Machine Learning with R: Logistic Regression

Armand Tossou

9/6/2021

For this lecture we will be working with the Titanic Data Set from Kaggle. This is a very famous data set and very often is a student's first step in machine learning! We'll be trying to predict a classification- survival or deceased.

Let's begin our understanding of implementing Logistic Regression in R for classification.

We'll use a "semi-cleaned" version of the titanic data set, if you use the data set hosted directly on Kaggle, you may need to do some additional cleaning not shown in this lecture notebook.

The Data

We can begin by loading in our training data into data frames:

```
# load the dataset
df.train <- read.csv('C:/Users/adtos/Dropbox/Data_Science/R programming/Data Science and Machine Learni

# preview the data
head(df.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
```

Get a summary of the data:

```
summary(df)
```

```
## Error in object[[i]]: object of type 'closure' is not subsettable
```

Exploratory Data Analysis (EDA)

Let's explore how much missing data we have, we can use the `Amelia` pacakge for this. Install it if you want to follow along, you'll need to install it later for your logistic regression project.

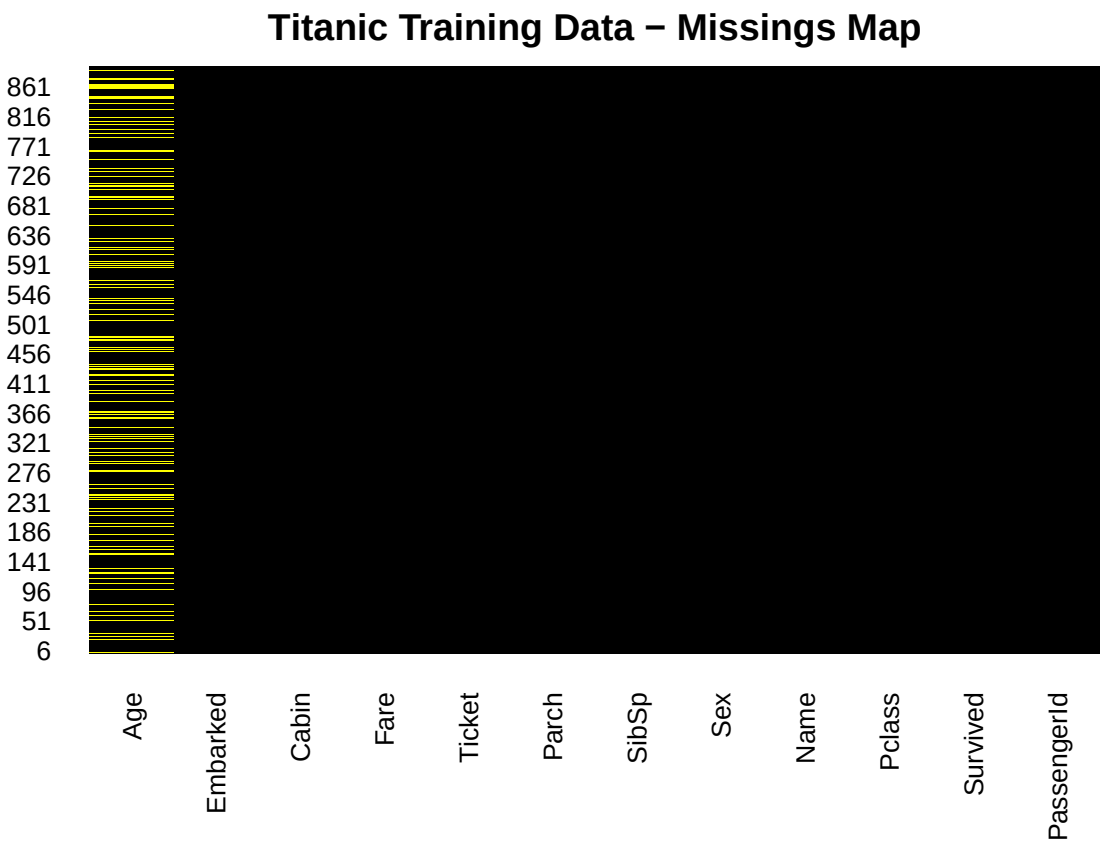
```
# install the Amelia library
#install.packages("Amelia")

# call the Amelia library
library(Amelia)
```

```
## Loading required package: Rcpp
```

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.8.0, built: 2021-05-26)
## ## Copyright (C) 2005-2021 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
# plot a map of missing values
missmap(df.train, main="Titanic Training Data - Missings Map",
        col=c("yellow", "black"), legend=FALSE)
```



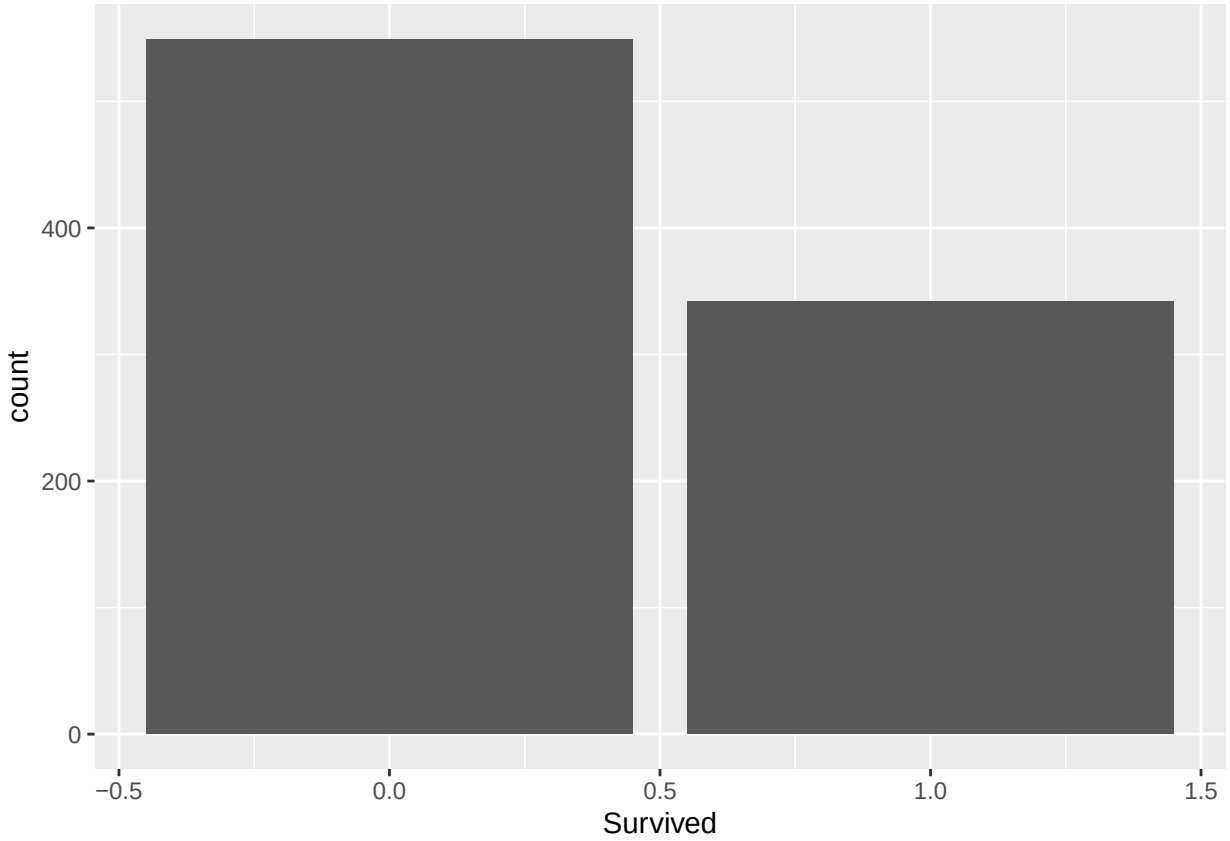
Roughly 20 percent of the Age data is missing. The proportion of Age "missings" is likely small enough for reasonable replacement with some form of imputation.

Let's continue on by visualizing some of the data.

Data Visualization with ggplot2

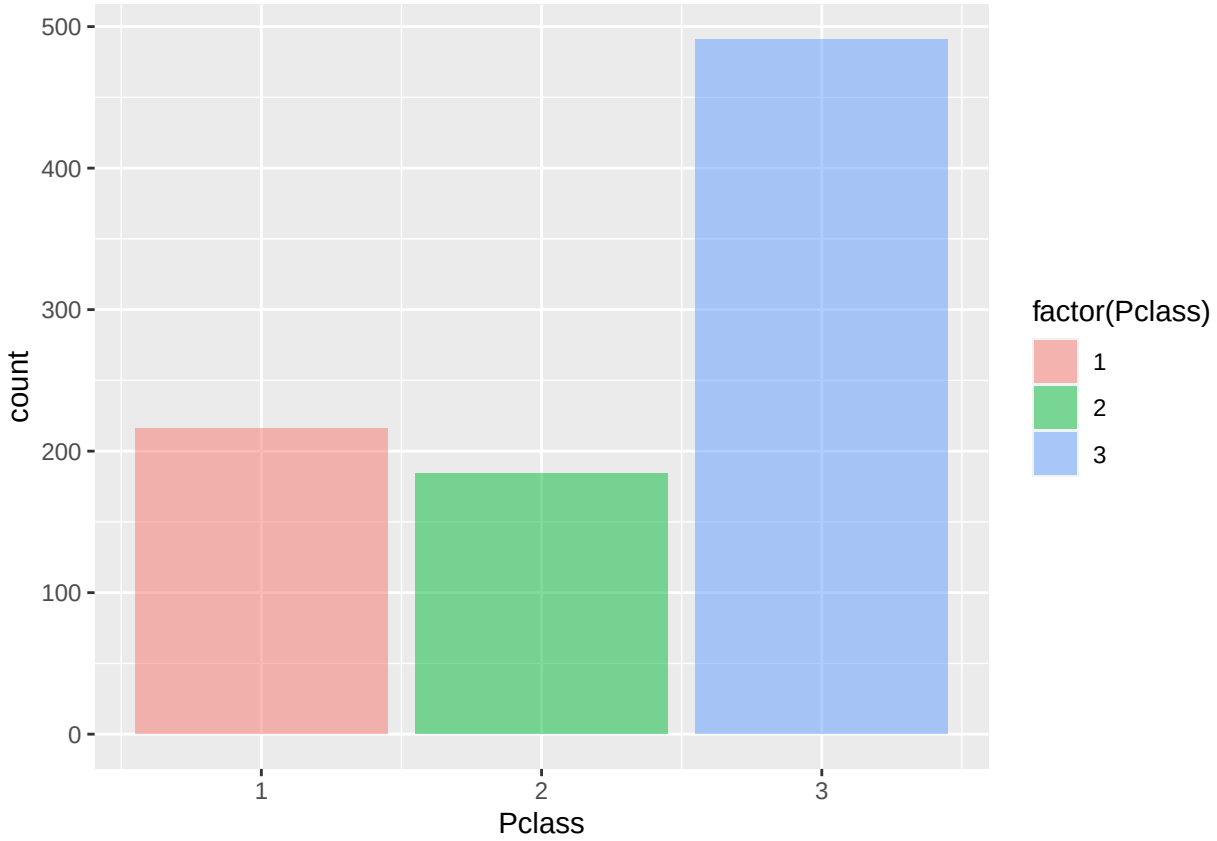
```
# load necessary library
library(ggplot2)

# create a barplot of 'Survived'
ggplot(df.train,aes(Survived)) + geom_bar()
```



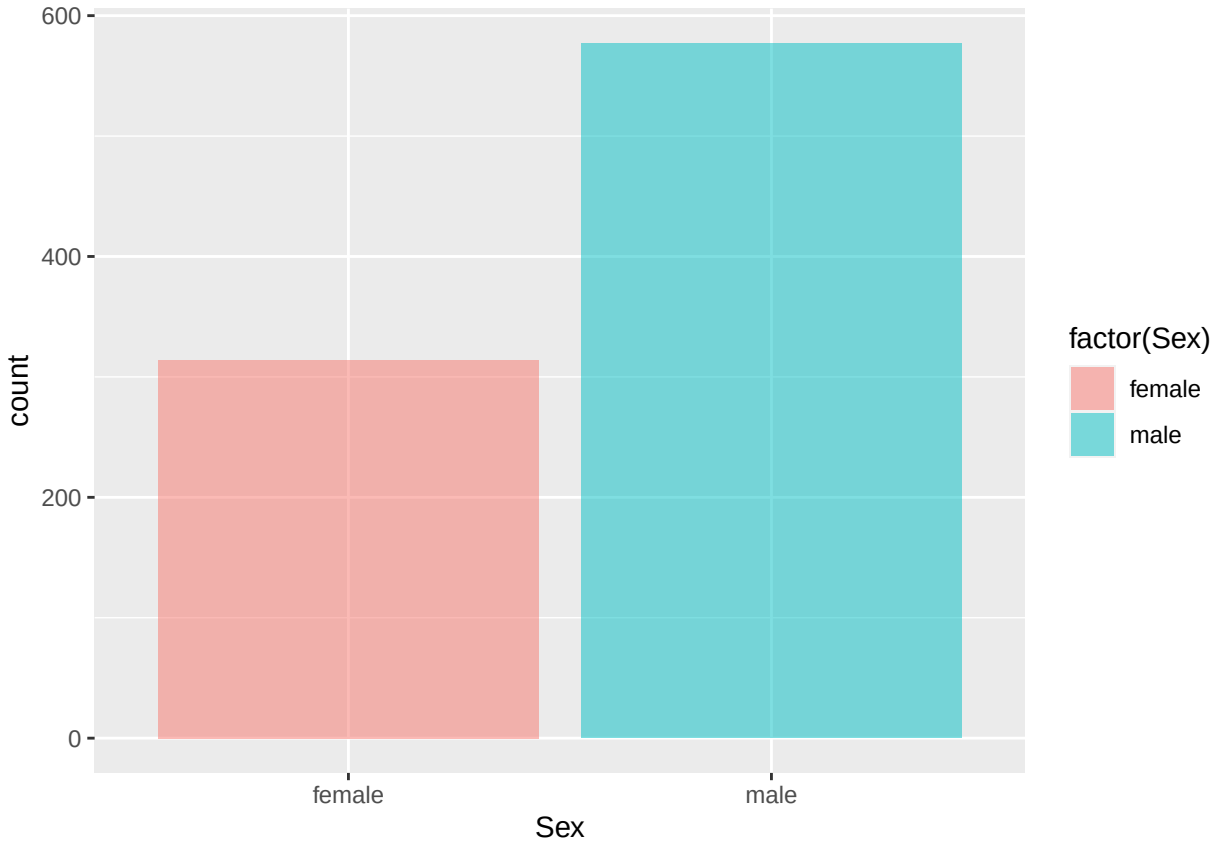
Also create a barplot of 'PClass'.

```
ggplot(df.train,aes(Pclass)) + geom_bar(aes(fill=factor(Pclass)),alpha=0.5)
```



Create a barplot of 'Sex'.

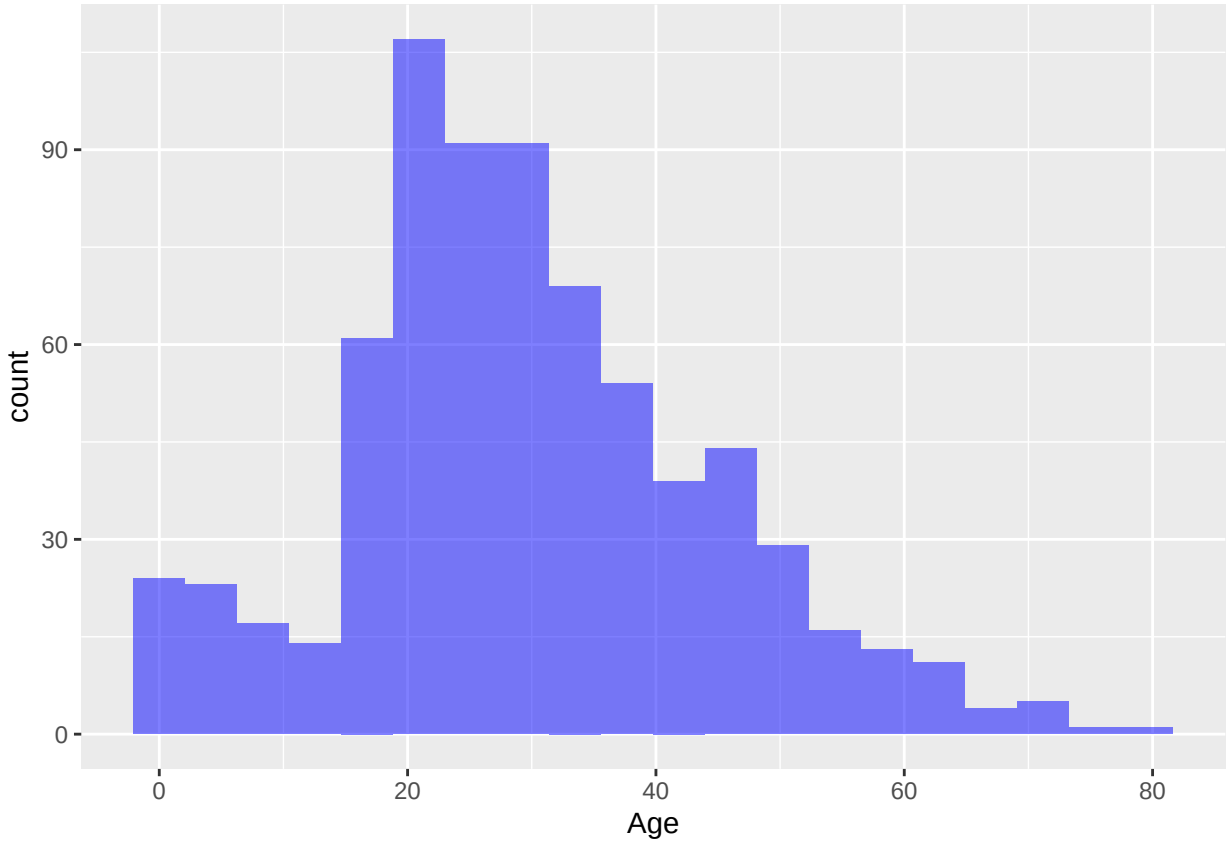
```
ggplot(df.train,aes(Sex)) + geom_bar(aes(fill=factor(Sex)),alpha=0.5)
```



Create a histogram of 'Age'.

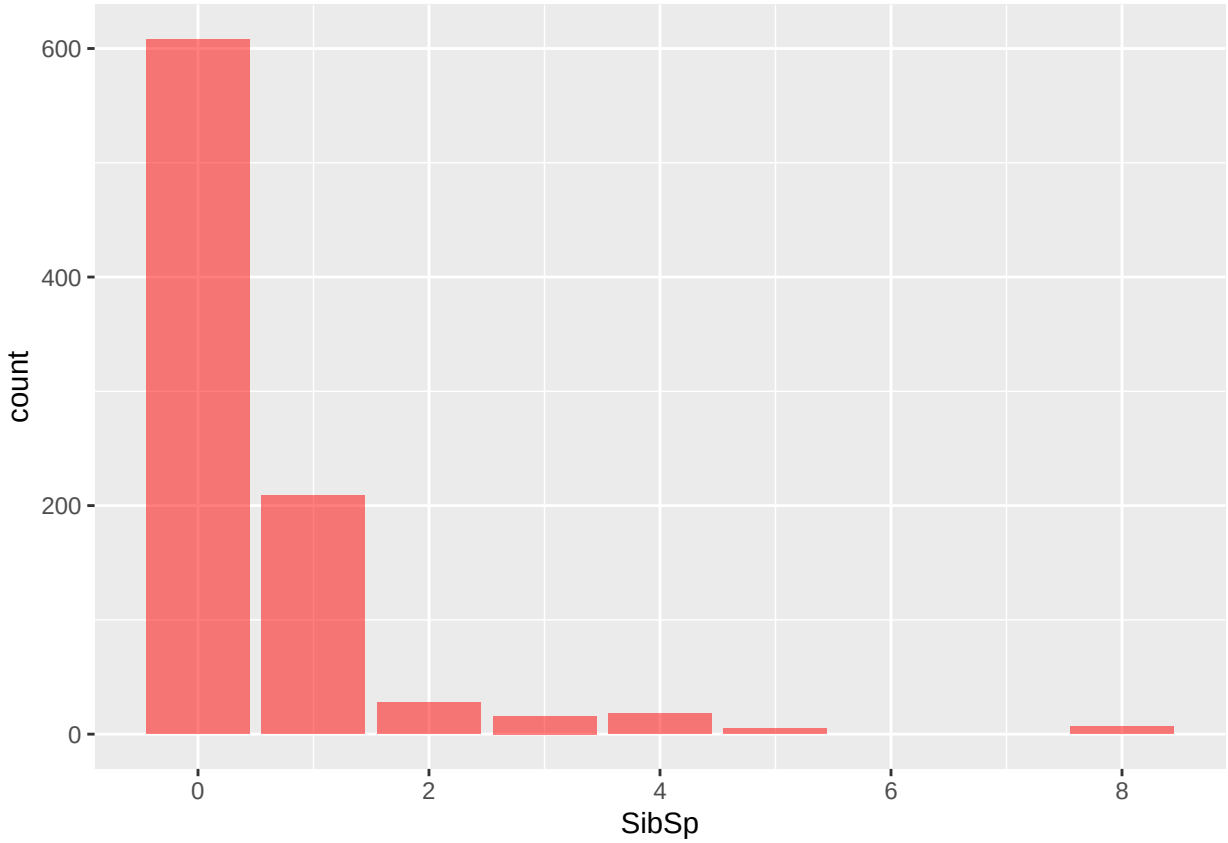
```
ggplot(df.train,aes(Age)) + geom_histogram(fill='blue',bins=20,alpha=0.5)
```

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



Create a barplot of 'SibSp'.

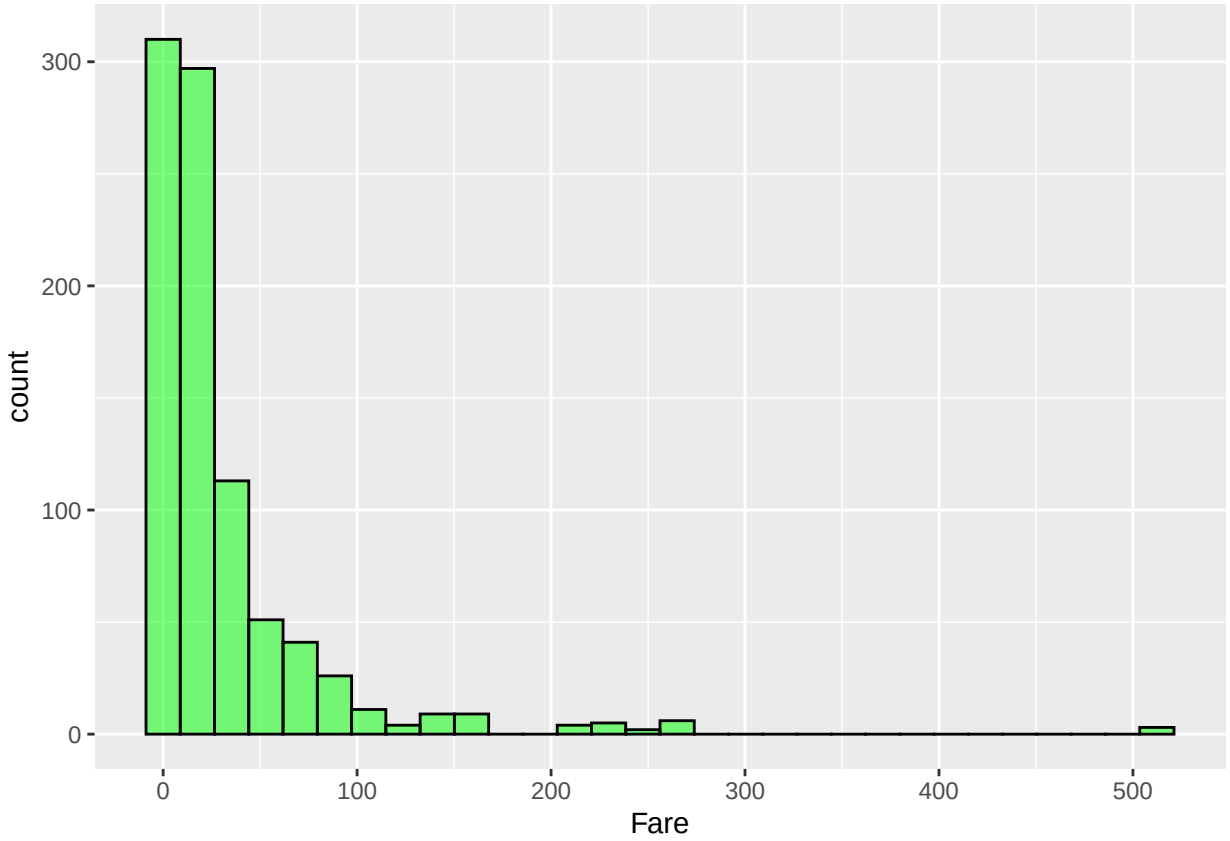
```
ggplot(df.train,aes(SibSp)) + geom_bar(fill='red',alpha=0.5)
```



Create a histogram of 'Fare'.

```
ggplot(df.train,aes(Fare)) + geom_histogram(fill='green',color='black',alpha=0.5)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



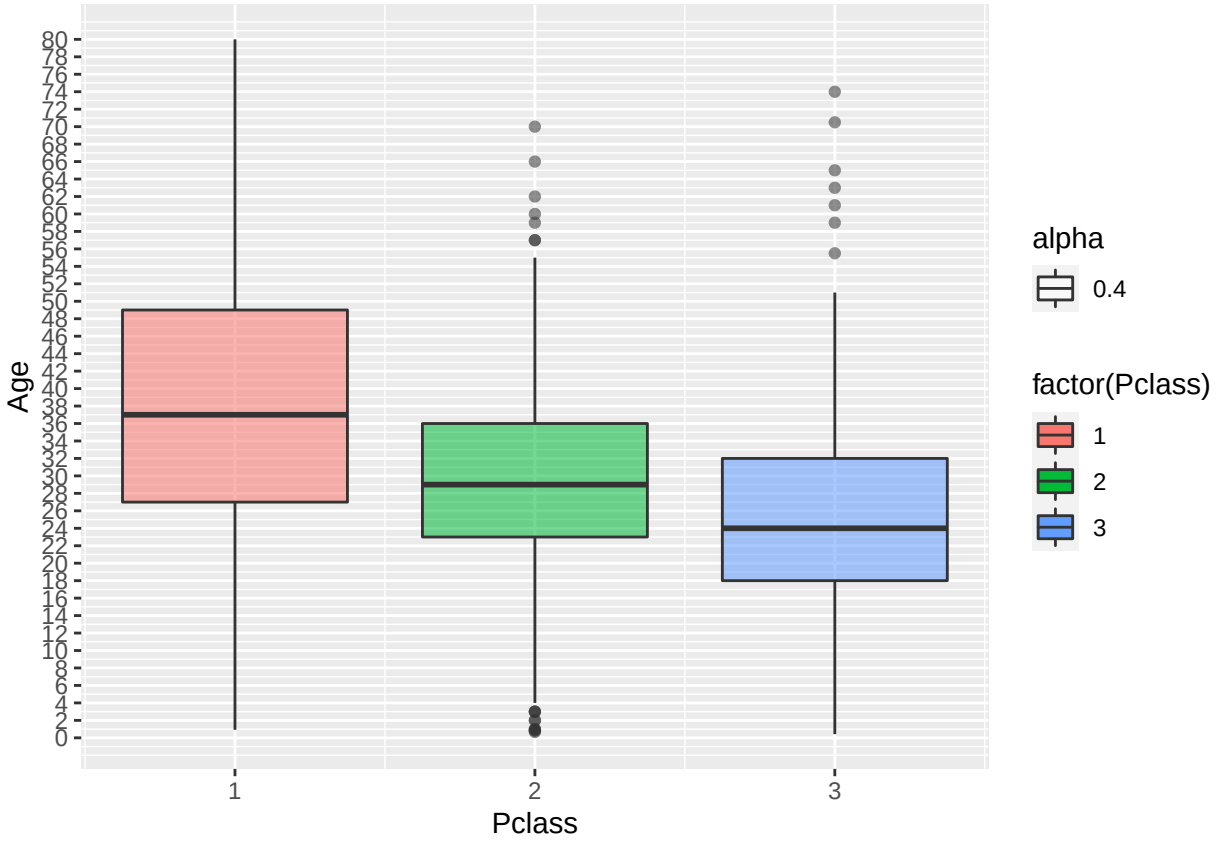
Data Cleaning

We want to fill in missing age data instead of just dropping the missing age data rows. One way to do this is by filling in the mean age of all the passengers (i.e., imputation).

However, we can be smarter about this and check the average age by passenger class. For example:

```
pl <- ggplot(df.train,aes(Pclass,Age)) + geom_boxplot(aes(group=Pclass,fill=factor(Pclass),alpha=0.4))
pl + scale_y_continuous(breaks = seq(min(0), max(80), by = 2))
```

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



We can see the wealthier passengers in the higher classes tend to be older, which makes sense. We'll use these average age values to impute based on Pclass for Age.

Let's create a function to use for imputation:

```
impute_age <- function(age,class){
    out <- age
    for (i in 1:length(age)){

        if (is.na(age[i])){

            if (class[i] == 1){
                out[i] <- 37

            }else if (class[i] == 2){
                out[i] <- 29

            }else{
                out[i] <- 24
            }
        }else{
            out[i]<-age[i]
        }
    }
    return(out)
}
```

Now, let's apply the imputation function:

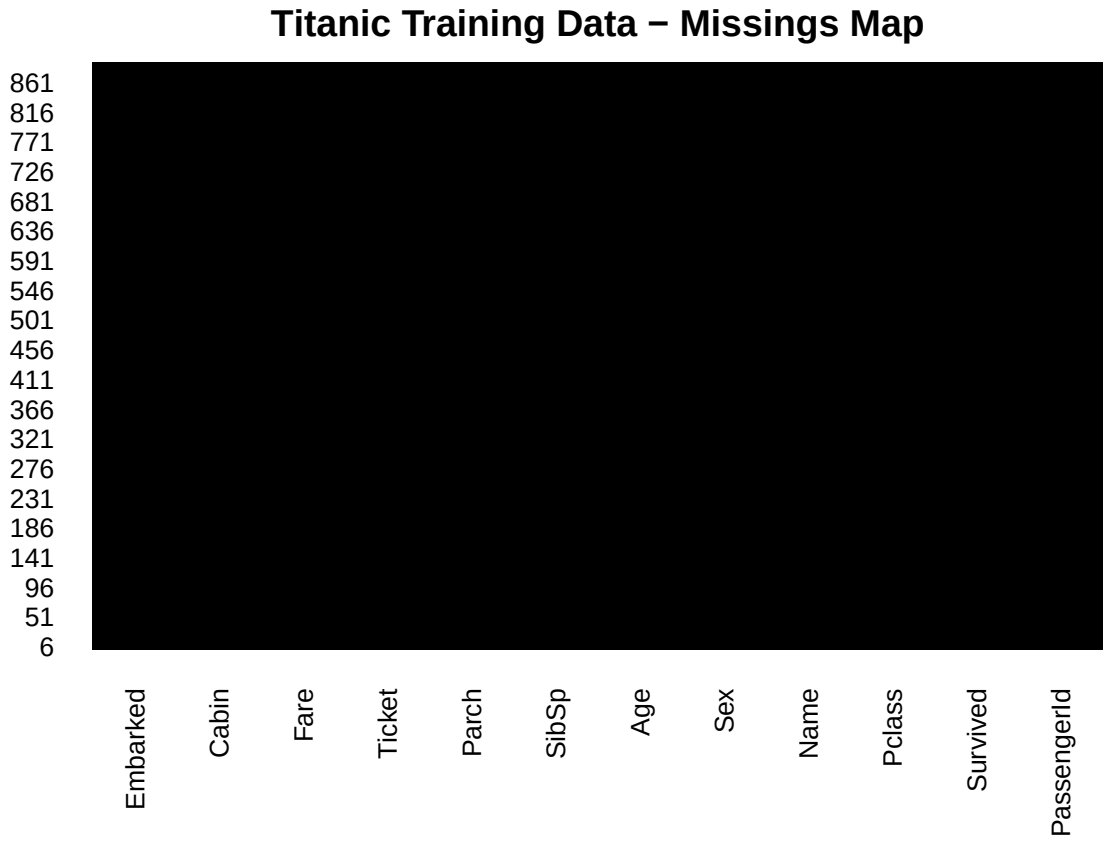
```
fixed.ages <- impute_age(df.train$Age,df.train$Pclass)
```

Replace 'Age' in the training dataset with the imputed version:

```
df.train$Age <- fixed.ages
```

Now let's check to see if our imputation approach for missing values worked. We plot the 'missmap' again using the `Amelia` package:

```
missmap(df.train, main="Titanic Training Data - Missings Map",
        col=c("yellow", "black"), legend=FALSE)
```



Great let's continue with building our model!

Building a Logistic Regression Model

Now it is time to build our model! Let's begin by doing a final "clean-up" of our data by removing the features we won't be using and making sure that the features are of the correct data type.

First, let's check the structure of our data:

```
str(df.train)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 24 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     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

Let's remove what we won't use:

```
head(df.train,3)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      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
##             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
```

Let's select the relevant columns for training:

```
# load the 'dplyr' library
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

We'll exclude these 4 columns from our analysis: - PassengerId, - Name, - Ticket, - Cabin

```
df.train <- select(df.train,-PassengerId,-Name,-Ticket,-Cabin)

head(df.train,3)
```

```
##   Survived Pclass    Sex Age SibSp Parch    Fare Embarked
## 1        0      3   male  22     1     0  7.2500        S
## 2        1      1 female  38     1     0 71.2833        C
## 3        1      3 female  26     0     0  7.9250        S
```

Now let's set factor columns.

```
df.train$Survived <- factor(df.train$Survived)
df.train$Pclass <- factor(df.train$Pclass)
df.train$Parch <- factor(df.train$Parch)
df.train$SibSp <- factor(df.train$SibSp)
```

Check to make sure the column types are correct:

```
str(df.train)
```

```
## 'data.frame':    891 obs. of  8 variables:
##  $ Survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass  : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Sex     : chr  "male" "female" "female" "female" ...
##  $ Age     : num  22 38 26 35 35 24 54 2 27 14 ...
##  $ SibSp   : Factor w/ 7 levels "0","1","2","3",..: 2 2 1 2 1 1 1 4 1 2 ...
##  $ Parch   : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 2 3 1 ...
##  $ Fare    : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked: chr  "S" "C" "S" "S" ...
```

Train the Model

Now let's train the model!

```
# fit the model
log.model <- glm(formula=Survived ~ . , family = binomial(link='logit'),data = df.train)

# get summary
summary(log.model)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial(link = "logit"),
##     data = df.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8158  -0.6134  -0.4138   0.5808   2.4896
##
## Coefficients:
```

```
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.845e+01  1.660e+03   0.011 0.991134
## Pclass2     -1.079e+00  3.092e-01  -3.490 0.000484 ***
## Pclass3     -2.191e+00  3.161e-01  -6.930 4.20e-12 ***
## Sexmale     -2.677e+00  2.040e-01 -13.123  < 2e-16 ***
## Age         -3.971e-02  8.758e-03  -4.534 5.79e-06 ***
## SibSp1       8.135e-02  2.245e-01   0.362 0.717133
## SibSp2      -2.897e-01  5.368e-01  -0.540 0.589361
## SibSp3      -2.241e+00  7.202e-01  -3.111 0.001862 **
## SibSp4      -1.675e+00  7.620e-01  -2.198 0.027954 *
## SibSp5      -1.595e+01  9.588e+02  -0.017 0.986731
## SibSp8      -1.607e+01  7.578e+02  -0.021 0.983077
## Parch1       3.741e-01  2.895e-01   1.292 0.196213
## Parch2       3.862e-02  3.824e-01   0.101 0.919560
## Parch3       3.655e-01  1.056e+00   0.346 0.729318
## Parch4      -1.586e+01  1.055e+03  -0.015 0.988007
## Parch5      -1.152e+00  1.172e+00  -0.983 0.325771
## Parch6      -1.643e+01  2.400e+03  -0.007 0.994536
## Fare         2.109e-03  2.490e-03   0.847 0.397036
## EmbarkedC   -1.458e+01  1.660e+03  -0.009 0.992995
## EmbarkedQ   -1.456e+01  1.660e+03  -0.009 0.993001
## EmbarkedS   -1.486e+01  1.660e+03  -0.009 0.992857
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.66  on 890  degrees of freedom
## Residual deviance:  763.41  on 870  degrees of freedom
## AIC: 805.41
##
## Number of Fisher Scoring iterations: 15
```

We can see clearly that Sex,Age, and Class are the most significant features. This makes sense, given the women and children first policy.

Predicting using Test Cases

Let's make a test set out of our training set, retrain on the smaller version of our training set and check it against the test subset.

```
library(caTools)
set.seed(101)

split = sample.split(df.train$Survived, SplitRatio = 0.70)

final.train = subset(df.train, split == TRUE)
final.test = subset(df.train, split == FALSE)
```

Now let's rerun our model on only our final training set:

```r
# fit the model
final.log.model <- glm(formula=Survived ~ . , family = binomial(link='logit'), data = final.train)

# get model summary
summary(final.log.model)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial(link = "logit"),
##     data = final.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8288  -0.5607  -0.4096   0.6174   2.4898
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.777e+01  2.400e+03   0.007 0.994091
## Pclass2     -1.230e+00  3.814e-01  -3.225 0.001261 **
## Pclass3     -2.160e+00  3.841e-01  -5.624 1.87e-08 ***
## Sexmale     -2.660e+00  2.467e-01 -10.782  < 2e-16 ***
## Age         -3.831e-02  1.034e-02  -3.705 0.000212 ***
## SibSp1      -2.114e-02  2.755e-01  -0.077 0.938836
## SibSp2      -4.000e-01  6.463e-01  -0.619 0.536028
## SibSp3      -2.324e+00  8.994e-01  -2.584 0.009765 **
## SibSp4      -1.196e+00  8.302e-01  -1.440 0.149839
## SibSp5      -1.603e+01  9.592e+02  -0.017 0.986666
## SibSp8      -1.633e+01  1.004e+03  -0.016 0.987019
## Parch1       7.290e-01  3.545e-01   2.056 0.039771 *
## Parch2       1.406e-01  4.504e-01   0.312 0.754892
## Parch3       7.919e-01  1.229e+00   0.645 0.519226
## Parch4      -1.498e+01  1.552e+03  -0.010 0.992300
## Parch5      -9.772e-03  1.378e+00  -0.007 0.994343
## Parch6      -1.635e+01  2.400e+03  -0.007 0.994563
## Fare         3.128e-03  3.091e-03   1.012 0.311605
## EmbarkedC   -1.398e+01  2.400e+03  -0.006 0.995353
## EmbarkedQ   -1.387e+01  2.400e+03  -0.006 0.995386
## EmbarkedS   -1.431e+01  2.400e+03  -0.006 0.995243
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 829.60  on 622  degrees of freedom
## Residual deviance: 530.63  on 602  degrees of freedom
## AIC: 572.63
##
## Number of Fisher Scoring iterations: 15
```

Now let's check our prediction accuracy!

```
fitted.probabilities <- predict(final.log.model,newdata=final.test,type='response')
```

Now let's calculate from the predicted values:

```
fitted.results <- ifelse(fitted.probabilities > 0.5,1,0)
```

And let's compute the accuracy score for the model:

```
misClasificError <- mean(fitted.results != final.test$Survived)
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.798507462686567"
```

Looks like we were able to achieve around 80% accuracy, where as random guessing would have just been 50% accuracy. Let's see the confusion matrix:

```
table(final.test$Survived, fitted.probabilities > 0.5)
```

```
##
##     FALSE TRUE
##   0   140   25
##   1    29   74
```