RENNES SCHOOL OF BUSINESS

Exploring the Titanic Dataset

Vignesh Hariharan

Introduction

We chose to take up the Titanic dataset to do our R programming in Data Analysis project . We tried illustrative data visualizations to show analysis part. The tragic incident of Titanic is well known to everybody. We tried to create a Survival model to predict the survival of passenger on the Titanic.

There are three parts to script as follows

Representation: Featuring the Data
Imputation: Cleaning and inserting Missing Values
Prediction: Predicting a model Passenger

Feature Engineering

The **passenger name** variable can be broken down into additional meaningful variables. These variables can be broken used for predictions or additional new variables can be created for our use.

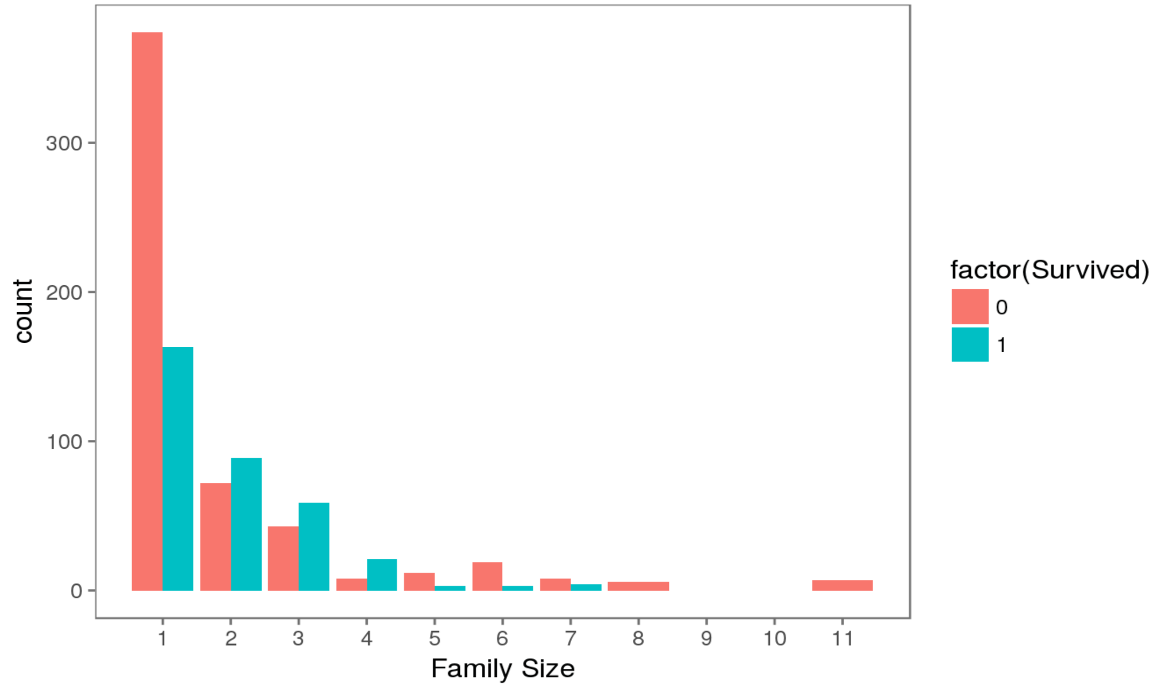
As we have used in the code, for example, the passenger name variable contains the **passenger title** and we can use **surname** to represent families. Lets do some representation using **feature engineering**!

```
# Grab title from passenger names
full$Title <- gsub('(.*, )|(\\..*)', '', full$Name)
```

Families : Survival probability as whole?

We spit passenger name into new variables. With the help of this new variable we create new family variables. At the begining?, we made a **family size** variable based on number of siblings/spouse(s) (we assumed that someone might have more than one spouse?) and number of children/parents.

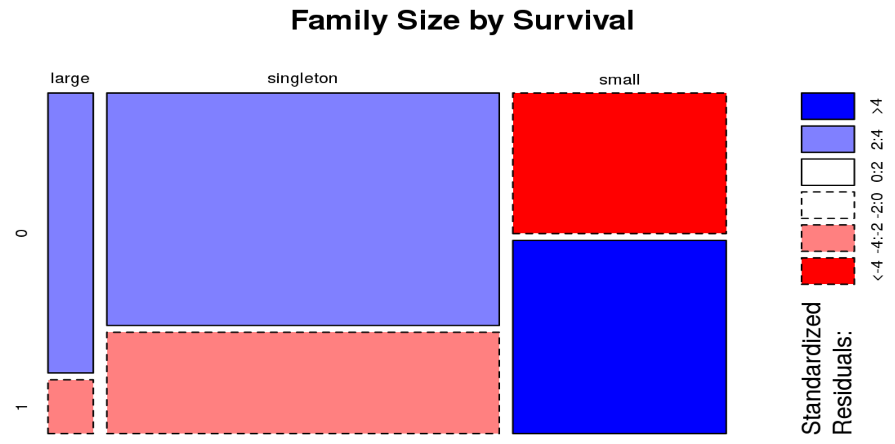
```
# Create a family size variable including the passenger themselves
full$Fsize <- full$SibSp + full$Parch + 1

# Create a family variable
full$Family <- paste(full$Surname, full$Fsize, sep='_')
```



For purpose of understanding relation between survival and how our family size variable looks like let's plot it among the training data.

```
# Use ggplot2 to visualize the relationship between family size & survival
ggplot(full[1:891,], aes(x = Fsize, fill = factor(Survived))) +
  geom_bar(stat='count', position='dodge') +
  scale_x_continuous(breaks=c(1:11)) +
  labs(x = 'Family Size') +
  theme_few()
```

As per our observation there is a survival penalty to singles and t family sizes that are more than 4. We collapsed family size to survival variable into three levels which will be helpful since there are less large families compared overly. We created a **discretized family size** variable



The mosaic plot shows that we preserve our rule that there's a survival penalty among singletons and large families, but a benefit for passengers in small families.

Treat a few more variables

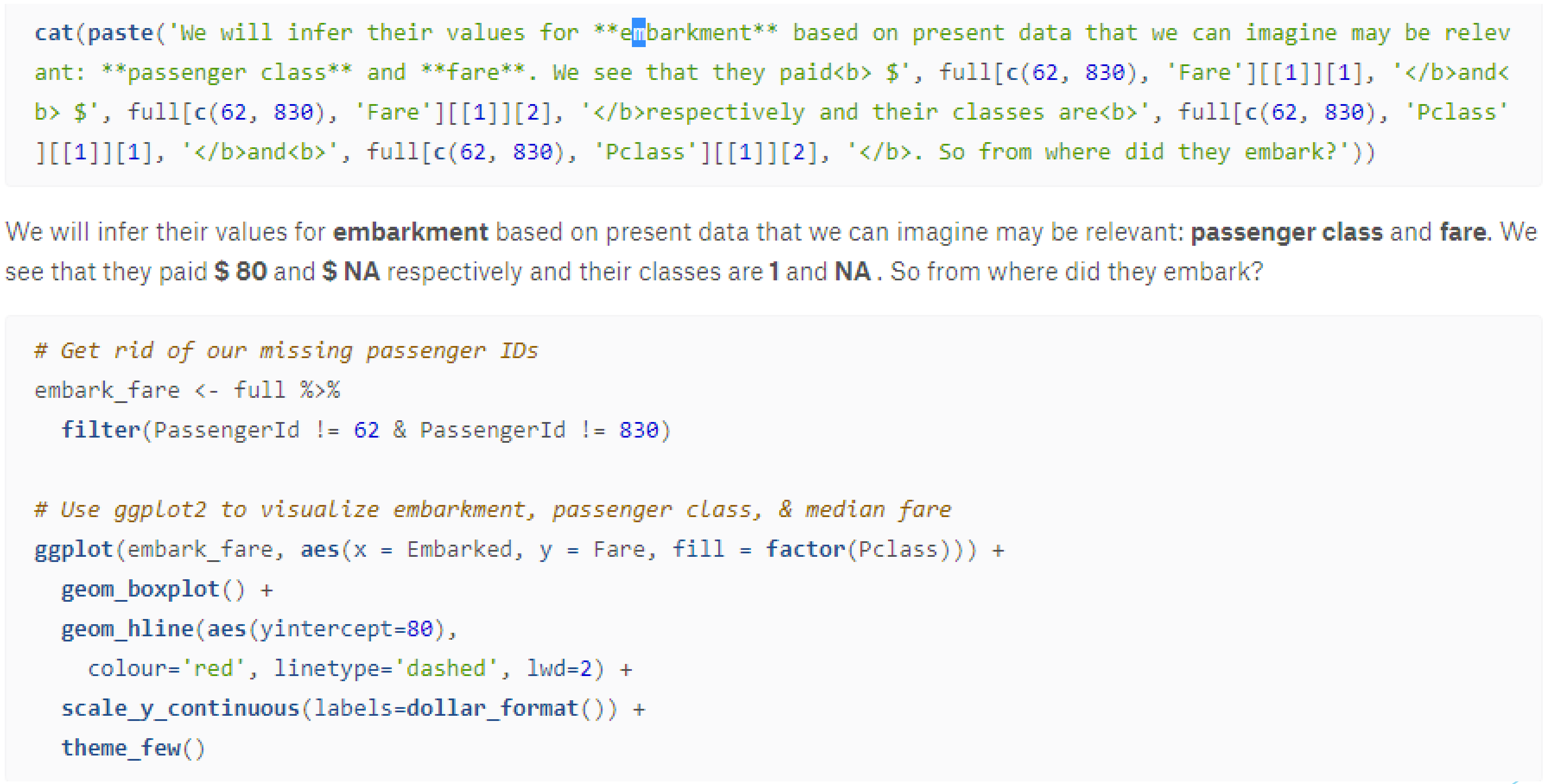
There is information that we can extract from **passenger cabin** variable including about their **deck**.

```
# This variable appears to have a lot of missing values
full$Cabin[1:28]


##  [1] ""         "C85"       ""          "C123"       ""
##  [6] ""         "E46"       ""          ""           ""
## [11] "G6"       "C103"      ""          ""           ""
## [16] ""         ""          ""          ""           ""
## [21] ""         "D56"       ""          "A6"         ""
## [26] ""         ""          "C23 C25 C27“

# The first character is the deck. For example:
strsplit(full$Cabin[2], NULL)[[1]]
```

Missingness

We had to discover missing data and rectify it through imputation. We had different options to do it but considering the small size of the dataset, we opted out of deleting either entire observations (rows) or variables (columns) containing missing values. So we decided to go forward with either replacing missing values with a sensible values as per the distribution of the data, e.g., the mean, median or mode. After this step we went ahead with prediction.

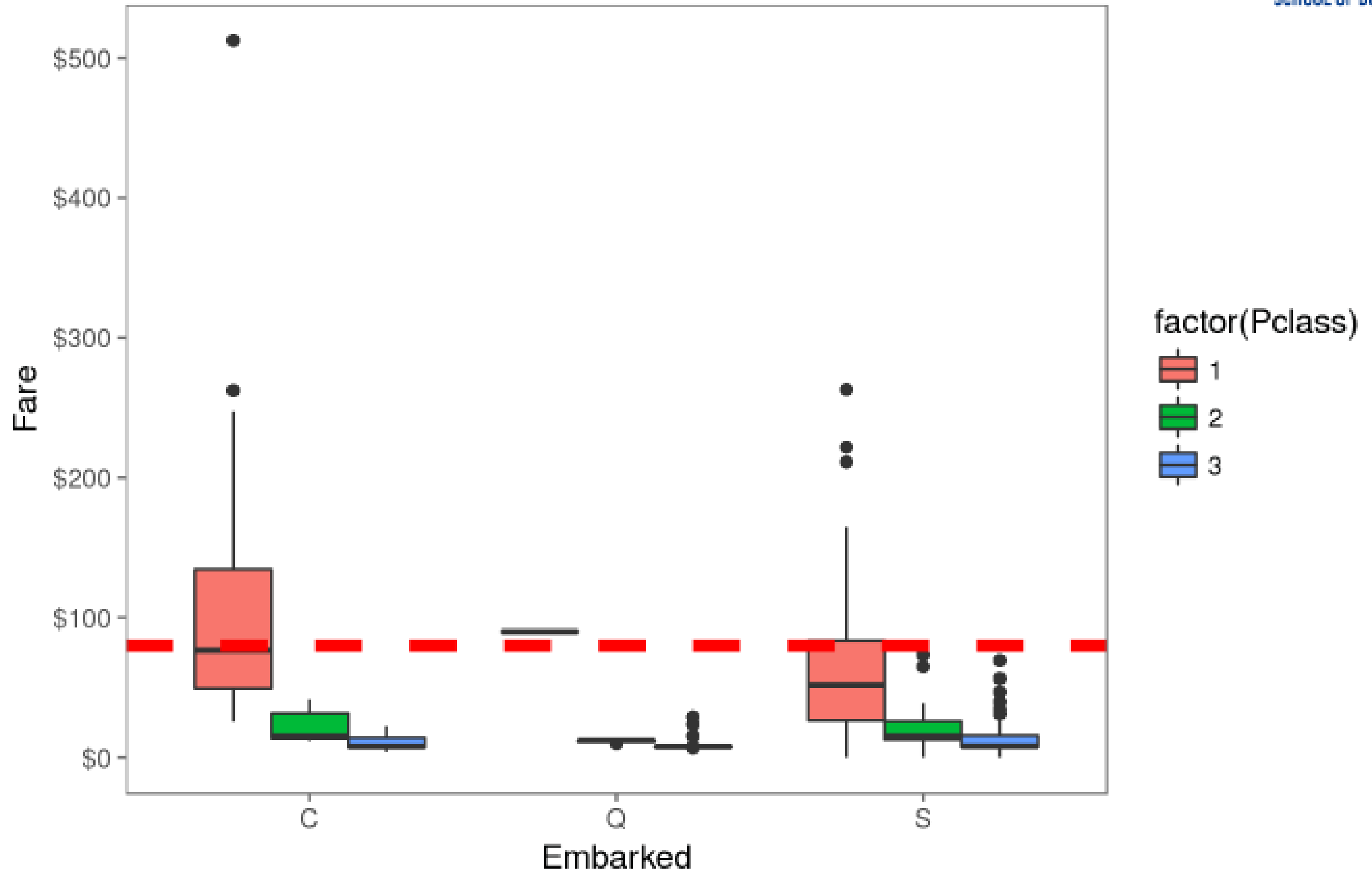
Sensible value imputation



```
cat(paste('We will infer their values for **embarkment** based on present data that we can imagine may be relev
ant: **passenger class** and **fare**. We see that they paid<b> $', full[c(62, 830), 'Fare'][[1]][1], '</b>and<
b> $', full[c(62, 830), 'Fare'][[1]][2], '</b>respectively and their classes are<b>', full[c(62, 830), 'Pclass'
][[1]][1], '</b>and<b>', full[c(62, 830), 'Pclass'][[1]][2], '</b>. So from where did they embark?'))
```

We will infer their values for **embarkment** based on present data that we can imagine may be relevant: **passenger class** and **fare**. We see that they paid **$ 80** and **$ NA** respectively and their classes are **1** and **NA** . So from where did they embark?

```
# Get rid of our missing passenger IDs
embark_fare <- full %>%
  filter(PassengerId != 62 & PassengerId != 830)

# Use ggplot2 to visualize embarkment, passenger class, & median fare
ggplot(embark_fare, aes(x = Embarked, y = Fare, fill = factor(Pclass))) +
  geom_boxplot() +
  geom_hline(aes(yintercept=80),
    colour='red', linetype='dashed', lwd=2) +
  scale_y_continuous(labels=dollar_format()) +
  theme_few()
```


Fare
$500
$400
$300
$200
$100
$0
C
Q
S
Embarked
factor(Pclass)
1
2
3

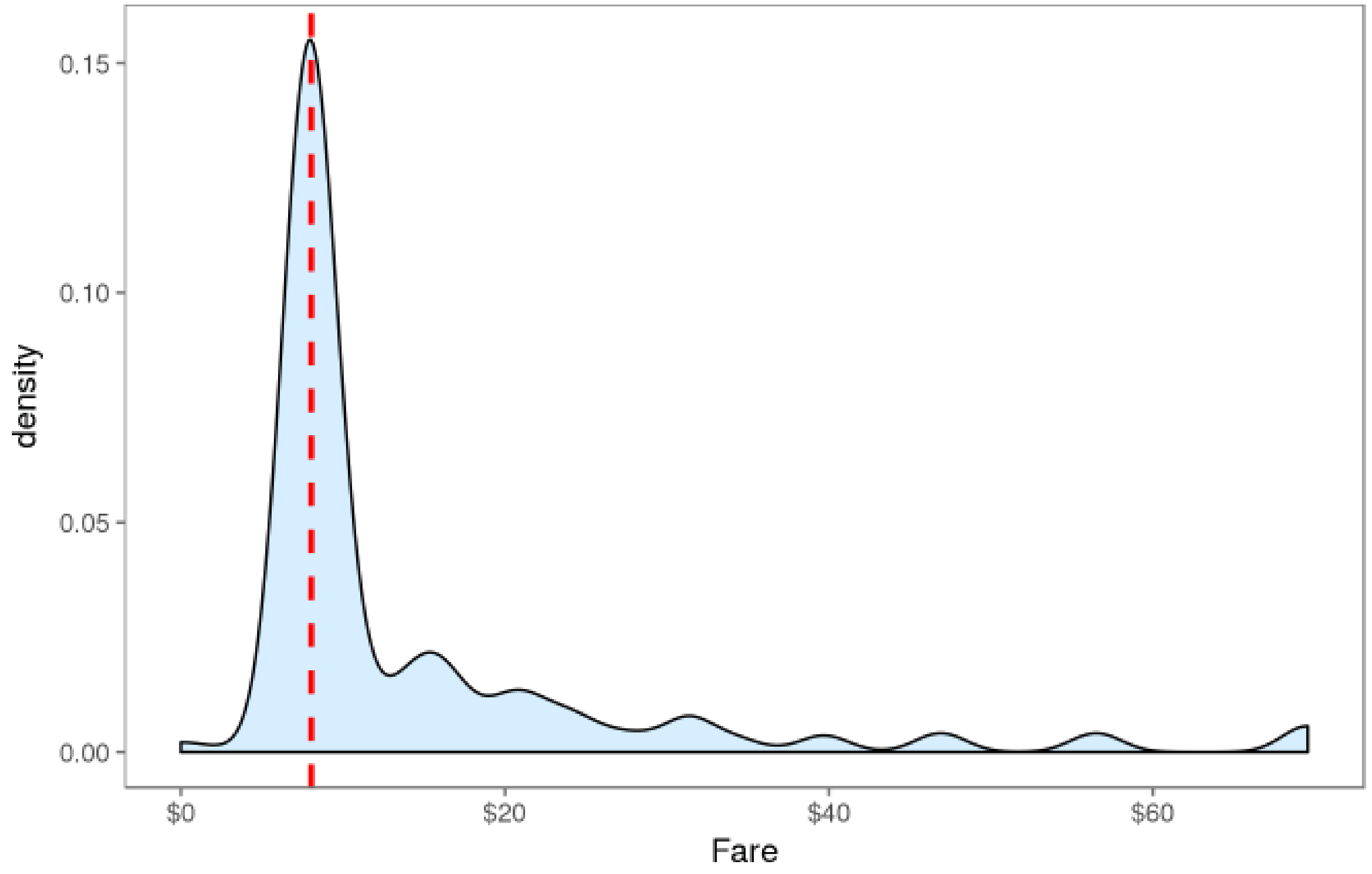
We're close to fixing the handful of NA values here and there. Passenger on row 1044 has an NA Fare value.

```r
# Show row 1044
full[1044, ]
```

```
##      PassengerId Survived Pclass               Name  Sex  Age SibSp Parch
## 1044        1044       NA      3 Storey, Mr. Thomas male 60.5     0     0
##      Ticket Fare Cabin Embarked Title Surname Fsize   Family    FsizeD
## 1044   3701   NA              S    Mr  Storey     1 Storey_1 singleton
##      Deck
## 1044 <NA>
```

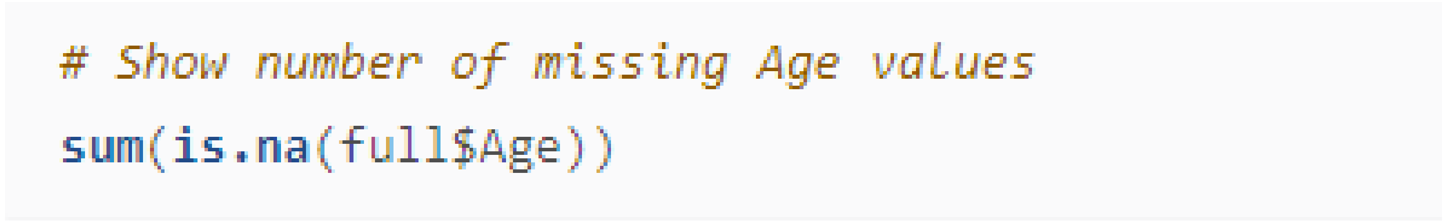
This is a third class passenger who departed from Southampton ('S'). Let's visualize Fares among all others sharing their class and embarkment (n = 494).

```r
ggplot(full[full$Pclass == '3' & full$Embarked == 'S', ],
  aes(x = Fare)) +
  geom_density(fill = '#99d6ff', alpha=0.4) +
  geom_vline(aes(xintercept=median(Fare, na.rm=T)),
    colour='red', linetype='dashed', lwd=1) +
  scale_x_continuous(labels=dollar_format()) +
  theme_few()
```

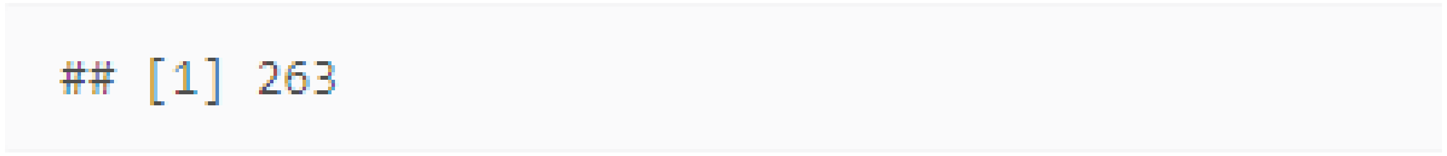

density
0.15
0.10
0.05
0.00
$0
$20
$40
$60
Fare

Predictive imputation

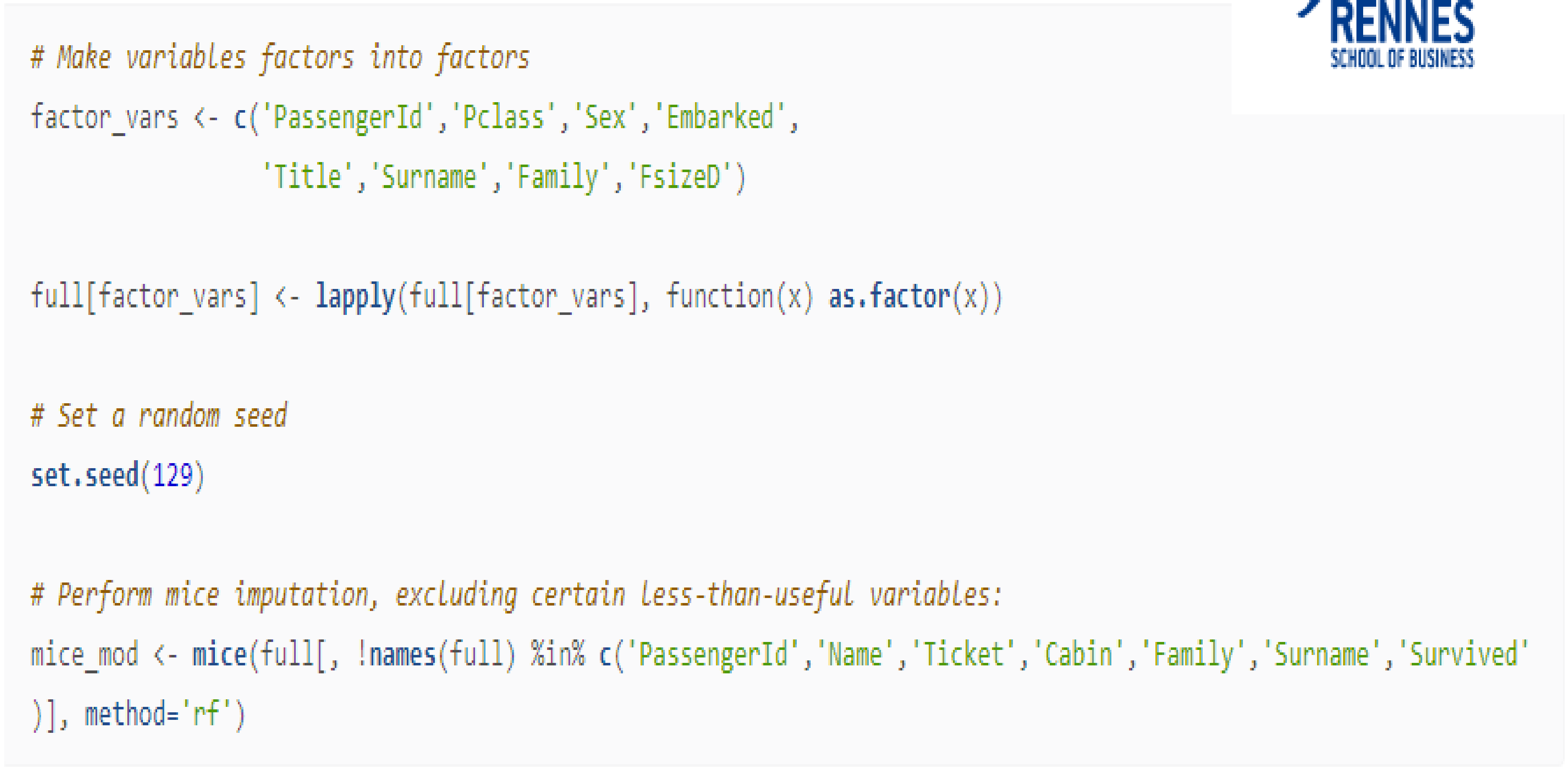
As per our former observation we encountered few missing **Age** values in our data. This we resolved by various intricate tricks of imputing missing age values. We will create a model predicting ages based on other variables.



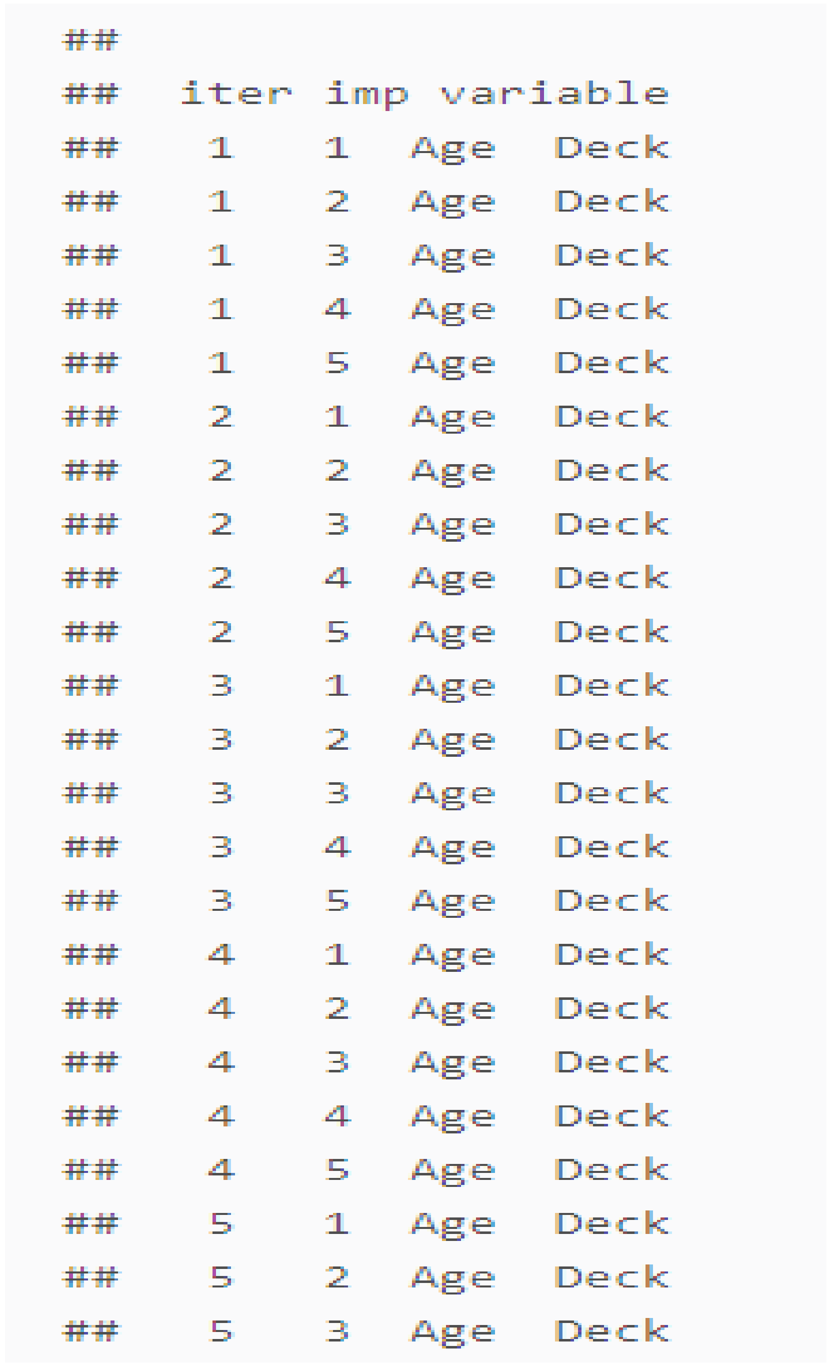
```r
# Show number of missing Age values
sum(is.na(full$Age))
```



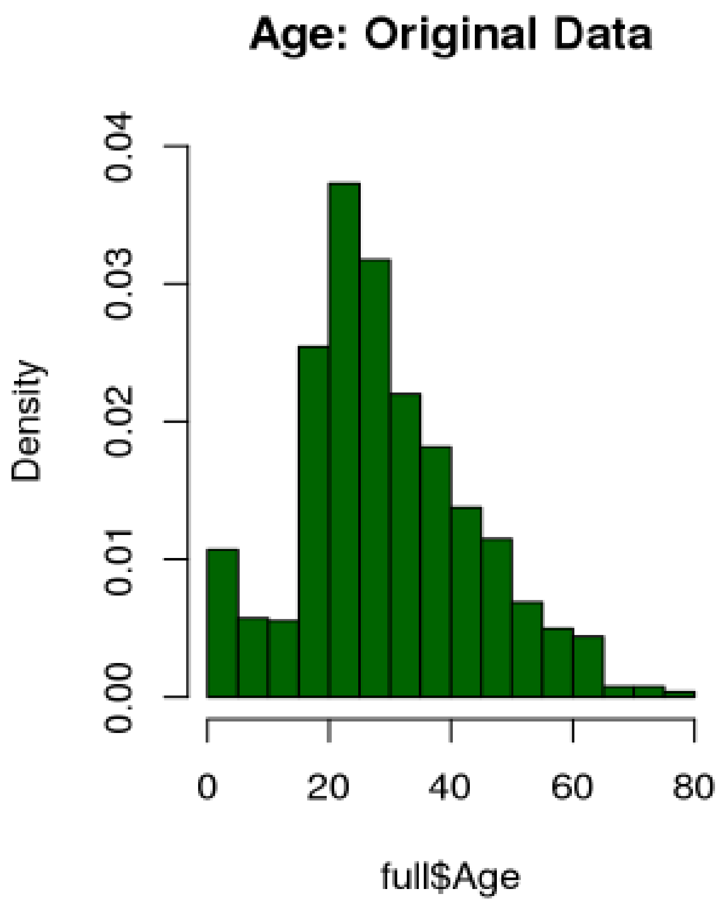
```
## [1] 263
```

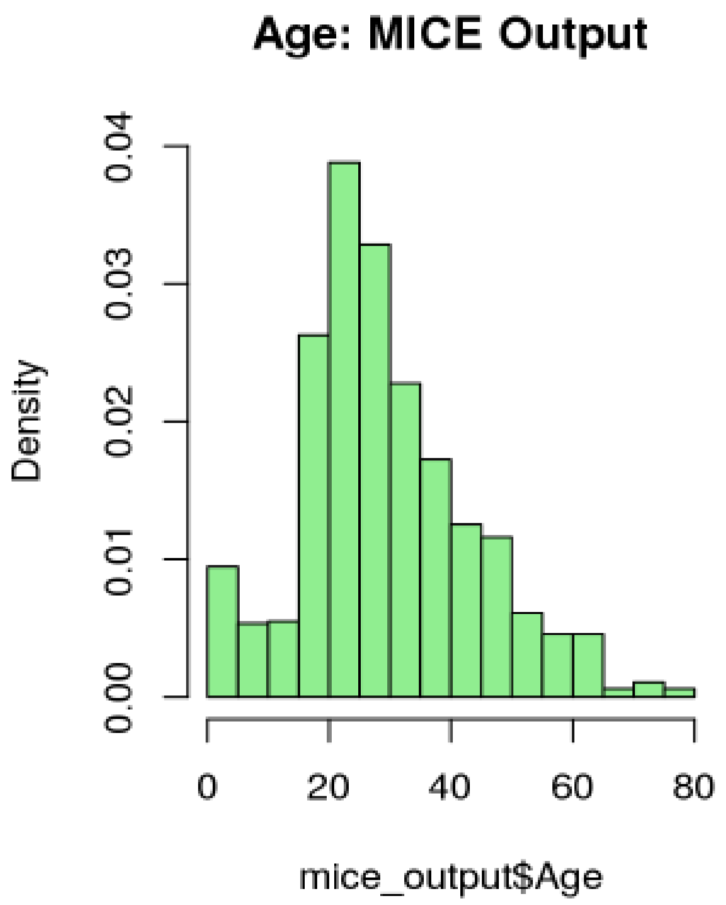


```r
# Make variables factors into factors
factor_vars <- c('PassengerId','Pclass','Sex','Embarked',
                 'Title','Surname','Family','FsizeD')

full[factor_vars] <- lapply(full[factor_vars], function(x) as.factor(x))

# Set a random seed
set.seed(129)

# Perform mice imputation, excluding certain less-than-useful variables:
mice_mod <- mice(full[, !names(full) %in% c('PassengerId','Name','Ticket','Cabin','Family','Surname','Survived'
)], method='rf')
```



```
##
##   iter imp variable
##    1   1  Age  Deck
##    1   2  Age  Deck
##    1   3  Age  Deck
##    1   4  Age  Deck
##    1   5  Age  Deck
##    2   1  Age  Deck
##    2   2  Age  Deck
##    2   3  Age  Deck
##    2   4  Age  Deck
##    2   5  Age  Deck
##    3   1  Age  Deck
##    3   2  Age  Deck
##    3   3  Age  Deck
##    3   4  Age  Deck
##    3   5  Age  Deck
##    4   1  Age  Deck
##    4   2  Age  Deck
##    4   3  Age  Deck
##    4   4  Age  Deck
##    4   5  Age  Deck
##    5   1  Age  Deck
##    5   2  Age  Deck
##    5   3  Age  Deck
```

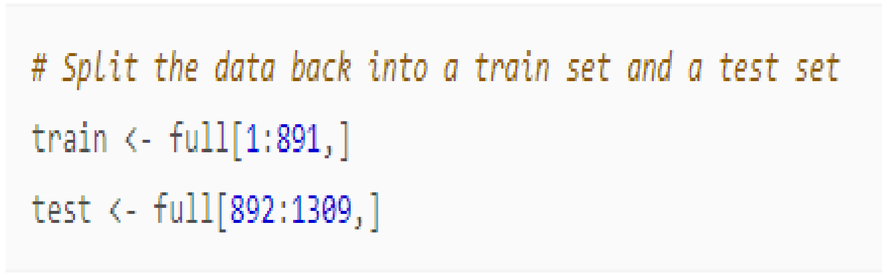

Age: Original Data
Density
0.00
0.01
0.02
0.03
0.04
0
20
40
60
80
full$Age


Age: MICE Output
Density
0.00
0.01
0.02
0.03
0.04
0
20
40
60
80
mice_output$Age

Prediction

Split into training & test sets

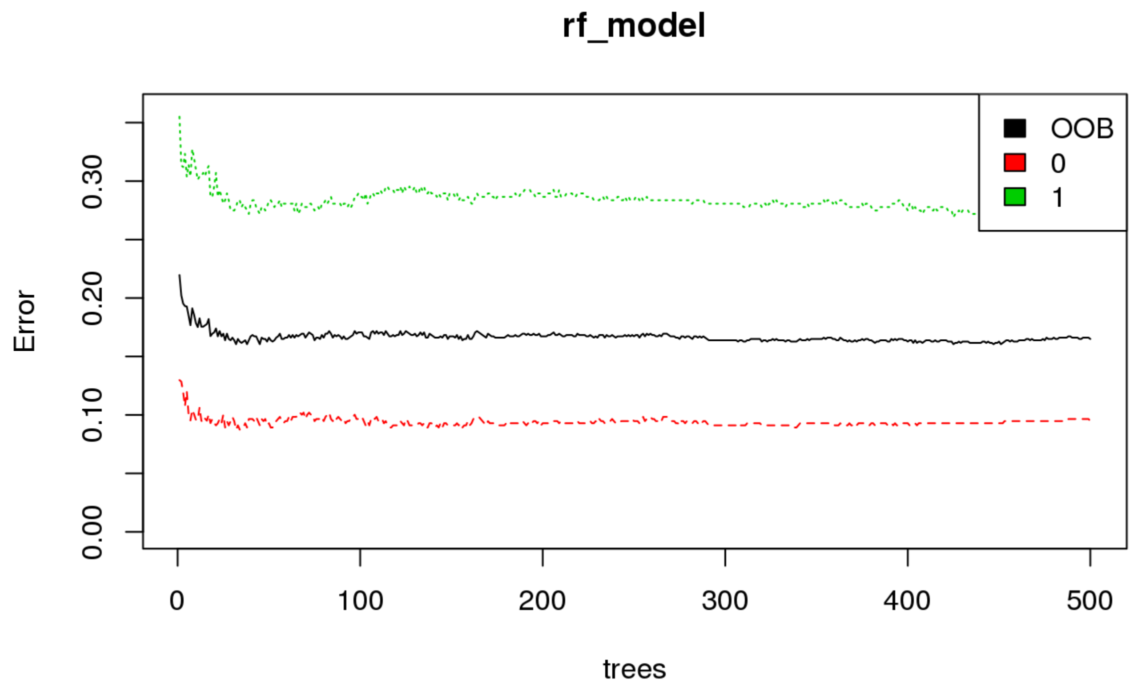
split the data back into the original test and training sets.



```
# Split the data back into a train set and a test set
train <- full[1:891,]
test <- full[892:1309,]
```

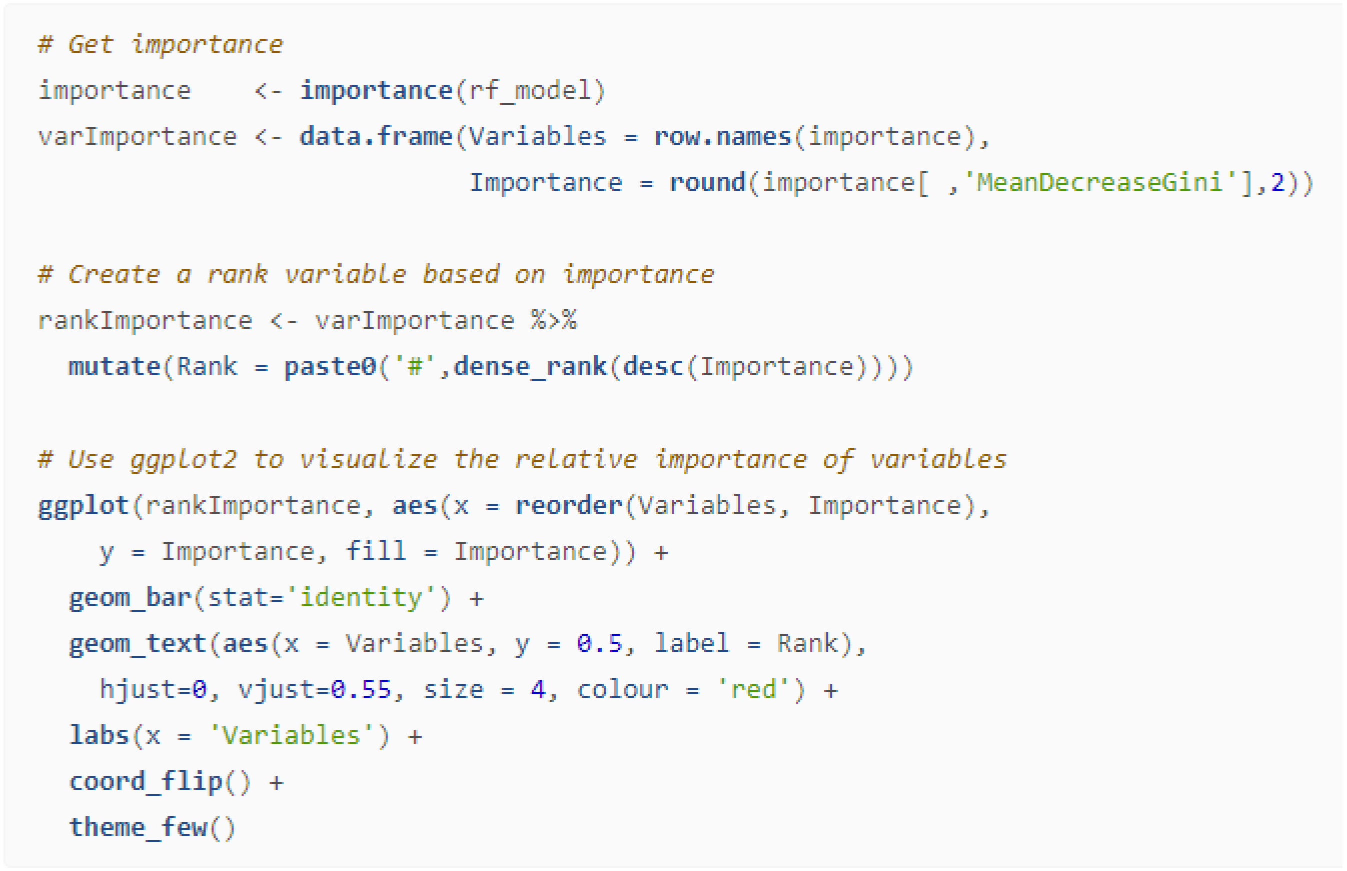
Building the model

```
# Set a random seed
set.seed(754)

# Build the model (note: not all possible variables are used)
rf_model <- randomForest(factor(Survived) ~ Pclass + Sex + Age + SibSp + Parch +
                                            Fare + Embarked + Title +
                                            FsizeD + Child + Mother,
                                            data = train)

# Show model error
plot(rf_model, ylim=c(0,0.36))
legend('topright', colnames(rf_model$err.rate), col=1:3, fill=1:3)
```

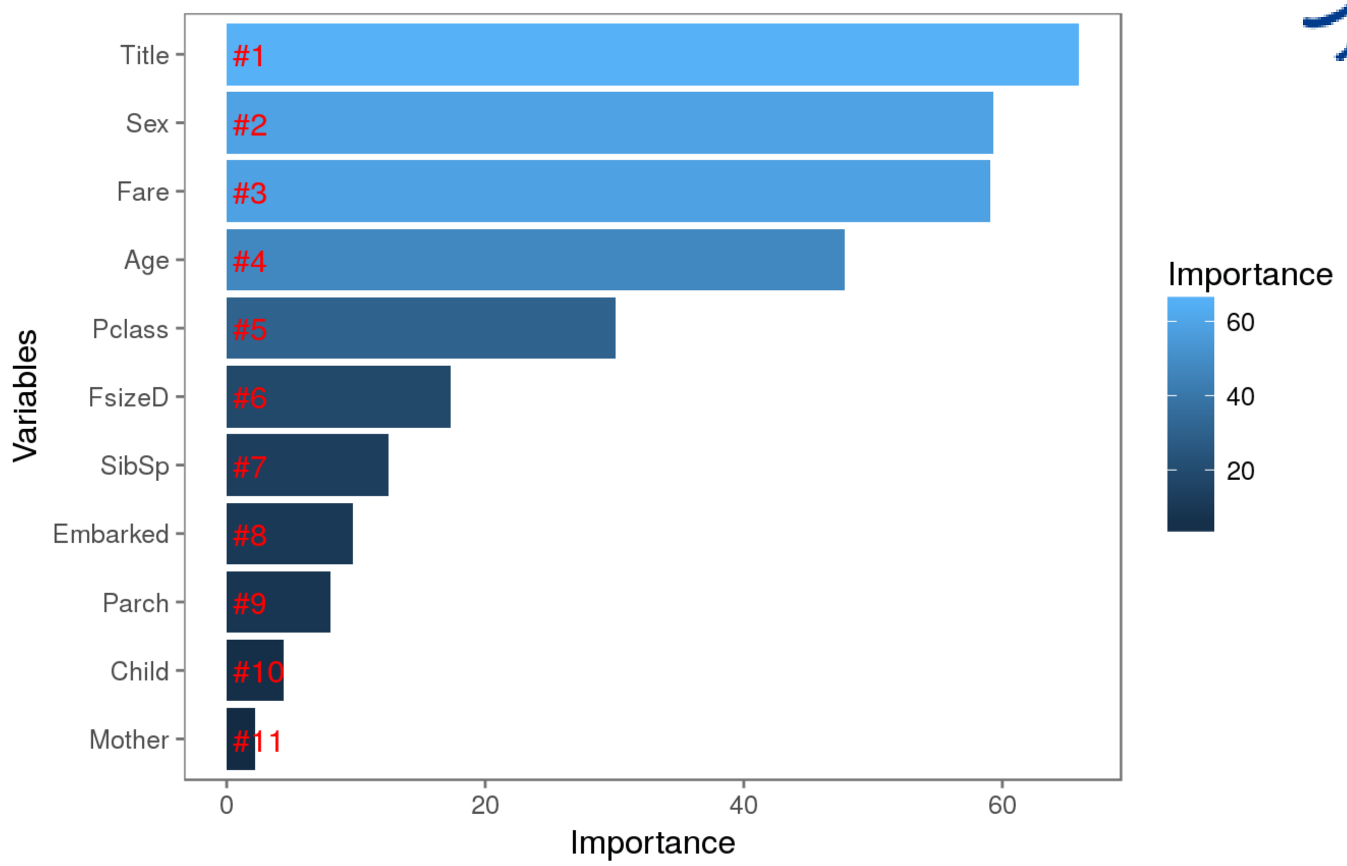


The black line shows the overall error rate which falls below 20%. The red lines show the error rate for 'died' and green lines show the error rate for 'survived' . Now we're efficient in predicting death than we are at predicting the survival.

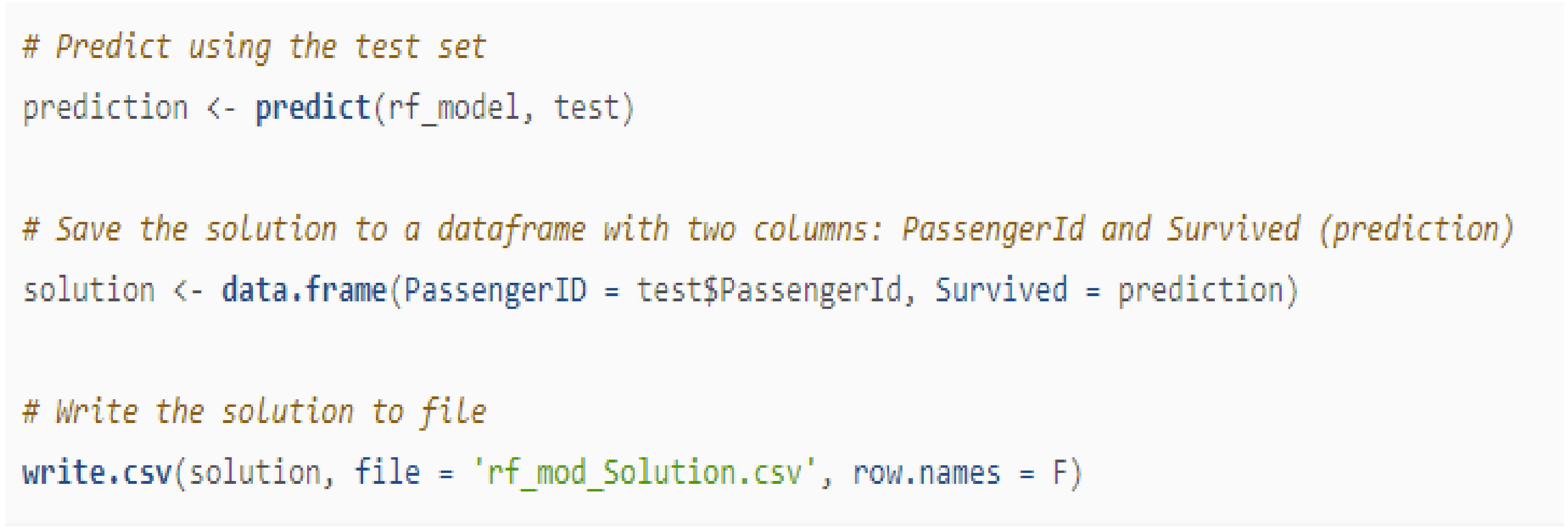
Variable importance



```r
# Get importance
importance    <- importance(rf_model)
varImportance <- data.frame(Variables = row.names(importance),
                            Importance = round(importance[ ,'MeanDecreaseGini'],2))

# Create a rank variable based on importance
rankImportance <- varImportance %>%
  mutate(Rank = paste0('#',dense_rank(desc(Importance))))

# Use ggplot2 to visualize the relative importance of variables
ggplot(rankImportance, aes(x = reorder(Variables, Importance),
    y = Importance, fill = Importance)) +
  geom_bar(stat='identity') +
  geom_text(aes(x = Variables, y = 0.5, label = Rank),
    hjust=0, vjust=0.55, size = 4, colour = 'red') +
  labs(x = 'Variables') +
  coord_flip() +
  theme_few()
```


Variables
Title
Sex
Fare
Age
Pclass
FsizeD
SibSp
Embarked
Parch
Child
Mother
#1
#2
#3
#4
#5
#6
#7
#8
#9
#10
#11
0
20
40
60
Importance
Importance
60
40
20

Conclusion



```r
# Predict using the test set
prediction <- predict(rf_model, test)

# Save the solution to a dataframe with two columns: PassengerId and Survived (prediction)
solution <- data.frame(PassengerID = test$PassengerId, Survived = prediction)

# Write the solution to file
write.csv(solution, file = 'rf_mod_Solution.csv', row.names = F)
```