive <- crossValClassifier(df=train, folds=2, method="Naive", rep=25,
                               response="Survived")
cv.rf <- crossValClassifier(df=train, folds=2, method="Random Forest", rep=25,
                            response="Survived")
```

Next, we'll plot the test errors and compare

```
# Create Box Plots
my_par <- par(mfrow=c(2,2))
for (val in c("Accuracy", "Error", "Sensitivity", "Specificity")){
  boxplot(cv.logistic[,val], cv.lda[,val], cv.qda1[,val], cv.qda2[,val], cv.k1[,val],
          cv.k5[,val], cv.k10[,val], cv.k25[,val], cv.naive[,val], cv.rf[,val],
          ylab=paste(val),
          names=c("Logistic", "LDA", "QDA1", "QDA2", "KNN1", "KNN5", "KNN10",
                  "KNN25", "NB", "RF"),
          main=paste(val, "of Classifier"), las=2)
}
```


Accuracy of Classifier
Accuracy
60
65
70
75
80
85
Logistic
LDA
QDA1
QDA2
KNN1
KNN5
KNN10
KNN25
NB
RF
Error of Classifier
Error
15
20
25
30
35
40
Logistic
LDA
QDA1
QDA2
KNN1
KNN5
KNN10
KNN25
NB
RF
Sensitivity of Classifier
Sensitivity
40
50
60
70
80
Logistic
LDA
QDA1
QDA2
KNN1
KNN5
KNN10
KNN25
NB
RF
Specificity of Classifier
Specificity
65
70
75
80
85
90
95
Logistic
LDA
QDA1
QDA2
KNN1
KNN5
KNN10
KNN25
NB
RF

```
par(my_par)
```

We can see that Logistic Regression, LDA, and Random Forest produce very similar results. Either would be acceptable to use. We'll use random forest for our submission!

```
rf.fit <- randomForest(train[,-1],train$Survived)
final.prediction <-data.frame(PassengerId=id.test, Survived=predict(rf.fit,newdata=test))
write.csv(final.prediction, "submission.csv", row.names = FALSE)
```