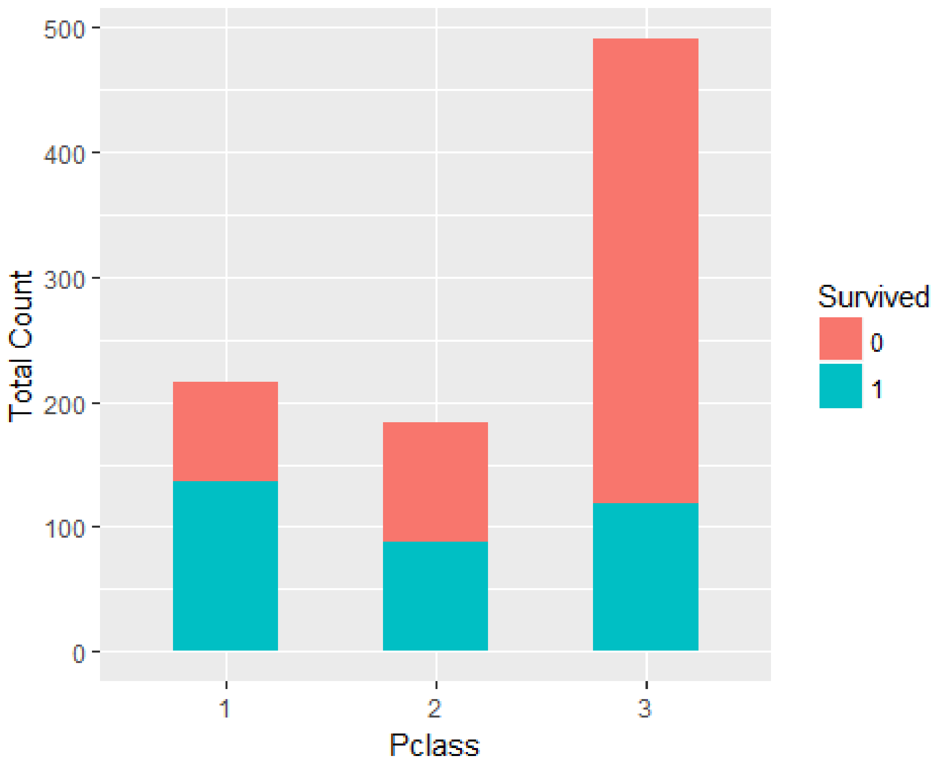
Titanic Detailed Analysis

Kushal Kapadia

October 7, 2017

```r
# Loading the new data
train = read.csv("train.csv", header = T)
test = read.csv("test.csv", header = T)

# Combining the dataset the old way by adding a column instead of doing
bind_rows fom dplyr package
test.survived = data.frame(Survived = rep("None", nrow(test)), test[,])
data.combined = rbind(train,test.survived)

# Let's have a close look at the structure of the data
str(data.combined)

## 'data.frame':    1309 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : chr  "0" "1" "1" "1" ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 1307 levels "Abbing, Mr. Anthony",..: 109 191
358 277 16 559 520 629 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1
...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 929 levels "110152","110413",..: 524 597 670 50
473 276 86 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 187 levels "","A10","A14",..: 1 83 1 57 1 1 131
1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...

# Letting R know the categorical variables in the dataset
data.combined$Survived = as.factor(data.combined$Survived)
data.combined$Pclass = as.factor(data.combined$Pclass)

# Let's have a general look at how many survived and how many did not
table(data.combined$Survived)

##
##    0    1 None
##  549  342  418

# Distribution across classes
table(data.combined$Pclass)
```

```
##
##   1   2   3
## 323 277 709

# Data visualization library
library(ggplot2)

train$Pclass = as.factor(train$Pclass)
ggplot(train, aes(x = Pclass, fill = factor(Survived))) + geom_bar(width =
0.5) + xlab("Pclass") + ylab("Total Count") + labs(fill="Survived")
```




```
# Examine the first few names in the training dataset
head(as.character(train$Name))

## [1] "Braund, Mr. Owen Harris"
## [2] "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
## [3] "Heikkinen, Miss. Laina"
## [4] "Futrelle, Mrs. Jacques Heath (Lily May Peel)"
## [5] "Allen, Mr. William Henry"
## [6] "Moran, Mr. James"

# how many unique names across both train and test dataset?
length(unique(as.character(data.combined$Name)))

## [1] 1307

#Thus, this shows us that there are two duplicate names
```

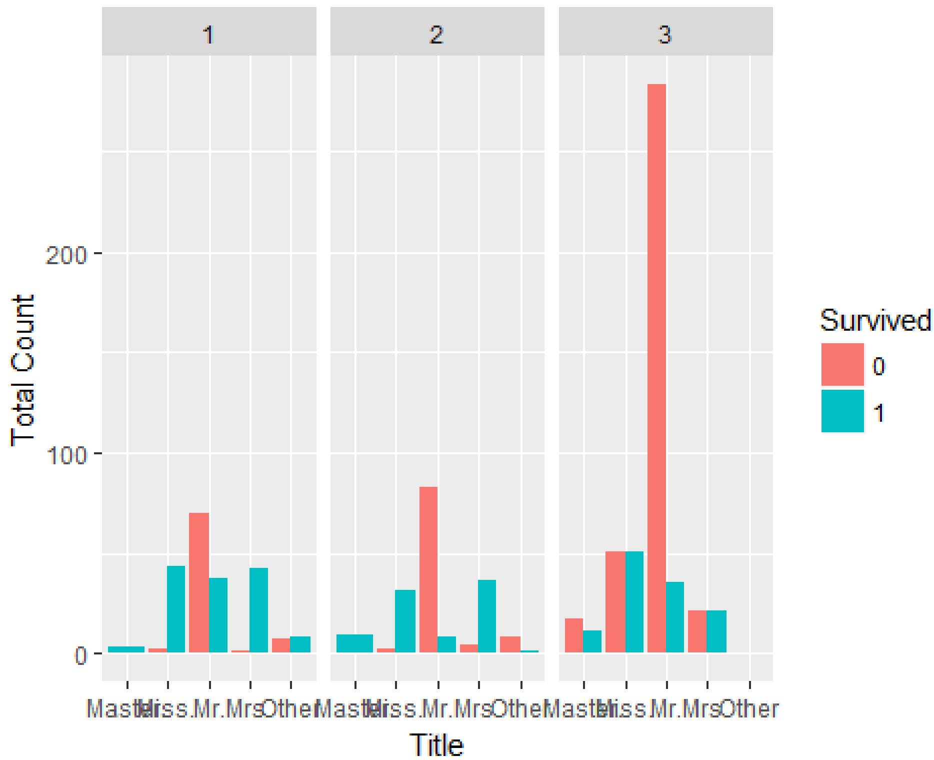
```r
# Finding the two duplicate names
which(duplicated(data.combined$Name))

## [1] 892 898

# One way to find the location of the two similar names
which(data.combined$Name == "Connolly, Miss. Kate")

## [1] 290 898

library(stringr)  #character extraction library

# Let's try to find the observations having only "Miss." using the str_detect
misses = data.combined[which(str_detect(data.combined$Name, "Miss.")), ]
misses[1:5,]

##    PassengerId Survived Pclass                                Name    Sex
## 3            3        1      3              Heikkinen, Miss. Laina female
## 11          11        1      3     Sandstrom, Miss. Marguerite Rut female
## 12          12        1      1            Bonnell, Miss. Elizabeth female
## 15          15        0      3 Vestrom, Miss. Hulda Amanda Adolfina female
## 23          23        1      3         McGowan, Miss. Anna "Annie" female
##    Age SibSp Parch           Ticket    Fare Cabin Embarked
## 3   26     0     0 STON/O2. 3101282  7.9250              S
## 11   4     1     1          PP 9549 16.7000    G6        S
## 12  58     0     0           113783 26.5500  C103        S
## 15  14     0     0           350406  7.8542              S
## 23  15     0     0           330923  8.0292              Q

# Let's also try to find the observations having only "Mrs." using the
str_detect
mrses = data.combined[which(str_detect(data.combined$Name, "Mrs.")),]
mrses[1:5,]

##    PassengerId Survived Pclass
## 2            2        1      1
## 4            4        1      1
## 9            9        1      3
## 10          10        1      2
## 16          16        1      2
##                                                   Name    Sex Age SibSp
## 2  Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1
## 4         Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1
## 9    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg) female  27     0
## 10                 Nasser, Mrs. Nicholas (Adele Achem) female  14     1
## 16                    Hewlett, Mrs. (Mary D Kingcome)  female  55     0
##    Parch   Ticket    Fare Cabin Embarked
## 2      0 PC 17599 71.2833   C85        C
## 4      0   113803 53.1000  C123        S
## 9      2   347742 11.1333              S
```

```
## 10     0   237736 30.0708              C
## 16     0   248706 16.0000              S

# check out if the pattern continues
males = data.combined[data.combined$Sex == "male",]
males[1:5,]

##   PassengerId Survived Pclass                           Name  Sex Age
## 1           1        0      3        Braund, Mr. Owen Harris male  22
## 5           5        0      3       Allen, Mr. William Henry male  35
## 6           6        0      3               Moran, Mr. James male  NA
## 7           7        0      1        McCarthy, Mr. Timothy J male  54
## 8           8        0      3 Palsson, Master. Gosta Leonard male   2
##   SibSp Parch    Ticket    Fare Cabin Embarked
## 1     1     0 A/5 21171  7.2500              S
## 5     0     0    373450  8.0500              S
## 6     0     0    330877  8.4583              Q
## 7     0     0     17463 51.8625   E46        S
## 8     3     1    349909 21.0750              S

# Now, we're gonna add variable "title" to the dataset
# Before adding the title variable, we first have to assign the title values
to all the observations

# Creating a utility function to help with title extraction
extractTitle = function(name){
  name = as.character(name)

  if(length(grep("Miss.",name))>0){
    return("Miss.")
  } else if (length(grep("Mrs.",name))>0){
    return("Mrs.")
  } else if (length(grep("Master.",name))>0){
    return("Master.")
  } else if (length(grep("Mr.",name))>0){
    return("Mr.")
  } else {
    return("Other")
  }
}

titles = NULL
for(i in 1:nrow(data.combined)){
  titles = c(titles, extractTitle(data.combined[i,"Name"]))
}
data.combined$title = as.factor(titles)
```

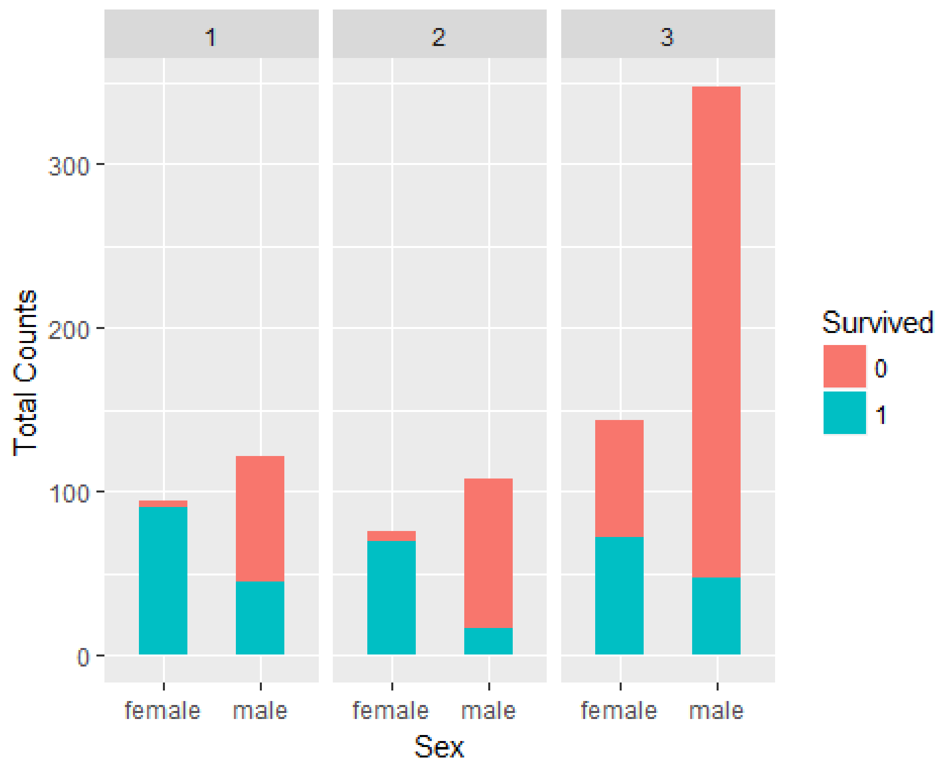
Data Visualization

```
par(mfrow = c(3,3))
```

```r
# Before ending this, let's also try visualizing data with 3 variables together namely survived, pclass and title
ggplot(data.combined[1:891,], aes(x = title , fill = Survived)) + #remember this to write stat=count and position=dodge
  geom_bar(stat = 'count', position = "dodge") +
  facet_wrap(~Pclass) +
  xlab("Title") +
  ylab("Total Count") +
  labs(fill = "Survived")
```

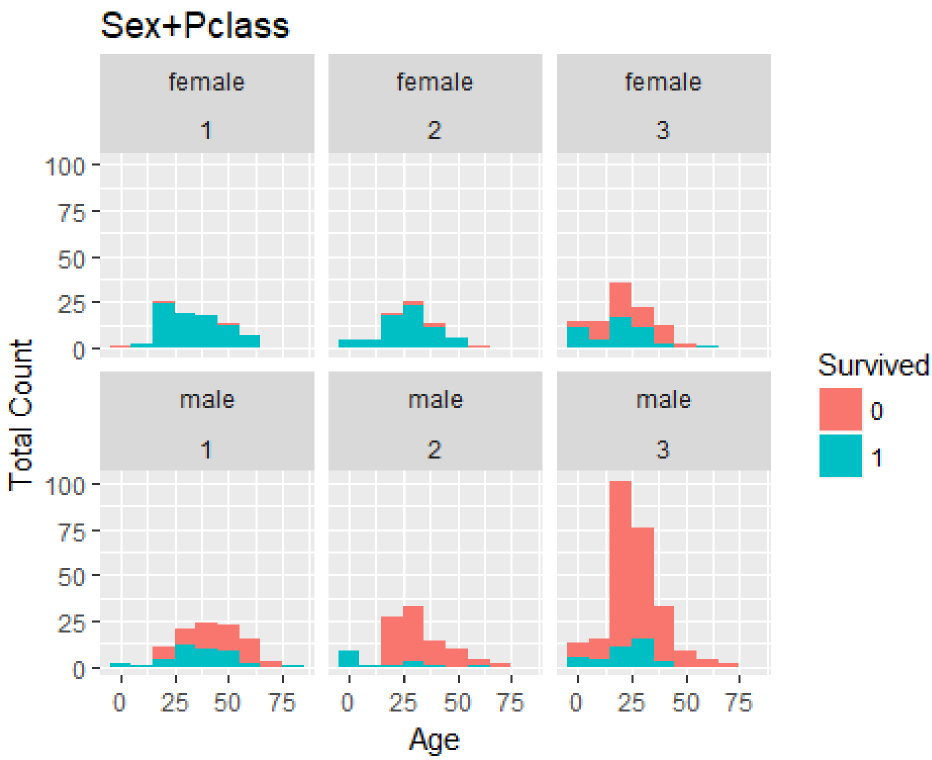



```r
# Distribution of male to female in the combined dataset
table(data.combined$Sex)

##
## female   male
##    466    843

# Now's let's visualize data a bit for pclass, sex and survived using ggplot
ggplot(data.combined[1:891,], aes(x = Sex, fill=Survived)) +
    geom_bar(width = 0.5) +
    facet_wrap(~Pclass) +
    xlab("Sex") +
    ylab("Total Counts") +
    labs(fill="Survived")
```



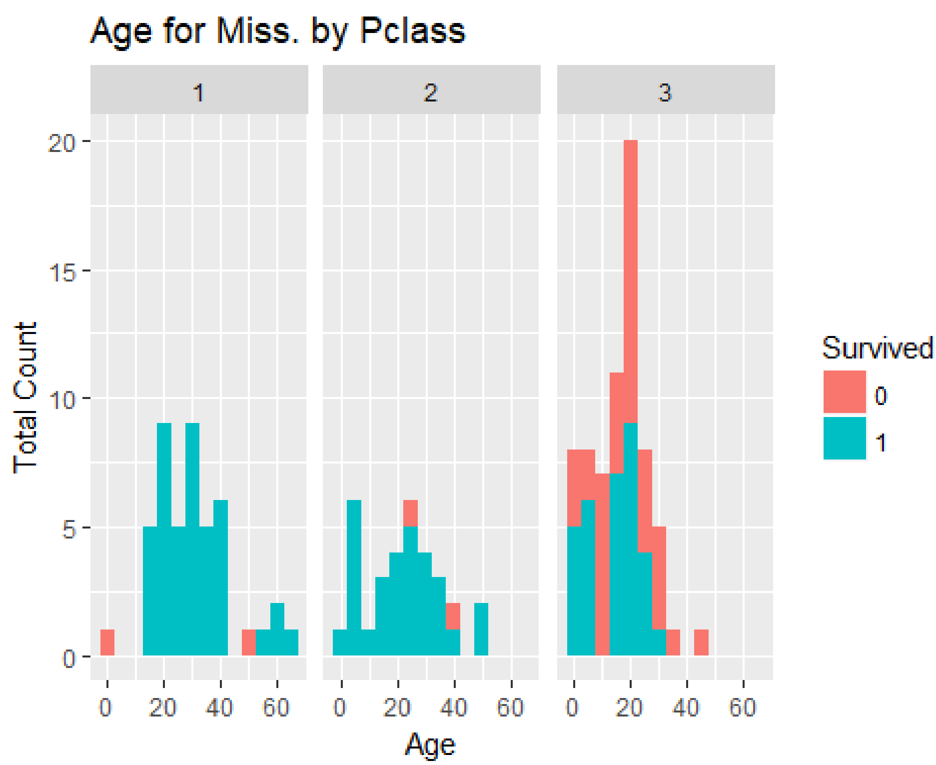

```
# Enough with the sex variable now. Age and sex seem to be related to each
other
# Let's explore the age variable a little bit more

summary(data.combined$Age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.17   21.00   28.00   29.88   39.00   80.00     263

summary(data.combined[1:891,"Age"]) #This actually indicates that there are
lot of missing values for age in the training data which is not a good thing.

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.42   20.12   28.00   29.70   38.00   80.00     177

# Ways to find missing values are mean, median, mean or median of the group,
imputation(basically training a model which would help us in getting the
missing values) and also proxy

# Let's visualize again by using age, sex, pclass and survived. Should be
interesting to code it! I'm excited!
ggplot(data.combined[1:891,], aes(x=Age, fill=Survived)) +
  geom_histogram(binwidth = 10) +
  facet_wrap(~Sex+Pclass) +
  ggtitle("Sex+Pclass") +
  xlab("Age") +
```

```
  ylab("Total Count") +
  labs(fill = "Survived")

## Warning: Removed 177 rows containing non-finite values (stat_bin).
```



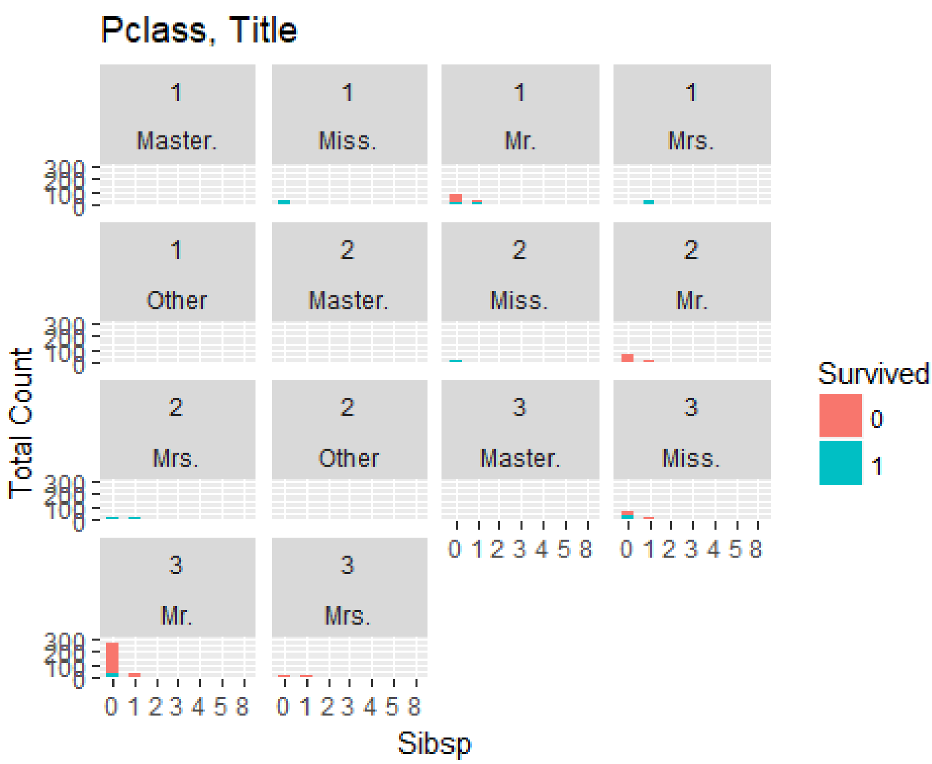

```
# Master is a good proxy for male children. Here's why:
boys = data.combined[which(data.combined$title == "Master."),]
summary(boys$Age) #This indeed confirms that the min age is 0.33 and max age
is 14.5 which means that they are male children

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.330   2.000   4.000   5.483   9.000  14.500       8

# Let's also delve deep into Miss title which is a bit complicated and we'll
see why:
misses = data.combined[data.combined$title == "Miss.",]
summary(misses$Age) # See, actually here the min age 0.17 rises to Max 63
which is why we can't say what they exactly are. Female children or female
adults

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.17   15.00   22.00   21.77   30.00   63.00      50

# Let's visualize something different using ggplot
ggplot(misses[misses$Survived != "None",], aes(x=Age, fill=Survived)) +
  facet_wrap(~Pclass) +
  geom_histogram(binwidth = 5) +
  ggtitle("Age for Miss. by Pclass") +
```

```
  xlab("Age") +
  ylab("Total Count")

## Warning: Removed 36 rows containing non-finite values (stat_bin).
```



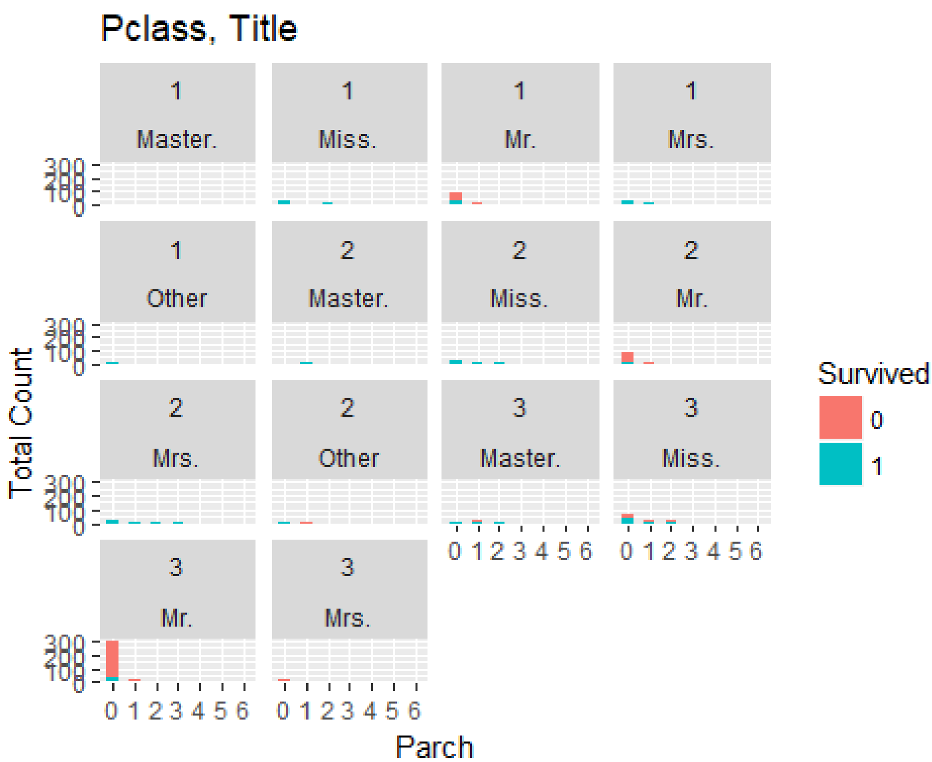

```
# Okay appears that female children might have a different survival rate
# Could be a candidate for feature engineering
misses.alone = misses[misses$SibSp == 0 & misses$Parch == 0,]
summary(misses.alone$Age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    5.00   21.00   26.00   27.23   32.50   58.00      33

length(which(misses.alone$Age <= 14.5))

## [1] 4

# Now, let's take a look at Sibsp variable
summary(data.combined$SibSp)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4989  1.0000  8.0000

table(data.combined$SibSp) #Not there in the video

##
##   0   1   2   3   4   5   8
## 891 319  42  20  22   6   9
```

```r
# As very evident from the 2 above lines of code, we can turn Sibsp into a factor
data.combined$SibSp = as.factor(data.combined$SibSp)

# Now that we know something about Sibsp, let's try visualizing it for a wee bit:
ggplot(data.combined[1:891,], aes(x = SibSp, fill = Survived)) +
  geom_bar(width = 0.5) +
  facet_wrap(~Pclass + title) +
  ggtitle("Pclass, Title") +
  xlab("Sibsp") +
  ylab("Total Count") +
    ylim(0,300) +
  labs(fill = "Survived")
```



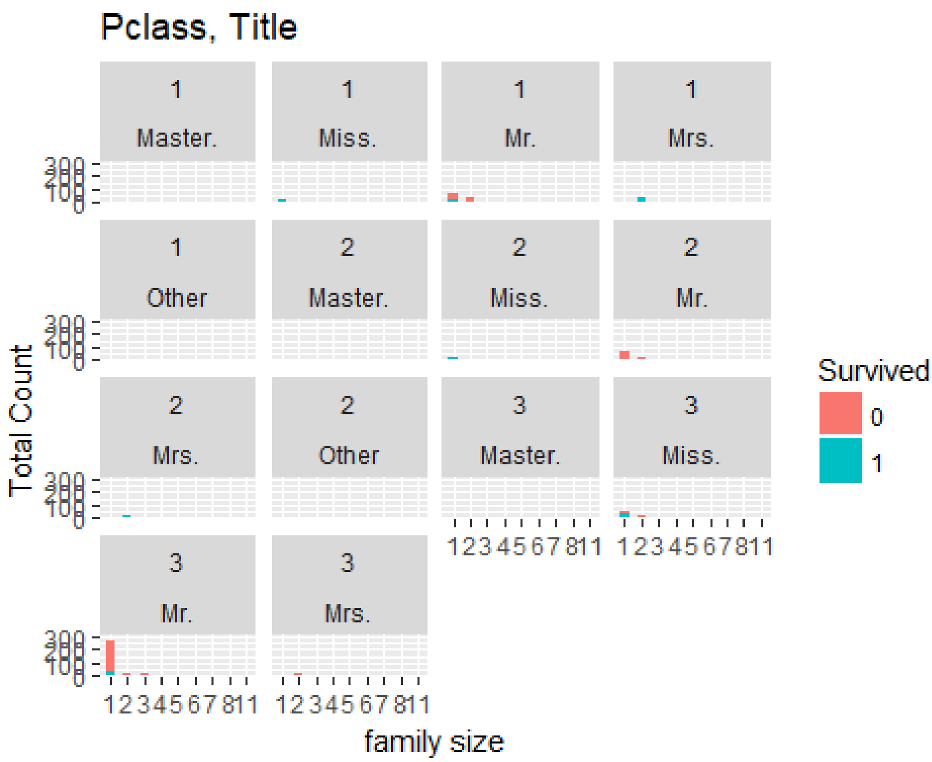

```r
# Let's have a look at parch variable which actually means parents or something. Ain't sure enough :P
unique(data.combined$Parch)

## [1] 0 1 2 5 3 4 6 9

table(data.combined$Parch)

##
##    0    1    2    3    4    5    6    9
## 1002  170  113    8    6    6    2    2
```

```r
# Let's go ahead and convert this also to a factor variable
data.combined$Parch = as.factor(data.combined$Parch)

# Same kind of visualization plot as Sibsp. Hence, Copying and  pasting it:
ggplot(data.combined[1:891,], aes(x = Parch, fill = Survived)) +
  geom_bar(width = 0.5) +
  facet_wrap(~Pclass + title) +
  ggtitle("Pclass, Title") +
  xlab("Parch") +
  ylab("Total Count") +
ylim(0,300) +
  labs(fill = "Survived")
```




```r
# Let's do some cool feature engineering here by creating a variable
familysize
# Let's first combine Sibsp and Parch from trainig and testing dataset
temp.Sibsp = c(train$SibSp, test$SibSp)
temp.Parch = c(train$Parch, test$Parch)
data.combined$familysize = as.factor(temp.Parch + temp.Sibsp + 1)

# Visualize it to see it has some predictive power in there or not:
ggplot(data.combined[1:891,], aes(x = familysize, fill = Survived)) +
  geom_bar(width = 0.5) +
  facet_wrap(~Pclass + title) +
  ggtitle("Pclass, Title") +
  xlab("family size") +
  ylab("Total Count") +
```

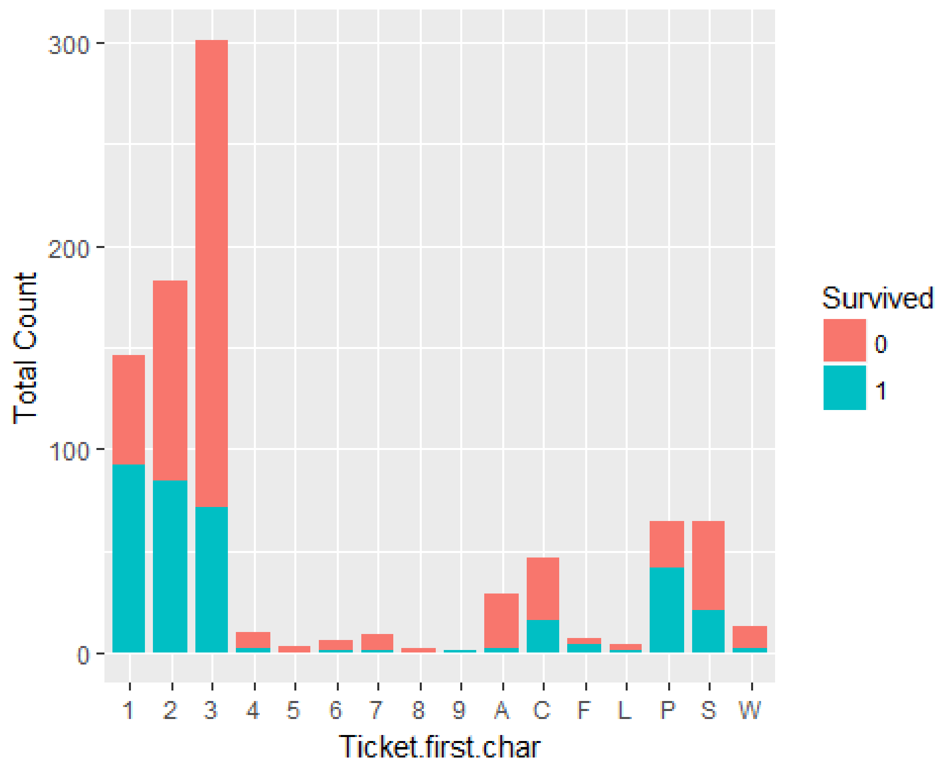
```r
ylim(0,300) +
  labs(fill = "Survived")
```



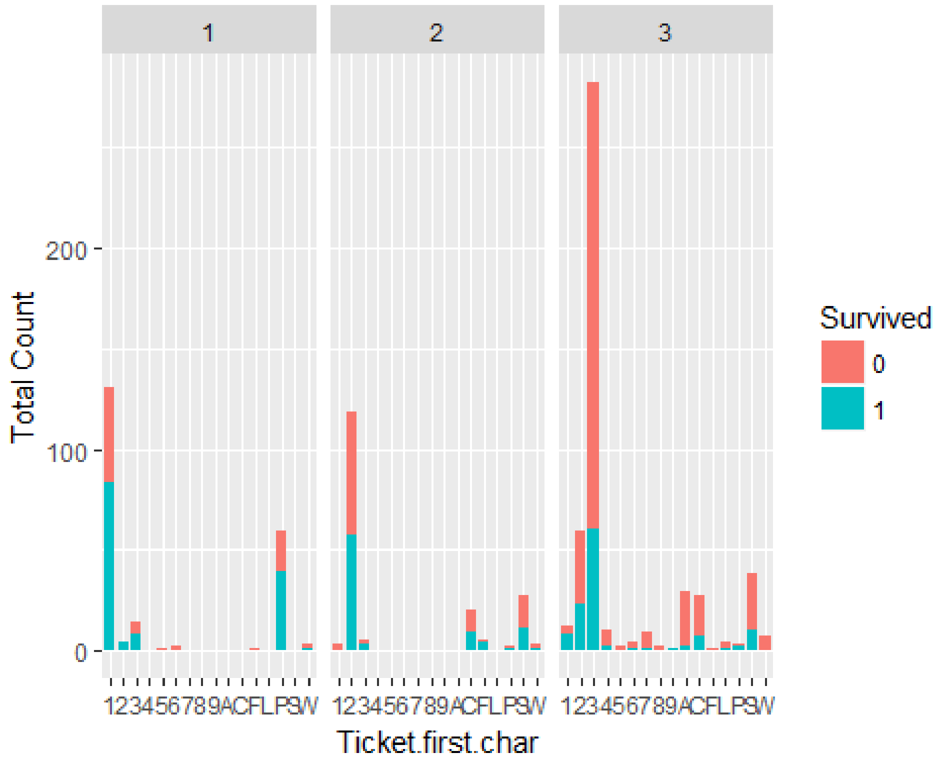

```r
par(mfrow = c(3,3))

# We need to look at the ticket variable

str(data.combined$Ticket)

##  Factor w/ 929 levels "110152","110413",..: 524 597 670 50 473 276 86 396
345 133 ...

summary(data.combined$Ticket)

##          CA. 2343             1601          CA 2144
##                11                8                8
##           3101295           347077           347082
##                 7                7                7
##          PC 17608    S.O.C. 14879           113781
##                 7                7                6
##             19950           347088           382652
##                 6                6                6
##            113503            16966           220845
##                 5                5                5
##            349909             4133         PC 17757
##                 5                5                5
##       W./C. 6608           113760            12749
```

```
##                5                 4                 4 
##            17421            230136             24160 
##                4                 4                 4 
##             2666             36928         C.A. 2315 
##                4                 4                 4 
##       C.A. 33112        C.A. 34651              LINE 
##                4                 4                 4 
##         PC 17483          PC 17755          PC 17760 
##                4                 4                 4 
##    SC/Paris 2123       W./C. 6607            110152 
##                4                 4                 3 
##           110413             11767             13502 
##                3                 3                 3 
##            19877             19928            230080 
##                3                 3                 3 
##           239853            248727            248738 
##                3                 3                 3 
##            26360              2650              2653 
##                3                 3                 3 
##             2661              2662              2668 
##                3                 3                 3 
##             2678             28220             29103 
##                3                 3                 3 
##            29106             29750            315153 
##                3                 3                 3 
##            33638            345773            347080 
##                3                 3                 3 
##           347742             35273            363291 
##                3                 3                 3 
##           367226            370129            371110 
##                3                 3                 3 
##         A/4 48871          A/5. 851            C 4001 
##                3                 3                 3 
##        C.A. 2673        C.A. 31921        C.A. 37671 
##                3                 3                 3 
##      F.C.C. 13529          PC 17558          PC 17569 
##                3                 3                 3 
##         PC 17572          PC 17582          PC 17756 
##                3                 3                 3 
##         PC 17758          PC 17761           PP 9549 
##                3                 3                 3 
##  S.C./PARIS 2079        C.A. 31029 SOTON/O.Q. 3101315 
##                3                 3                 3 
##           110465            110813            111361 
##                2                 2                 2 
##           112058            113059            113505 
##                2                 2                 2 
##           113509            113572            113773 
##                2                 2                 2 
##           113776            113789            113796 
```

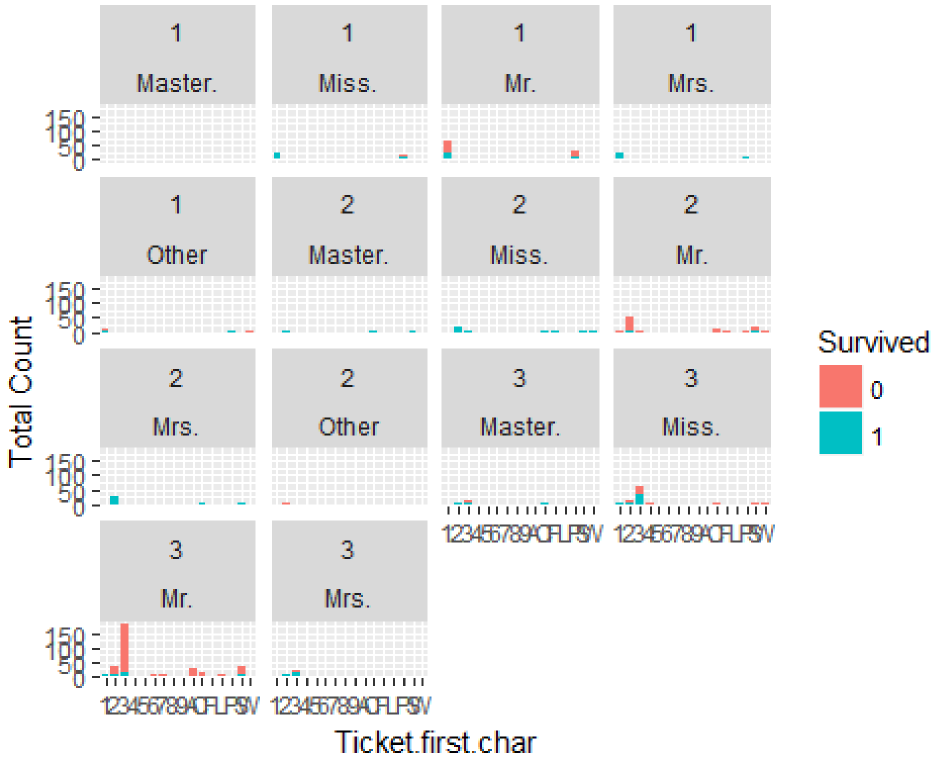
```
##                    2                   2                   2
##               113798              113803              113806
##                    2                   2                   2
##              (Other)
##                  947

# Thus, we're gonna transform this into a character variable
data.combined$Ticket = as.character(data.combined$Ticket)
data.combined$Ticket[1:20]

##  [1] "A/5 21171"          "PC 17599"           "STON/O2. 3101282"
##  [4] "113803"             "373450"             "330877"
##  [7] "17463"              "349909"             "347742"
## [10] "237736"             "PP 9549"            "113783"
## [13] "A/5. 2151"          "347082"             "350406"
## [16] "248706"             "382652"             "244373"
## [19] "345763"             "2649"

# Looking at a first few values, we can extract the first string to check if
something useful comes out
ticket.first.char = ifelse(data.combined$Ticket == "", " ",
substr(data.combined$Ticket,1,1))
unique(ticket.first.char)

##  [1] "A" "P" "S" "1" "3" "2" "C" "7" "W" "4" "F" "L" "9" "6" "5" "8"

# Okay so we can make it a factor for analysis purposes and visualize it
data.combined$ticket.first.char = as.factor(ticket.first.char)


# First, a high level plot of data
ggplot(data.combined[1:891,], aes(x = ticket.first.char, fill = Survived)) +
  geom_bar(width = 0.8) +
  ylab("Total Count") +
  xlab("Ticket.first.char") +
  labs(fill = "Survived")
```



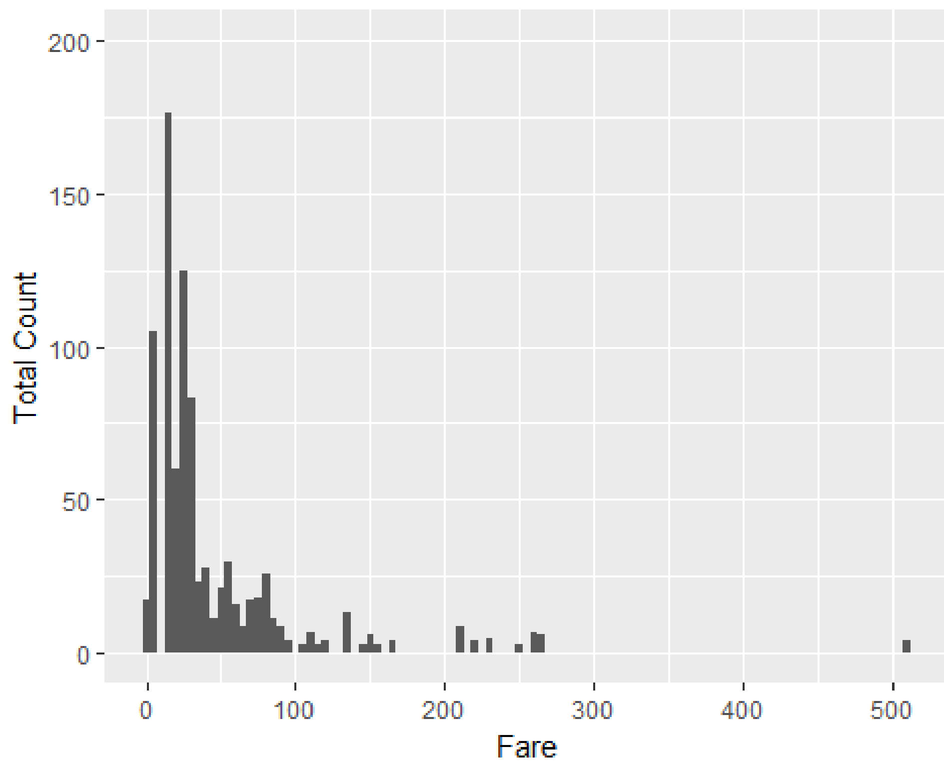

```r
# Using Pclass
ggplot(data.combined[1:891,], aes(x = ticket.first.char, fill = Survived)) +
  geom_bar(width = 0.8) +
  facet_wrap(~Pclass) +
  ylab("Total Count") +
  xlab("Ticket.first.char") +
  labs(fill = "Survived")
```



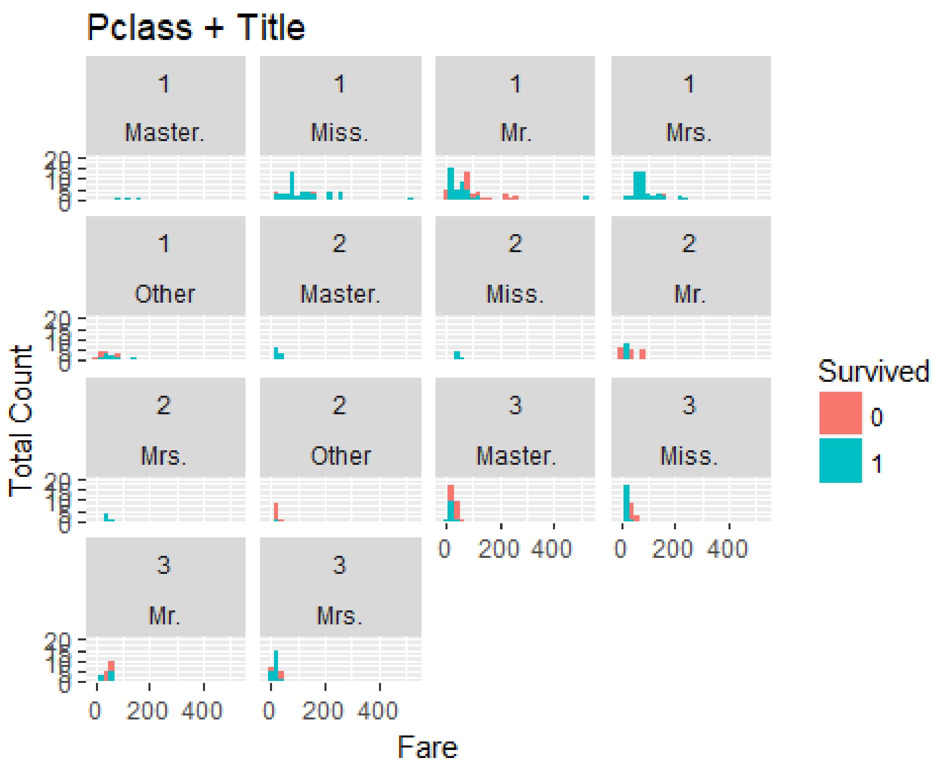

```
# Now using Pclass and Title both
  ggplot(data.combined[1:891,], aes(x = ticket.first.char, fill = Survived))
+
    geom_bar(width = 0.8) +
    facet_wrap(~Pclass + title) +
    ylab("Total Count") +
    xlab("Ticket.first.char") +
    labs(fill = "Survived")
```




```
# Next up is Fare
summary(data.combined$Fare)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   7.896  14.454  33.295  31.275 512.329       1

str(data.combined$Fare)    #numeric variable

##  num [1:1309] 7.25 71.28 7.92 53.1 8.05 ...

length(unique(data.combined$Fare))

## [1] 282

# We can relate the fair with Pclass
ggplot(data.combined, aes(x = Fare)) +
  geom_histogram(binwidth = 5) +
  xlab("Fare") +
  ylab("Total Count") +
  ylim(0,200)

## Warning: Removed 1 rows containing non-finite values (stat_bin).

## Warning: Removed 1 rows containing missing values (geom_bar).
```

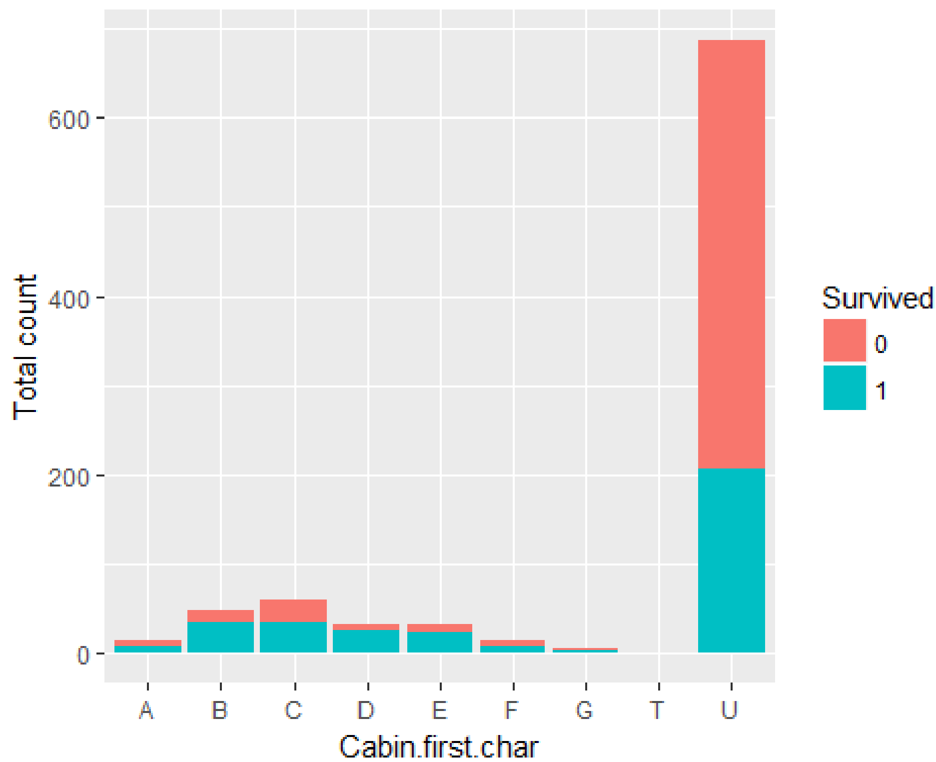



```
# Let's see if it has some predictive power or not
ggplot(data.combined[1:891,], aes(x = Fare, fill = Survived)) +
  geom_histogram(binwidth = 20) +
  facet_wrap(~Pclass + title) +
  xlab("Fare") +
  ylab("Total Count") +
  ggtitle("Pclass + Title") +
  labs(fill = "Survived") +
  ylim(0,20)

## Warning: Removed 15 rows containing missing values (geom_bar).
```

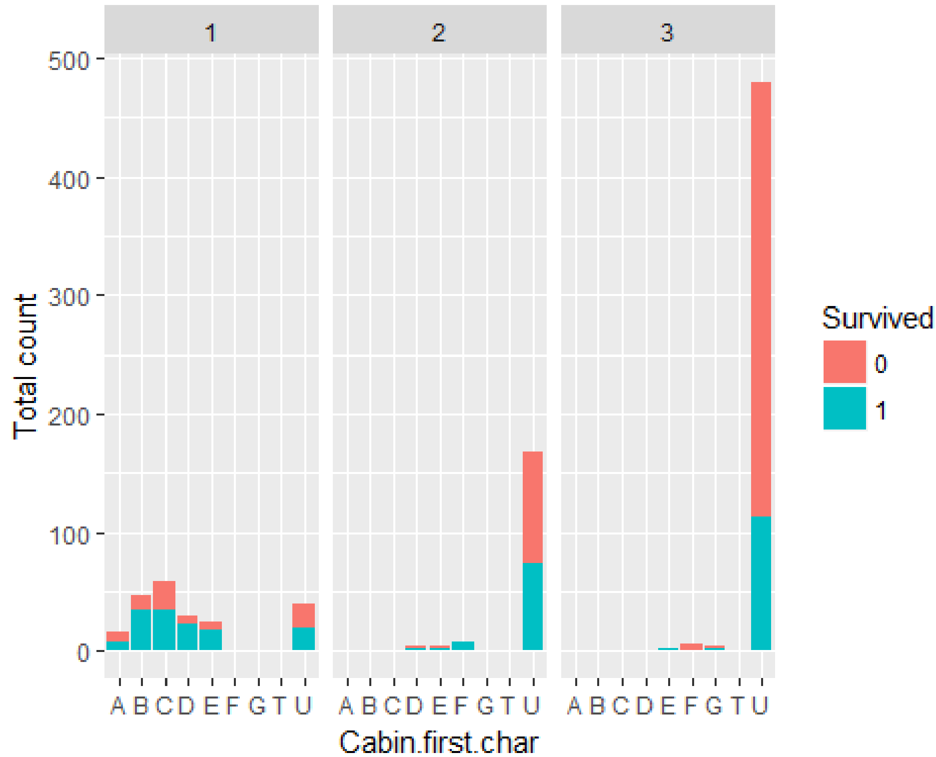



```
# Let's do something with cabin variable now
str(data.combined$Cabin)

##  Factor w/ 187 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...

# Clearly it's not a factor
data.combined$Cabin = as.character(data.combined$Cabin)
data.combined$Cabin[1:100]

##   [1] ""            "C85"         ""            "C123"        ""
##   [6] ""            "E46"         ""            ""            ""
##  [11] "G6"          "C103"        ""            ""            ""
##  [16] ""            ""            ""            ""            ""
##  [21] ""            "D56"         ""            "A6"          ""
##  [26] ""            ""            "C23 C25 C27" ""            ""
##  [31] ""            "B78"         ""            ""            ""
##  [36] ""            ""            ""            ""            ""
##  [41] ""            ""            ""            ""            ""
##  [46] ""            ""            ""            ""            ""
##  [51] ""            ""            "D33"         ""            "B30"
##  [56] "C52"         ""            ""            ""            ""
##  [61] ""            "B28"         "C83"         ""            ""
##  [66] ""            "F33"         ""            ""            ""
##  [71] ""            ""            ""            ""            ""
##  [76] "F G73"       ""            ""            ""            ""
##  [81] ""            ""            ""            ""            ""
##  [86] ""            ""            ""            "C23 C25 C27" ""
```

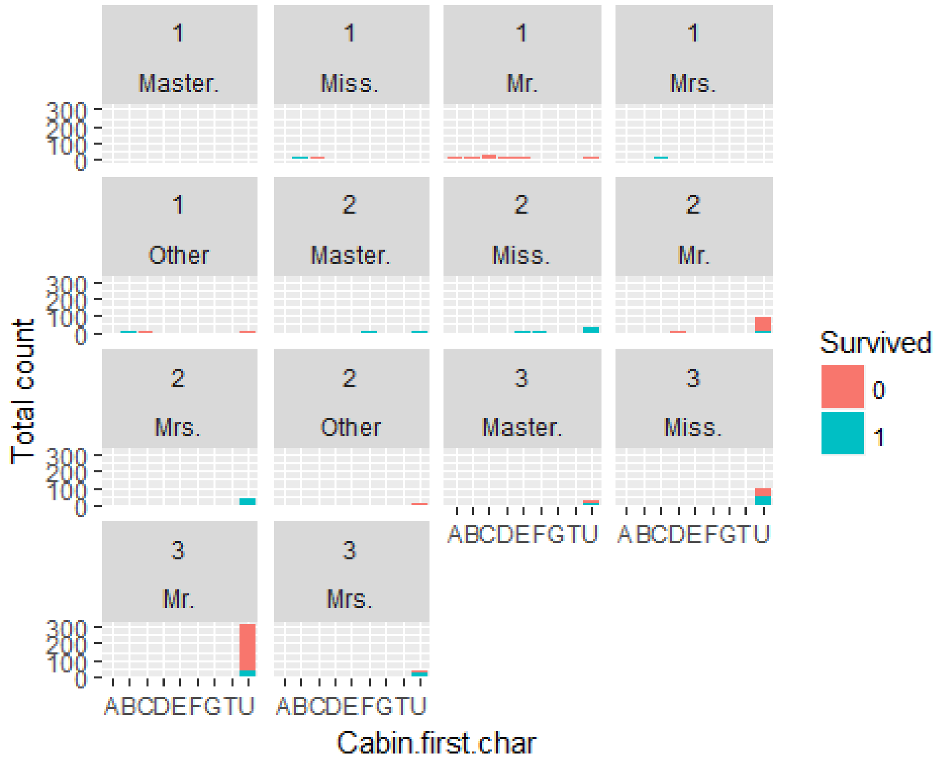
```r
##  [91] ""             ""             "E31"          ""             ""
##  [96] ""             "A5"           "D10 D12"      ""             ""

# Replace the missing cabins with U
data.combined[which(data.combined$Cabin == ""),"Cabin"] = "U"
data.combined$Cabin[1:100]

##   [1] "U"            "C85"          "U"            "C123"         "U"
##   [6] "U"            "E46"          "U"            "U"            "U"
##  [11] "G6"           "C103"         "U"            "U"            "U"
##  [16] "U"            "U"            "U"            "U"            "U"
##  [21] "U"            "D56"          "U"            "A6"           "U"
##  [26] "U"            "U"            "C23 C25 C27" "U"            "U"
##  [31] "U"            "B78"          "U"            "U"            "U"
##  [36] "U"            "U"            "U"            "U"            "U"
##  [41] "U"            "U"            "U"            "U"            "U"
##  [46] "U"            "U"            "U"            "U"            "U"
##  [51] "U"            "U"            "D33"          "U"            "B30"
##  [56] "C52"          "U"            "U"            "U"            "U"
##  [61] "U"            "B28"          "C83"          "U"            "U"
##  [66] "U"            "F33"          "U"            "U"            "U"
##  [71] "U"            "U"            "U"            "U"            "U"
##  [76] "F G73"        "U"            "U"            "U"            "U"
##  [81] "U"            "U"            "U"            "U"            "U"
##  [86] "U"            "U"            "U"            "C23 C25 C27" "U"
##  [91] "U"            "U"            "E31"          "U"            "U"
##  [96] "U"            "A5"           "D10 D12"      "U"            "U"

# Take a look at just first character or letter
cabin.first.char = as.factor(substr(data.combined$Cabin,1,1))
str(cabin.first.char)

##  Factor w/ 9 levels "A","B","C","D",..: 9 3 9 3 9 9 5 9 9 9 ...

levels(cabin.first.char)

## [1] "A" "B" "C" "D" "E" "F" "G" "T" "U"

# Adding it to combined data set and then we go on to plot it to see if
there's any predictive power in it or not
data.combined$cabin.first.char = cabin.first.char

par(mfrow = c(3,3))

# Data Visualization again!
ggplot(data.combined[1:891,], aes(x = cabin.first.char, fill = Survived)) +
  geom_bar() +
  xlab("Cabin.first.char") +
  ylab("Total count")
```



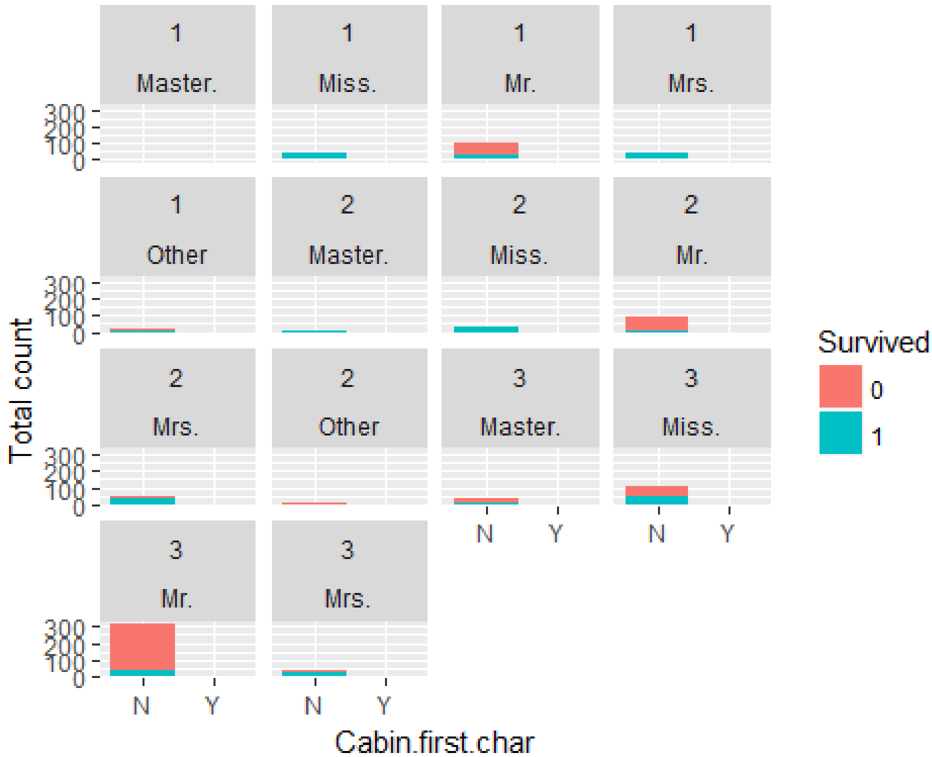

```
#Let's drill in a bit more
ggplot(data.combined[1:891,], aes(x = cabin.first.char, fill = Survived)) +
  geom_bar() +
  facet_wrap(~Pclass) +
  xlab("Cabin.first.char") +
  ylab("Total count")
```



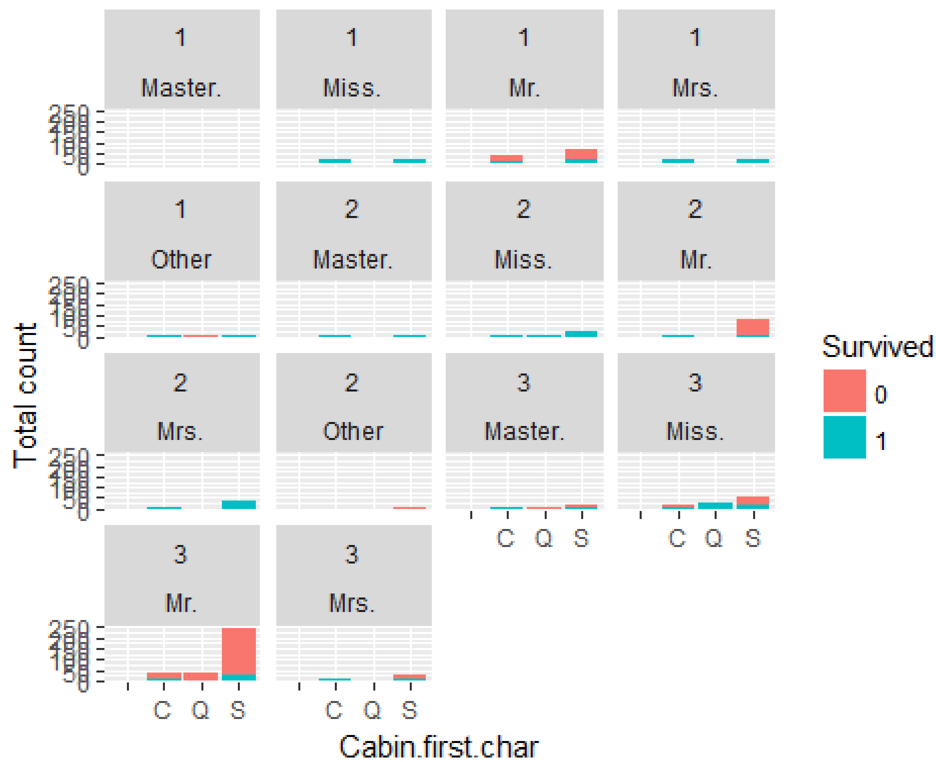

```
#Pclass + title
  ggplot(data.combined[1:891,], aes(x = cabin.first.char, fill = Survived)) +
    geom_bar() +
    facet_wrap(~Pclass + title) +
  xlab("Cabin.first.char") +
    ylab("Total count")
```




```
#What about folks with multiple cabins?
data.combined$cabin.multiple =
as.factor(ifelse(str_detect(data.combined$Cabin," "), "Y", "N"))

#Onto the ggplot thing, ofcourse.
ggplot(data.combined[1:891,], aes(x = cabin.multiple, fill = Survived)) +
  geom_bar() +
  facet_wrap(~Pclass + title)+
xlab("Cabin.first.char") +
  ylab("Total count")
```




```
#Not particularly interesting. We shall come to it later on, maybe.

#Let's have a look at the last variable embarked
str(data.combined$Embarked)

##  Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...

summary(data.combined$Embarked)

##        C   Q   S
##   2 270 123 914

levels(data.combined$Embarked)

## [1] ""  "C" "Q" "S"

#Some plotting again
ggplot(data.combined[1:891,], aes(x = Embarked, fill = Survived)) +
  geom_bar() +
  facet_wrap(~Pclass + title) +
xlab("Cabin.first.char") +
  ylab("Total count")
```

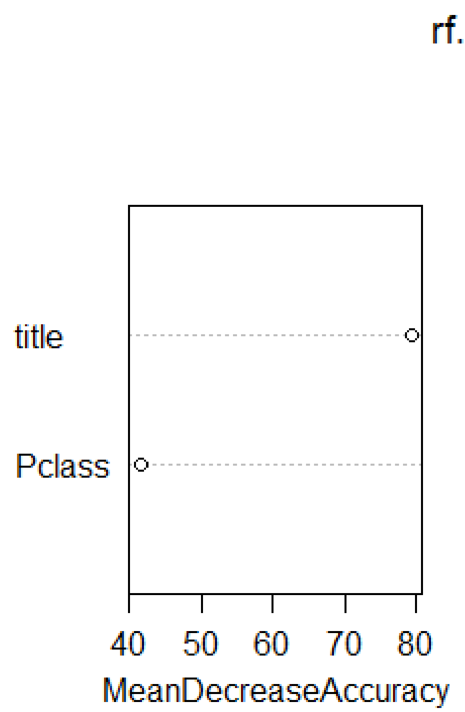


Exploratory Data Analysis

```
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.4.2

## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

#Let's train our first random forest model using just two predictor variables
Pclass and title

rf.train.1 = data.combined[1:891, c("Pclass", "title")]
rf.label = as.factor(train$Survived)

set.seed(1234)
rf.1 = randomForest(x = rf.train.1, y = rf.label, importance = T, ntree =
1000)
rf.1
```

```
##
## Call:
##  randomForest(x = rf.train.1, y = rf.label, ntree = 1000, importance = T)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 20.76%
## Confusion matrix:
##     0   1 class.error
## 0 538  11  0.02003643
## 1 174 168  0.50877193
```

```
varImpPlot(rf.1)
```

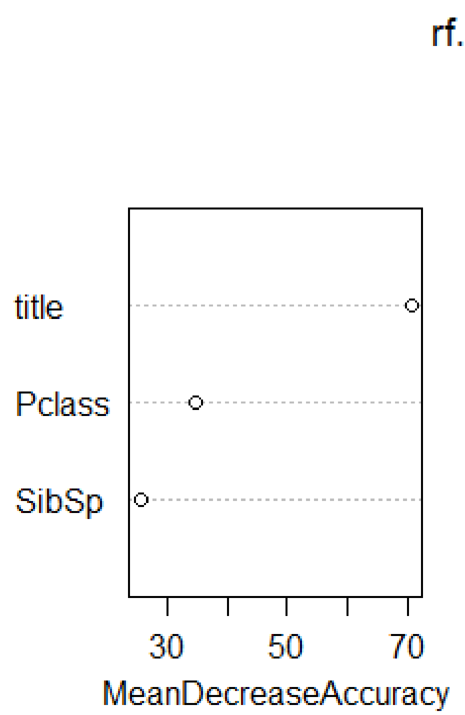



```
#Train a random forest model using Pclass, title and sibsp

rf.train.2 = data.combined[1:891, c("Pclass", "title", "SibSp")]

set.seed(1234)
rf.2 = randomForest(x = rf.train.2, y = rf.label, importance = T, ntree =
1000)
rf.2
```

```
##
## Call:
##  randomForest(x = rf.train.2, y = rf.label, ntree = 1000, importance = T)
```

```
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##          OOB estimate of  error rate: 19.75%
## Confusion matrix:
##     0   1 class.error
## 0 487  62   0.1129326
## 1 114 228   0.3333333
```

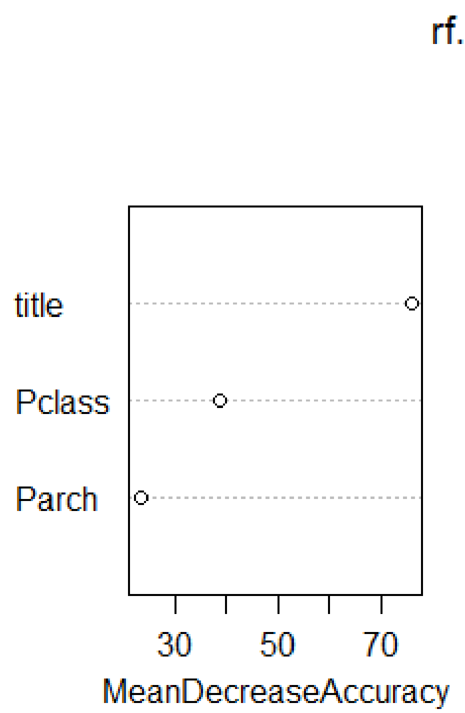
```
varImpPlot(rf.2)
```




```
#Train a random forest model using Pclass, title and Parch

rf.train.3 = data.combined[1:891, c("Pclass", "title", "Parch")]

set.seed(1234)
rf.3 = randomForest(x = rf.train.3, y = rf.label, importance = T, ntree = 
1000)
rf.3
```

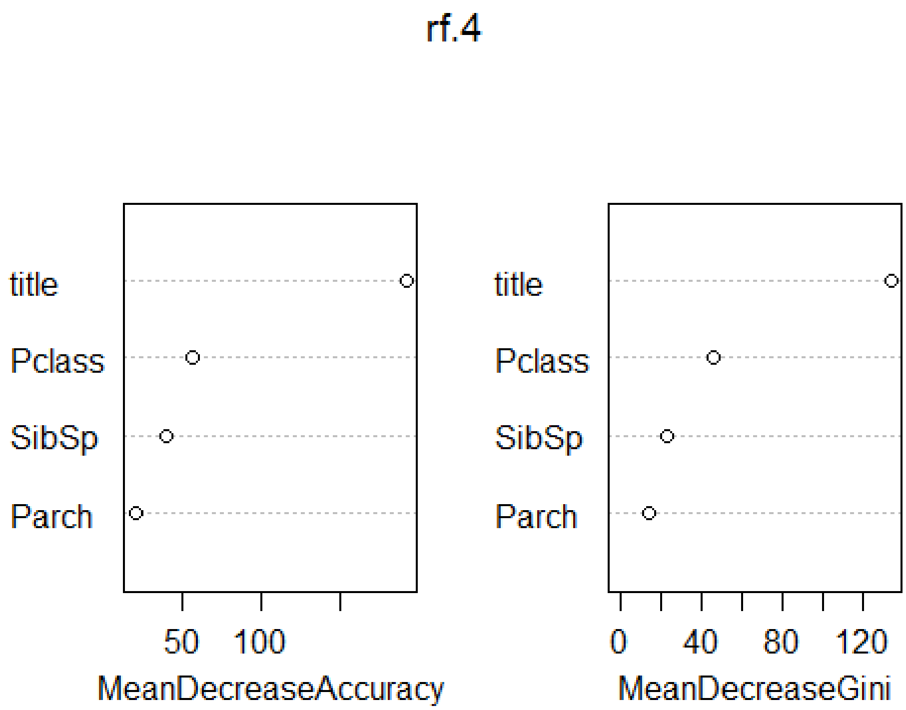
```
##
## Call:
##  randomForest(x = rf.train.3, y = rf.label, ntree = 1000, importance = T)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
```

```
##
##         OOB estimate of  error rate: 19.98%
## Confusion matrix:
##     0   1 class.error
## 0 495  54  0.09836066
## 1 124 218  0.36257310

varImpPlot(rf.3)
```



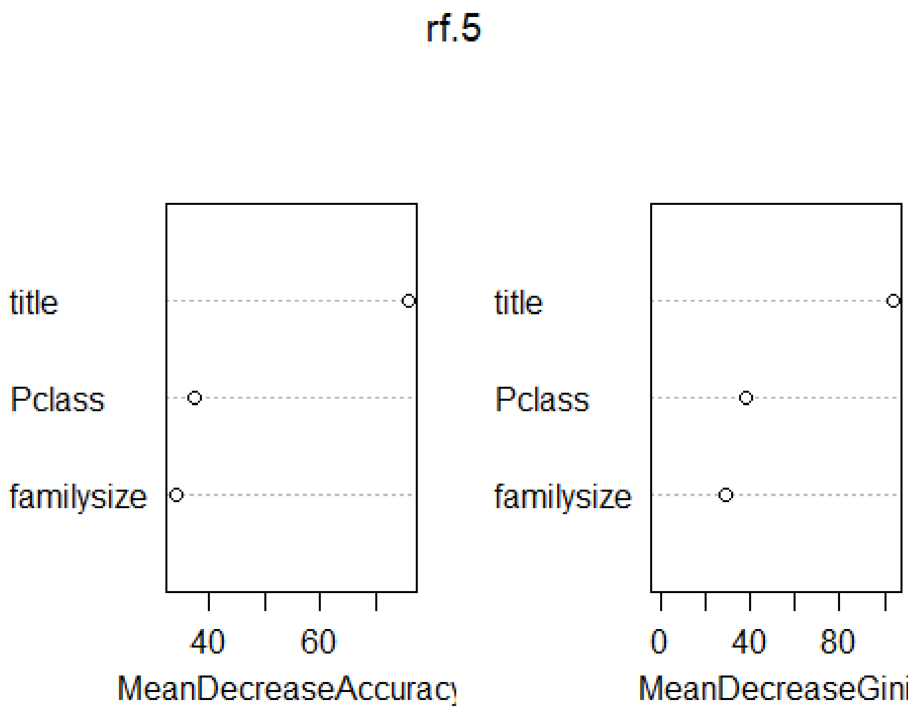

```
#Train a random forest model using Pclass, title, SibSp and Parch

rf.train.4 = data.combined[1:891, c("Pclass", "title", "Parch", "SibSp")]

set.seed(1234)
rf.4 = randomForest(x = rf.train.4, y = rf.label, importance = T, ntree =
1000)
rf.4

##
## Call:
##  randomForest(x = rf.train.4, y = rf.label, ntree = 1000, importance = T)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 18.63%
## Confusion matrix:
```

```
##     0   1 class.error
## 0 488  61   0.1111111
## 1 105 237   0.3070175

varImpPlot(rf.4)
```

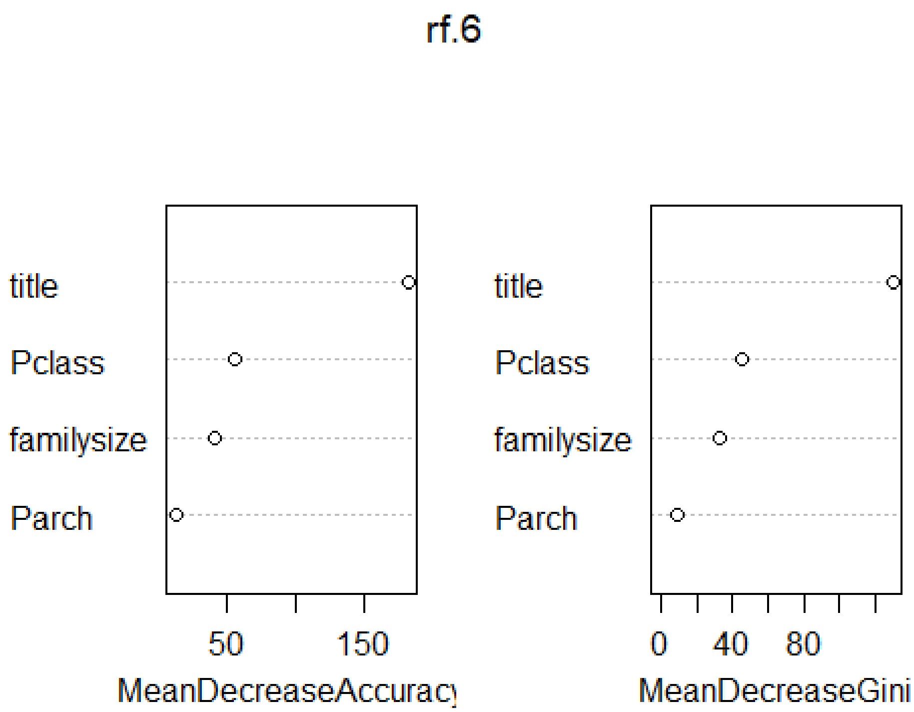



```
#Train a random forest model using Pclass, title and familysize

rf.train.5 = data.combined[1:891, c("Pclass", "title", "familysize")]

set.seed(1234)
rf.5 = randomForest(x = rf.train.5, y = rf.label, importance = T, ntree =
1000)
rf.5

##
## Call:
##  randomForest(x = rf.train.5, y = rf.label, ntree = 1000, importance = T)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 18.41%
## Confusion matrix:
##     0   1 class.error
## 0 485  64   0.1165756
## 1 100 242   0.2923977
```

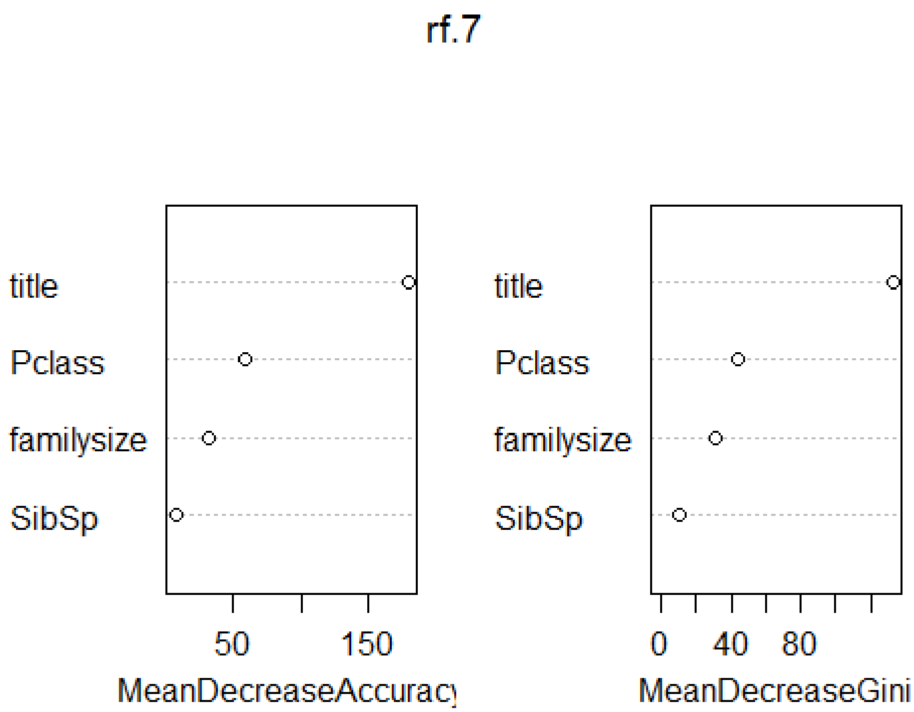
```r
varImpPlot(rf.5)
```




```r
#Train a random forest model using Pclass, title, familysize and Parch

rf.train.6 = data.combined[1:891, c("Pclass", "title", "Parch",
"familysize")]

set.seed(1234)
rf.6 = randomForest(x = rf.train.6, y = rf.label, importance = T, ntree =
1000)
rf.6

##
## Call:
##  randomForest(x = rf.train.6, y = rf.label, ntree = 1000, importance = T)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 18.97%
## Confusion matrix:
##     0   1 class.error
## 0 486  63   0.1147541
## 1 106 236   0.3099415

varImpPlot(rf.6)
```




```
#Train a random forest model using Pclass, title, SibSp and Familysize

rf.train.7 = data.combined[1:891, c("Pclass", "title", "SibSp",
"familysize")]

set.seed(1234)
rf.7 = randomForest(x = rf.train.7, y = rf.label, importance = T, ntree =
1000)
rf.7

##
## Call:
##  randomForest(x = rf.train.7, y = rf.label, ntree = 1000, importance = T)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 18.74%
## Confusion matrix:
##     0   1 class.error
## 0 486  63   0.1147541
## 1 104 238   0.3040936

varImpPlot(rf.7)
```




```
#As it is very evident from the above running models, the model with just
title, pclass and familysize gives the highest accuracy or lowest OOB rate
```

Cross Validation

```
#Let's try to submit these predictions to Kaggle first and see how we're
doing
test.submit.df = data.combined[892:1309, c("Pclass", "familysize", "title")]

#This is how you should predict
rf.5.preds = predict(rf.5, test.submit.df)
table(rf.5.preds)

## rf.5.preds
##   0   1
## 258 160

#Write out a CSV file for the submission to Kaggle
submit.df = data.frame(PassengerId = 892:1309, Survived = rf.5.preds)

write.csv(submit.df, file = "RF1.csv", row.names = F)

#Now, as we can see from the Kaggle, our score turns out be 0.79426 but the
OOB estimates predicted it to be 0.8159
#Let's dig deep into the concept of cross-validation

library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4

## Loading required package: lattice

library(doSNOW)

## Warning: package 'doSNOW' was built under R version 3.4.4

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.4.3

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 3.4.3

## Loading required package: snow

## Warning: package 'snow' was built under R version 3.4.4

#We're gonna be doing something called stratified cross-validation.
set.seed(2348)
cv.10.folds = createMultiFolds(y=rf.label, k=10, times = 10)

#Check stratification
table(rf.label)

## rf.label
##   0   1
## 549 342

342/549

## [1] 0.6229508

#Check for any fold now
table(rf.label[cv.10.folds[[33]]])

##
##   0   1
## 494 308

307/494

## [1] 0.6214575

#For stratification, the main property is that the ratio of folks who
perished to the folks who survived should be same in the each folds and the
y(rf.label)

#Now, let's setup traincontrol object per above
ctrl.1 = trainControl(method = "repeatedcv", number = 10, repeats = 10, index
= cv.10.folds)
```

```r
#Set up doSNOW package for multi-core training. This is helpful because we're
gonna be using a lot of trees
cl = makeCluster(6, type = "SOCK")
registerDoSNOW(cl)

#Set seed for reproducibility and train
set.seed(34324)
rf.5.cv.1 = train(x=rf.train.5, y=rf.label, method="rf", tuneLength=3,
ntree=1000, trControl= ctrl.1)

## note: only 2 unique complexity parameters in default grid. Truncating the
grid to 2 .

#Shut down cluster
stopCluster(cl)

#Check out results
rf.5.cv.1

## Random Forest
##
## 891 samples
##   3 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 801, 802, 802, 803, 802, 801, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8128058  0.594528
##   3     0.8093388  0.585973
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

#Let's also try 5-fold to see if there is improvment in accuracy
set.seed(5983)
cv.5.folds = createMultiFolds(y=rf.label, k=5, times = 10)

ctrl.2 = trainControl(method = "repeatedcv", number = 5, repeats = 10, index
= cv.5.folds)

#Set up doSNOW package for multi-core training. This is helpful because we're
gonna be using a lot of trees
cl = makeCluster(6, type = "SOCK")
registerDoSNOW(cl)

#Set seed for reproducibility and train
```

```r
set.seed(89472)
rf.5.cv.2 = train(x=rf.train.5, y=rf.label, method="rf", tuneLength=3,
ntree=1000, trControl= ctrl.2)

## note: only 2 unique complexity parameters in default grid. Truncating the
grid to 2 .

#Shut down cluster
stopCluster(cl)

#Check out results
rf.5.cv.2

## Random Forest
##
## 891 samples
##   3 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
## Summary of sample sizes: 713, 713, 713, 713, 712, 713, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8133608  0.5974520
##   3     0.8093159  0.5881247
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

#3-fold
set.seed(5986)
cv.3.folds = createMultiFolds(y=rf.label, k=3, times = 10)

ctrl.3 = trainControl(method = "repeatedcv", number = 3, repeats = 10, index
= cv.3.folds)

#Set up doSNOW package for multi-core training. This is helpful because we're
gonna be using a lot of trees
cl = makeCluster(6, type = "SOCK")
registerDoSNOW(cl)

#Set seed for reproducibility and train
set.seed(89465)
rf.5.cv.3 = train(x=rf.train.5, y=rf.label, method="rf", tuneLength=3,
ntree=1000, trControl= ctrl.3)

## note: only 2 unique complexity parameters in default grid. Truncating the
grid to 2 .
```

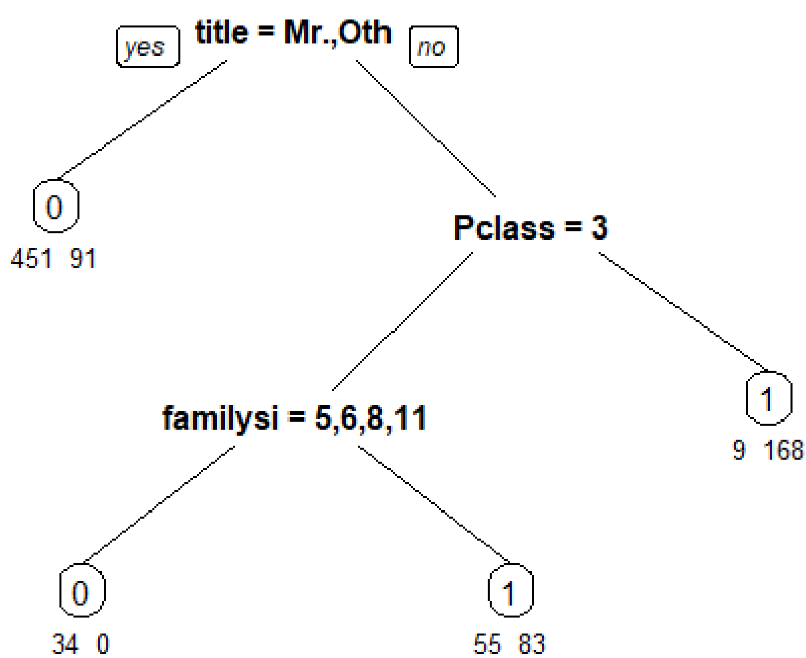
```
#Shut down cluster
stopCluster(cl)

#Check out results
rf.5.cv.3

## Random Forest
##
## 891 samples
##   3 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 10 times)
## Summary of sample sizes: 594, 594, 594, 594, 594, 594, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8143659  0.5956886
##   3     0.8104377  0.5857806
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

#Let's see where we might have gone wrong. Let's build a single decision tree
to check what exactly is happening on the inside
#Random forests are ofcourse way better than decision trees but when it comes
to easily understand the whole picture, deciion trees
#are way better than random forests

library(rpart)
#install.packages("rpart.plot")
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.4.4

#Create utility function
rpart.cv = function(seed, training, labels, ctrl) {
  cl = makeCluster(6, type = "SOCK")
  registerDoSNOW(cl)

  set.seed(seed)
  #Leverage formula interface for training
  rpart.cv = train(x=training, y=labels, method="rpart", tuneLength=30,
trControl = ctrl)

  #shutdown cluster
  stopCluster(cl)

  return(rpart.cv)
```

```
}

#Grab features
features = c("Pclass", "title", "familysize")
rpart.train.1 = data.combined[1:891, features]

#Run CV and check out results
rpart.1.cv.1 = rpart.cv(94622, rpart.train.1, rf.label, ctrl.3)
rpart.1.cv.1

## CART
##
## 891 samples
##   3 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 10 times)
## Summary of sample sizes: 594, 594, 594, 594, 594, 594, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.8116723  0.5917680
##   0.01542650  0.8205387  0.6145781
##   0.03085299  0.8205387  0.6145781
##   0.04627949  0.8129068  0.5999912
##   0.06170599  0.7888889  0.5527336
##   0.07713249  0.7888889  0.5527336
##   0.09255898  0.7867565  0.5511186
##   0.10798548  0.7848485  0.5477607
##   0.12341198  0.7842873  0.5469939
##   0.13883848  0.7842873  0.5469939
##   0.15426497  0.7842873  0.5469939
##   0.16969147  0.7842873  0.5469939
##   0.18511797  0.7842873  0.5469939
##   0.20054446  0.7842873  0.5469939
##   0.21597096  0.7842873  0.5469939
##   0.23139746  0.7842873  0.5469939
##   0.24682396  0.7842873  0.5469939
##   0.26225045  0.7842873  0.5469939
##   0.27767695  0.7842873  0.5469939
##   0.29310345  0.7842873  0.5469939
##   0.30852995  0.7842873  0.5469939
##   0.32395644  0.7842873  0.5469939
##   0.33938294  0.7842873  0.5469939
##   0.35480944  0.7842873  0.5469939
##   0.37023593  0.7842873  0.5469939
##   0.38566243  0.7842873  0.5469939
##   0.40108893  0.7777778  0.5267317
##   0.41651543  0.7582492  0.4661039
```

```
##   0.43194192  0.7202020  0.3472667
##   0.44736842  0.6868687  0.2382834
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03085299.
```

```
#Plot
prp(rpart.1.cv.1$finalModel, type = 0, extra = 1, under = T)
```

```
## Warning: Bad 'data' field in model 'call' field.
##          To make this warning go away:
##              Call prp with roundint=FALSE,
##              or rebuild the rpart model with model=TRUE.
```




```
#The plot brings out some interesting lines of investigation. Namely:
#     1 - Titles of Mr. and Others are predicted to perish at an overall
accuracy of 83.2%
#     2 - Titles of Master, Miss and Mrs. in 1st and 2nd class are predicted
to survive at
#          an overall accuracy of 94.9%
#     3 - Titles of Master, Miss & Mrs in class 3 and having a family size of
5,6,8,11
#          are predicted to perish at an overall accuracy of 100%
#     4 - Titles of Master, Miss & Mrs in class 3 and having a family size of
5,6,8,11
#          are predicted to survive at an overall accuracy of 59.6%
```

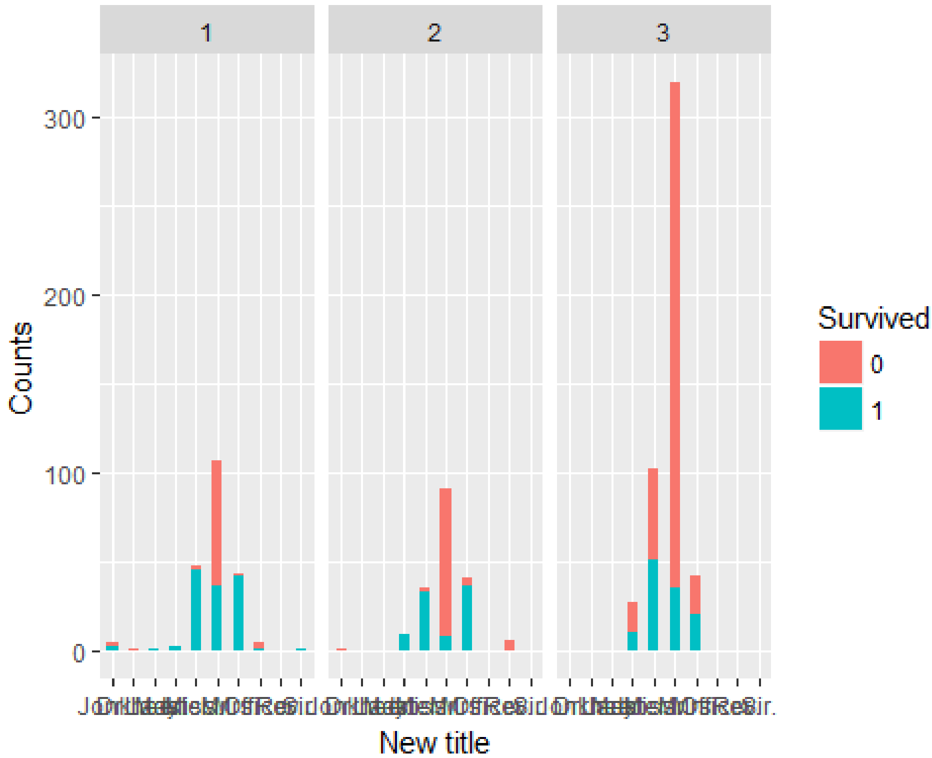
```r
#Both rpart and ef confirm that title is important. Let's investigate
further:
#Also, we're stressing more on the 1st point here that the title Mr and other
just seem blunt.
#Let's move ahead and investigate it further
table(data.combined$title)

##
## Master.   Miss.     Mr.    Mrs.   Other
##      61     260     758     199      31

#Parse out last name and title
data.combined$Name[1:5]

## [1] Braund, Mr. Owen Harris
## [2] Cumings, Mrs. John Bradley (Florence Briggs Thayer)
## [3] Heikkinen, Miss. Laina
## [4] Futrelle, Mrs. Jacques Heath (Lily May Peel)
## [5] Allen, Mr. William Henry
## 1307 Levels: Abbing, Mr. Anthony ... Zakarian, Mr. Ortin

name.splits = str_split(data.combined$Name, ",")
name.splits[1]

## [[1]]
## [1] "Braund"           " Mr. Owen Harris"

last.names = sapply(name.splits, "[", 1)
last.names[1:10]

##  [1] "Braund"    "Cumings"   "Heikkinen" "Futrelle"  "Allen"
##  [6] "Moran"     "McCarthy"  "Palsson"   "Johnson"   "Nasser"

#Add last names to the data.combined in case we might find it useful later
data.combined$last.name = last.names

#Now for titles
name.splits = str_split(sapply(name.splits,"[",2)," ")
titles = sapply(name.splits,"[",2)
unique(titles)

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

#What's up with the title of "the"?
which(titles == "the")

## [1] 760

data.combined[760,]
```

```
##     PassengerId Survived Pclass
## 760         760        1      1
##                                                            Name    Sex Age
## 760 Rothes, the Countess. of (Lucy Noel Martha Dyer-Edwards) female  33
##     SibSp Parch Ticket Fare Cabin Embarked title familysize
## 760     0     0 110152 86.5   B77        S Other          1
##     ticket.first.char cabin.first.char cabin.multiple last.name
## 760                 1                B              N    Rothes
```

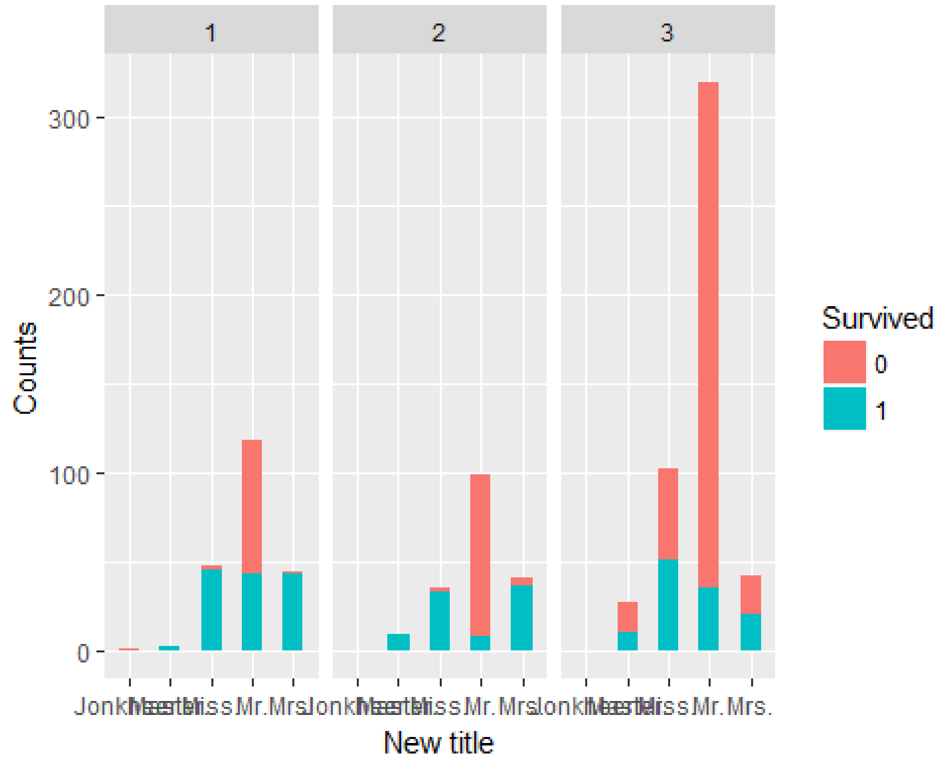
```
#Re-map titles to be more exact
titles[titles %in% c("the", "Dona.")] = "Lady."
titles[titles %in% c("Ms.", "Mlle.")] = "Miss."
titles[titles == "Mme."] = "Mrs."
titles[titles %in% c("Jonkheer", "Don.")] = "Sir."
titles[titles %in% c("Col.", "Capt.", "Major.")] = "Officer"
table(titles)
```

```
## titles
##       Dr. Jonkheer.     Lady.   Master.     Miss.       Mr.      Mrs.
##         8         1         3        61       264       757       198
##   Officer      Rev.      Sir.
##         7         8         2
```

```
#Now add this to the dataframe again
data.combined$new.title = as.factor(titles)

#Let's again use this for data visualization
ggplot(data.combined[1:891,], aes(x=new.title, fill=Survived)) +
  geom_bar(width = 0.5) +
  facet_wrap(~Pclass) +
  xlab("New title") +
  ylab("Counts") +
  labs(fill = "Survived")
```




```
#Collapse titles based on visual analysis
indexes = which(data.combined$new.title == "Lady.")
data.combined$new.title[indexes] = "Mrs."

indexes = which(data.combined$new.title == "Dr." |
                  data.combined$new.title == "Rev." |
                  data.combined$new.title == "Sir." |
                  data.combined$new.title == "Officer")
data.combined$new.title[indexes] = "Mr."

#Visualize again
ggplot(data.combined[1:891,], aes(x=new.title, fill=Survived)) +
  geom_bar(width = 0.5) +
  facet_wrap(~Pclass) +
  xlab("New title") +
  ylab("Counts") +
  labs(fill = "Survived")
```

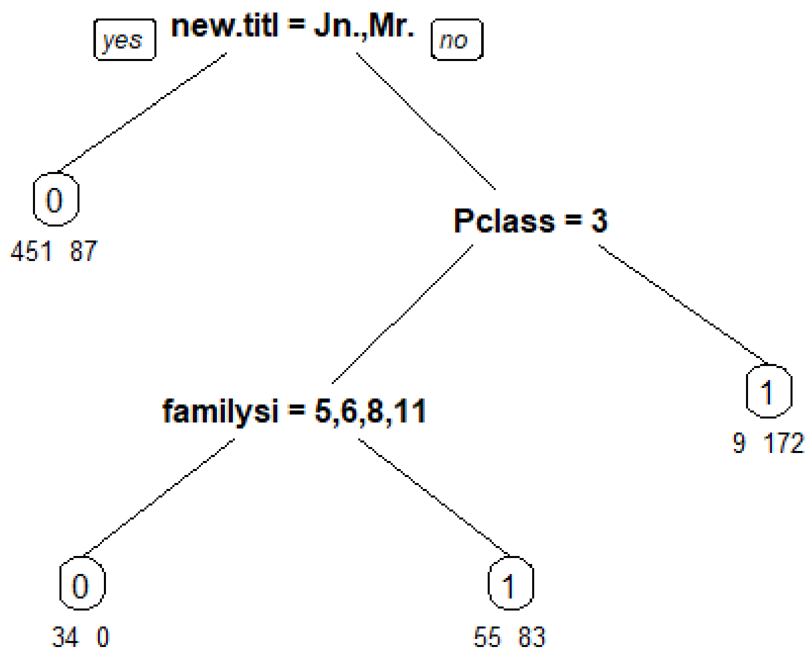



```
#Grab features
features = c("Pclass", "new.title", "familysize")
rpart.train.2 = data.combined[1:891, features]

#Run CV and check out results
rpart.2.cv.1 = rpart.cv(94622, rpart.train.2, rf.label, ctrl.3)
rpart.2.cv.1

## CART
##
## 891 samples
##   3 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 10 times)
## Summary of sample sizes: 594, 594, 594, 594, 594, 594, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.8191919  0.6070484
##   0.01582980  0.8287318  0.6319625
##   0.03165961  0.8287318  0.6319625
##   0.04748941  0.8210999  0.6173617
##   0.06331922  0.7959596  0.5677607
##   0.07914902  0.7973064  0.5721536
##   0.09497883  0.7938272  0.5660315
```

```
##   0.11080863  0.7929293  0.5643804
##   0.12663844  0.7923681  0.5636067
##   0.14246824  0.7923681  0.5636067
##   0.15829804  0.7923681  0.5636067
##   0.17412785  0.7923681  0.5636067
##   0.18995765  0.7923681  0.5636067
##   0.20578746  0.7923681  0.5636067
##   0.22161726  0.7923681  0.5636067
##   0.23744707  0.7923681  0.5636067
##   0.25327687  0.7923681  0.5636067
##   0.26910667  0.7923681  0.5636067
##   0.28493648  0.7923681  0.5636067
##   0.30076628  0.7923681  0.5636067
##   0.31659609  0.7923681  0.5636067
##   0.33242589  0.7923681  0.5636067
##   0.34825570  0.7923681  0.5636067
##   0.36408550  0.7923681  0.5636067
##   0.37991531  0.7923681  0.5636067
##   0.39574511  0.7923681  0.5636067
##   0.41157491  0.7850730  0.5419190
##   0.42740472  0.7645342  0.4793351
##   0.44323452  0.7193042  0.3386920
##   0.45906433  0.7012346  0.2816186
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03165961.
```

```r
#Plot
prp(rpart.2.cv.1$finalModel, type = 0, extra = 1, under = T)
```

```
## Warning: Bad 'data' field in model 'call' field.
##          To make this warning go away:
##              Call prp with roundint=FALSE,
##              or rebuild the rpart model with model=TRUE.
```




```
#Dive in on 1st class Mr.
indexes.first.mr = which(data.combined$new.title == "Mr." &
data.combined$Pclass == "1")
first.mr.df = data.combined[indexes.first.mr,]
summary(first.mr.df)

##   PassengerId     Survived   Pclass
##  Min.   :   7.0   0   :76    1:174
##  1st Qu.: 371.8   1   :43    2:  0
##  Median : 647.0   None:55    3:  0
##  Mean   : 656.8
##  3rd Qu.: 966.5
##  Max.   :1299.0
##
##                                     Name          Sex           Age
##  Anderson, Mr. Harry                  :  1   female:  1   Min.   :17.00
##  Andrews, Mr. Thomas Jr               :  1   male  :173   1st Qu.:31.00
##  Artagaveytia, Mr. Ramon              :  1                Median :42.00
##  Barkworth, Mr. Algernon Henry Wilson:  1                Mean   :42.27
##  Baumann, Mr. John D                  :  1                3rd Qu.:50.75
##  Baxter, Mr. Quigg Edmond             :  1                Max.   :80.00
##  (Other)                              :168                NA's   :28
##  SibSp        Parch         Ticket                Fare
##  0:121   0      :145   Length:174         Min.   :  0.00
##  1: 50   1      : 21   Class :character   1st Qu.: 27.72
##  2:  2   2      :  6   Mode  :character   Median : 46.30
##  3:  1   3      :  1                      Mean   : 67.73
```

```
##  4:  0   4      :  1                      3rd Qu.: 78.46  
##  5:  0   5      :  0                      Max.   :512.33  
##  8:  0   (Other):  0                                      
##     Cabin              Embarked     title        familysize  ticket.first.char
##  Length:174          :  0    Master.:  0   1      :108   1      :109   
##  Class :character   C: 68    Miss.  :  0   2      : 45   P      : 45   
##  Mode  :character   Q:  1    Mr.    :160   3      : 16   3      :  8   
##                     S:105    Mrs.   :  0   4      :  2   6      :  4   
##                              Other  : 14   6      :  2   2      :  3   
##                                            5      :  1   F      :  2   
##                                            (Other):  0   (Other):  3   
##  cabin.first.char cabin.multiple  last.name             new.title  
##  C      :47       N:161          Length:174         Mr.      :174  
##  U      :43       Y: 13          Class :character   Dr.      :  0  
##  B      :27                      Mode  :character   Jonkheer.:  0  
##  D      :20                                         Lady.    :  0  
##  E      :19                                         Master.  :  0  
##  A      :17                                         Miss.    :  0  
##  (Other): 1                                         (Other)  :  0
```

```
#1 female?
first.mr.df[first.mr.df$Sex == "female",]
```

```
##     PassengerId Survived Pclass                          Name    Sex Age
## 797         797        1      1 Leader, Dr. Alice (Farnham) female  49
##     SibSp Parch Ticket    Fare Cabin Embarked title familysize
## 797     0     0  17465 25.9292   D17        S Other          1
##     ticket.first.char cabin.first.char cabin.multiple last.name new.title
## 797                 1                D              N    Leader       Mr.
```

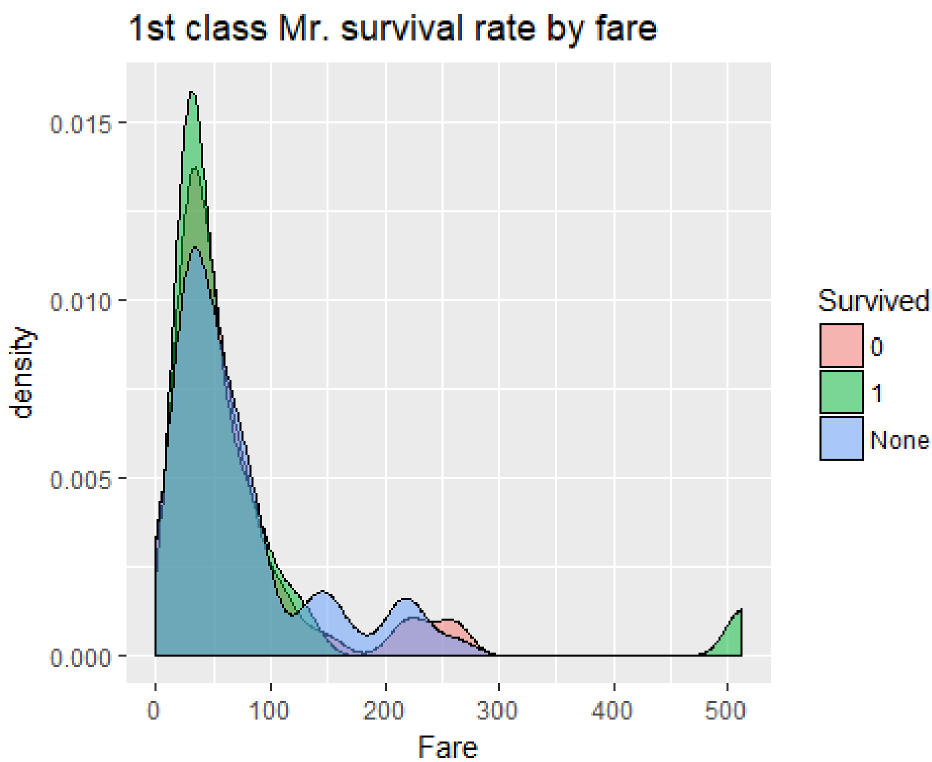
```
#Here, we can see that she is Dr. and has been classified as a "Mr."

#Let's update new.title feature
indexes = which(data.combined$Sex == "female" & data.combined$new.title == 
"Mr.")
data.combined$new.title[indexes] = "Mrs."

#Any other gender slip-ups?
length(which(data.combined$Sex == "female" & (data.combined$new.title == 
"Master." | data.combined$new.title == "Mr.")))
```

```
## [1] 0
```

```
#Refresh dataframe
indexes.first.mr = which(data.combined$new.title == "Mr." & 
data.combined$Pclass == "1")
first.mr.df = data.combined[indexes.first.mr,]

#Let's look at surviving 1st class "Mr."
summary(first.mr.df[first.mr.df$Survived == "1",])
```

```
##   PassengerId     Survived  Pclass
##  Min.   : 24.0   0   : 0   1:42
##  1st Qu.:435.2   1   :42   2: 0
##  Median :594.0   None: 0   3: 0
##  Mean   :528.5
##  3rd Qu.:681.5
##  Max.   :890.0
##
##                                          Name         Sex
##  Anderson, Mr. Harry                       : 1   female: 0
##  Barkworth, Mr. Algernon Henry Wilson      : 1   male  :42
##  Beckwith, Mr. Richard Leonard             : 1
##  Behr, Mr. Karl Howell                     : 1
##  Bishop, Mr. Dickinson H                   : 1
##  Bjornstrom-Steffansson, Mr. Mauritz Hakan: 1
##  (Other)                                   :36
##       Age          SibSp      Parch       Ticket               Fare
##  Min.   :17.00   0:28   0      :36   Length:42          Min.   : 26.29
##  1st Qu.:28.00   1:13   1      : 4   Class :character   1st Qu.: 27.34
##  Median :36.00   2: 1   2      : 2   Mode  :character   Median : 35.50
##  Mean   :38.76   3: 0   3      : 0                      Mean   : 71.55
##  3rd Qu.:48.00   4: 0   4      : 0                      3rd Qu.: 73.39
##  Max.   :80.00   5: 0   5      : 0                      Max.   :512.33
##  NA's   :5       8: 0   (Other): 0
##     Cabin           Embarked     title      familysize ticket.first.char
##  Length:42          : 0     Master.: 0   1      :25   1      :30
##  Class :character   C:17     Miss.  : 0   2      :12   P      :11
##  Mode  :character   Q: 0     Mr.    :38   3      : 4   2      : 1
##                     S:25     Mrs.   : 0   4      : 1   3      : 0
##                              Other  : 4   5      : 0   4      : 0
##                                           6      : 0   5      : 0
##                                           (Other): 0   (Other): 0
##  cabin.first.char cabin.multiple  last.name             new.title
##  C      :10       N:39           Length:42          Mr.      :42
##  E      : 8       Y: 3           Class :character   Dr.      : 0
##  B      : 7                      Mode  :character   Jonkheer.: 0
##  D      : 6                                         Lady.    : 0
##  U      : 6                                         Master.  : 0
##  A      : 5                                         Miss.    : 0
##  (Other): 0                                         (Other)  : 0

View(first.mr.df[first.mr.df$Survived == "1",])

#Take a look at some of the high fares
indexes = which(data.combined$Ticket == "PC 17755" |
                data.combined$Ticket ==  "113760" |
                data.combined$Ticket == "PC 17611")
View(data.combined[indexes,])

#Visualize survival rates for 1st class "Mr." by fare
```

```
ggplot(first.mr.df, aes(x = Fare, fill = Survived)) +
  geom_density(alpha = 0.5) +
  ggtitle("1st class Mr. survival rate by fare")
```

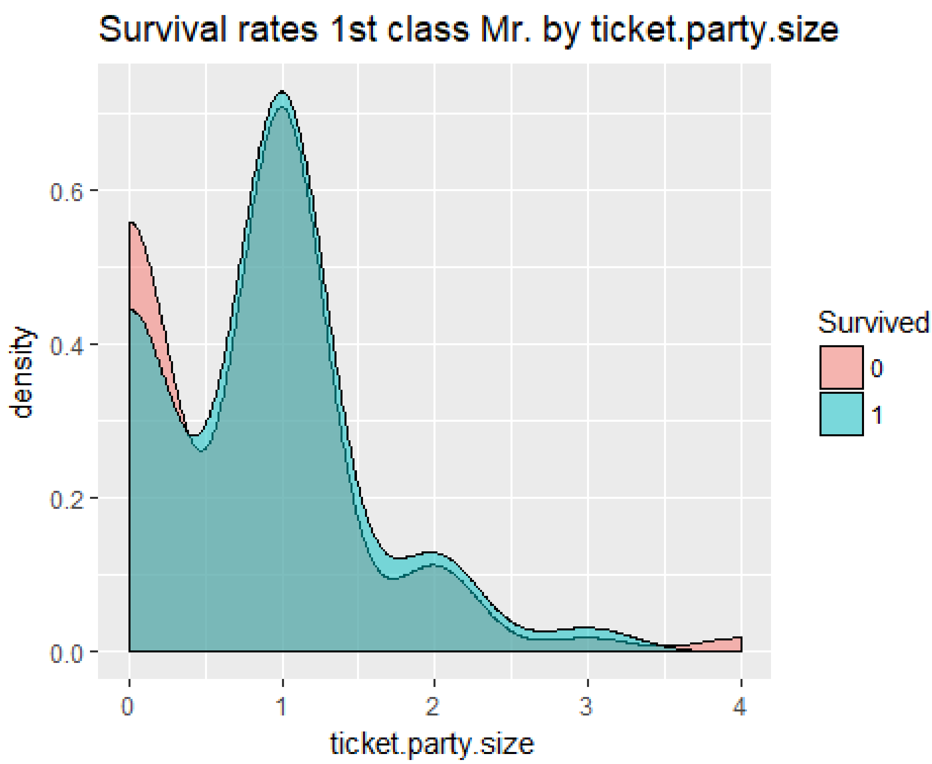



```
#Engineer features based on all the passengers with the same ticket
ticket.party.size = rep(0, nrow(data.combined))
avg.fare = rep(0.0, nrow(data.combined) )
tickets = unique(data.combined$Ticket)

for(i in 1:length(tickets)) {
  current.ticket = tickets[i]
  party.indexes = which(data.combined$Ticket == current.ticket)
  current.avg.fare = data.combined[party.indexes[1], "Fare"] /
length(party.indexes)

  for(k in i:length(party.indexes)){
    ticket.party.size[party.indexes[k]] = length(party.indexes)
    avg.fare[party.indexes[k]] = current.avg.fare
  }
}

data.combined$ticket.party.size = ticket.party.size
data.combined$avg.fare = avg.fare
data.combined$`avg. fare` = NULL

#Refresh 1st class "Mr." dataframe
```

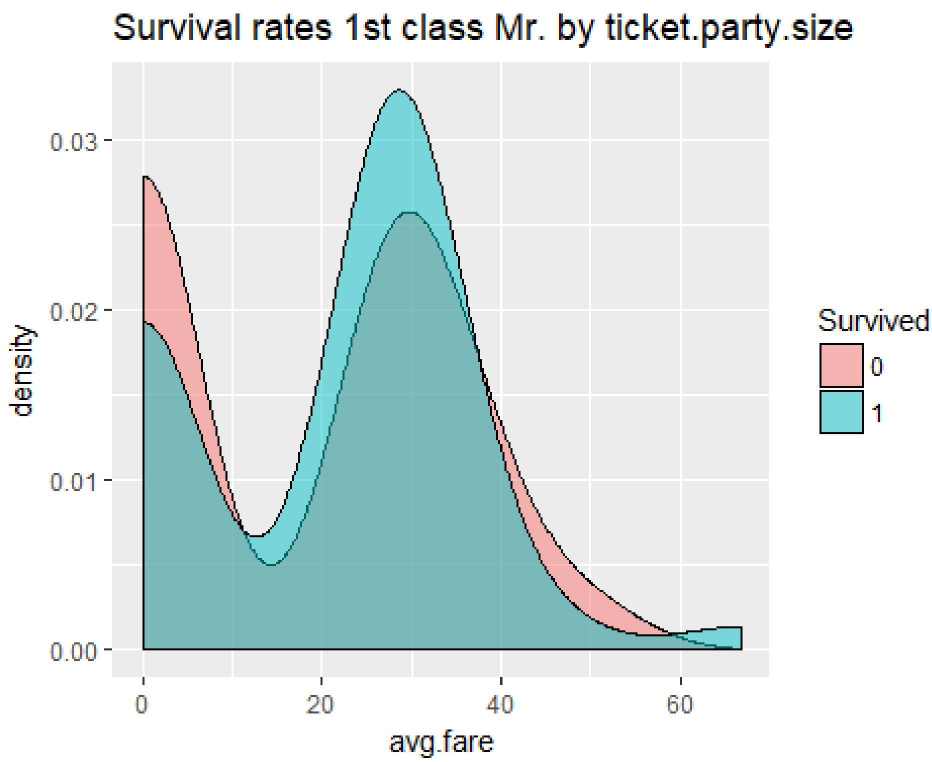
```r
first.mr.df = data.combined[indexes.first.mr,]
summary(first.mr.df)
```

```
##   PassengerId    Survived   Pclass 
##  Min.   :   7   0   :76   1:173  
##  1st Qu.: 371   1   :42   2:  0  
##  Median : 646   None:55   3:  0  
##  Mean   : 656                    
##  3rd Qu.: 967                    
##  Max.   :1299                    
##                                                                   
##                                   Name          Sex           Age       
##  Anderson, Mr. Harry                :  1   female:  0   Min.   :17.00  
##  Andrews, Mr. Thomas Jr             :  1   male  :173   1st Qu.:31.00  
##  Artagaveytia, Mr. Ramon            :  1                Median :42.00  
##  Barkworth, Mr. Algernon Henry Wilson:  1                Mean   :42.22  
##  Baumann, Mr. John D                :  1                3rd Qu.:51.00  
##  Baxter, Mr. Quigg Edmond           :  1                Max.   :80.00  
##  (Other)                            :167                NA's   :28     
##  SibSp       Parch          Ticket               Fare        
##  0:120   0      :144   Length:173         Min.   :  0.00  
##  1: 50   1      : 21   Class :character   1st Qu.: 27.72  
##  2:  2   2      :  6   Mode  :character   Median : 47.10  
##  3:  1   3      :  1                      Mean   : 67.98  
##  4:  0   4      :  1                      3rd Qu.: 78.85  
##  5:  0   5      :  0                      Max.   :512.33  
##  8:  0   (Other):  0                                      
##     Cabin             Embarked     title       familysize  ticket.first.char
##  Length:173          :  0     Master.:  0   1      :107   1      :108      
##  Class :character   C: 68     Miss.  :  0   2      : 45   P      : 45      
##  Mode  :character   Q:  1     Mr.    :160   3      : 16   3      :  8      
##                     S:104     Mrs.   :  0   4      :  2   6      :  4      
##                               Other  : 13   6      :  2   2      :  3      
##                                             5      :  1   F      :  2      
##                                             (Other):  0   (Other):  3      
##  cabin.first.char cabin.multiple  last.name            new.title  
##  C      :47       N:160          Length:173         Mr.      :173  
##  U      :43       Y: 13          Class :character   Dr.      :  0  
##  B      :27                      Mode  :character   Jonkheer.:  0  
##  D      :19                                         Lady.    :  0  
##  E      :19                                         Master.  :  0  
##  A      :17                                         Miss.    :  0  
##  (Other): 1                                         (Other)  :  0  
##  ticket.party.size    avg.fare    
##  Min.   :0.0000    Min.   : 0.00  
##  1st Qu.:0.0000    1st Qu.: 0.00  
##  Median :1.0000    Median :26.55  
##  Mean   :0.9711    Mean   :20.52  
##  3rd Qu.:1.0000    3rd Qu.:30.50  
```

```
##   Max.   :6.0000    Max.   :66.83
##

#Visualize new features
ggplot(first.mr.df[first.mr.df$Survived != "None",], aes(x =
ticket.party.size, fill=Survived)) +
  geom_density(alpha = 0.5) +
  ggtitle("Survival rates 1st class Mr. by ticket.party.size")
```




```
ggplot(first.mr.df[first.mr.df$Survived != "None",], aes(x = avg.fare,
fill=Survived)) +
  geom_density(alpha = 0.5) +
  ggtitle("Survival rates 1st class Mr. by ticket.party.size")
```



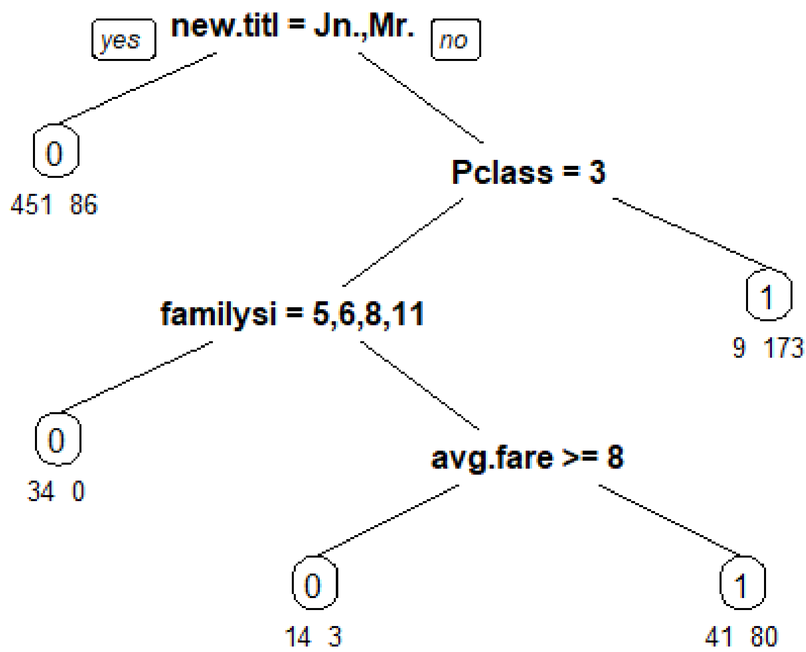

```
#Hypothesis - ticket.party.size is highly correlated with avg. fare
summary(data.combined$avg.fare)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.000   7.775   9.513  10.500 128.082       1

#Let's figure out the NA value
which(is.na(data.combined$avg.fare))

## [1] 1044

data.combined[1044,]

##      PassengerId Survived Pclass                Name  Sex  Age SibSp Parch
## 1044        1044     None      3 Storey, Mr. Thomas male 60.5     0     0
##      Ticket Fare Cabin Embarked title familysize ticket.first.char
## 1044   3701   NA     U        S   Mr.          1                 3
##      cabin.first.char cabin.multiple last.name new.title ticket.party.size
## 1044                U              N    Storey       Mr.                 1
##      avg.fare
## 1044       NA

#Get records for similar passengers and summarize avg. fares
indexes = with(data.combined, which(Pclass == "3" & title == "Mr." &
familysize == "1" & Ticket != "3701"))
similar.na.passengers = data.combined[indexes,]
summary(similar.na.passengers$avg.fare)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   7.250   7.798   7.359   8.050  10.171

#Use median since it is very close to mean and slightly higher than mean
data.combined[is.na(avg.fare),"avg.fare"] = 7.840

#Leverage caret's preProcess function to normalize data
prepoc.data.combined = data.combined[,c("ticket.party.size", "avg.fare")]
prePoc = preProcess(prepoc.data.combined, method = c("center", "scale"))

postproc.data.combined = predict(prePoc, prepoc.data.combined)

#Let's check the correlation between avg.fare and ticket.party.size
cor(postproc.data.combined$ticket.party.size,
postproc.data.combined$avg.fare)

## [1] 0.4485116

#Correlation always results between -1 and 1 where -1 is negatively
correlated and +1 means completely correlated. 0 means no correlation at all
#Here, they are highly uncorrelated means we have two new potential features
that we could add

#How about just 1st class all up?
indexes = which(data.combined$Pclass == "1")
cor(postproc.data.combined$ticket.party.size[indexes],
postproc.data.combined$avg.fare[indexes])

## [1] 0.7088304

#Hypothesis refuted again

#Okay. Let's see if our feature engineering has made any difference or not
features = c("Pclass", "new.title","familysize", "ticket.party.size",
"avg.fare")
rpart.train.3 = data.combined[1:891,features]

#Run CV and check out results
rpart.3.cv.1 = rpart.cv(94622, rpart.train.3, rf.label, ctrl.3)
rpart.3.cv.1

## CART
##
## 891 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 10 times)
## Summary of sample sizes: 594, 594, 594, 594, 594, 594, ...
## Resampling results across tuning parameters:
```

```
##
##   cp          Accuracy   Kappa
##   0.00000000  0.8234568  0.6186544
##   0.01593063  0.8318743  0.6359498
##   0.03186126  0.8316498  0.6356703
##   0.04779189  0.8219978  0.6192369
##   0.06372252  0.7970819  0.5703850
##   0.07965316  0.7984287  0.5747427
##   0.09558379  0.7949495  0.5686167
##   0.11151442  0.7940516  0.5669656
##   0.12744505  0.7934905  0.5661919
##   0.14337568  0.7934905  0.5661919
##   0.15930631  0.7934905  0.5661919
##   0.17523694  0.7934905  0.5661919
##   0.19116757  0.7934905  0.5661919
##   0.20709821  0.7934905  0.5661919
##   0.22302884  0.7934905  0.5661919
##   0.23895947  0.7934905  0.5661919
##   0.25489010  0.7934905  0.5661919
##   0.27082073  0.7934905  0.5661919
##   0.28675136  0.7934905  0.5661919
##   0.30268199  0.7934905  0.5661919
##   0.31861262  0.7934905  0.5661919
##   0.33454325  0.7934905  0.5661919
##   0.35047389  0.7934905  0.5661919
##   0.36640452  0.7934905  0.5661919
##   0.38233515  0.7934905  0.5661919
##   0.39826578  0.7934905  0.5661919
##   0.41419641  0.7789001  0.5227001
##   0.43012704  0.7585859  0.4606306
##   0.44605767  0.7261504  0.3597443
##   0.46198830  0.7016835  0.2826611
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01593063.
```

```
#Plot
prp(rpart.3.cv.1$finalModel, type = 0, extra = 1, under = T)
```

```
## Warning: Bad 'data' field in model 'call' field.
##          To make this warning go away:
##              Call prp with roundint=FALSE,
##              or rebuild the rpart model with model=TRUE.
```




```
##########################################################
######## Submitting, scoring and some analysis #######
##########################################################

#Rpart

#Subset our test records and features
test.submit.df = data.combined[892:1309, features]

#Make predictions
rpart.3.preds = predict(rpart.3.cv.1$finalModel, test.submit.df, type =
"class")
table(rpart.3.preds)

## rpart.3.preds
##   0   1
## 263 155

#Write out a CSV file for submission to kaggle
submit.df = data.frame(PassengerId = 892:1309, Survived = rpart.3.preds)

write.csv(submit.df, file = "Rpart2.csv", row.names = F)



# random forest
```

```r
features = c("Pclass", "new.title","familysize", "ticket.party.size",
"avg.fare")
rf.train.temp = data.combined[1:891,features]

set.seed(1234)
rf.temp = randomForest(x = rf.train.temp, y = rf.label, ntree = 1000)
rf.temp

##
## Call:
##  randomForest(x = rf.train.temp, y = rf.label, ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 16.61%
## Confusion matrix:
##     0   1 class.error
## 0 500  49  0.08925319
## 1  99 243  0.28947368

test.submit.df = data.combined[892:1309, features]

# Make predictions
rf.preds = predict(rf.temp, test.submit.df)
table(rf.preds)

## rf.preds
##   0   1
## 277 141

# Write out a CSV file
submit.df = data.frame(PassengerId = 892:1309, Survived = rf.preds)

write.csv(submit.df, file = "RF2.csv", row.names = F)
```