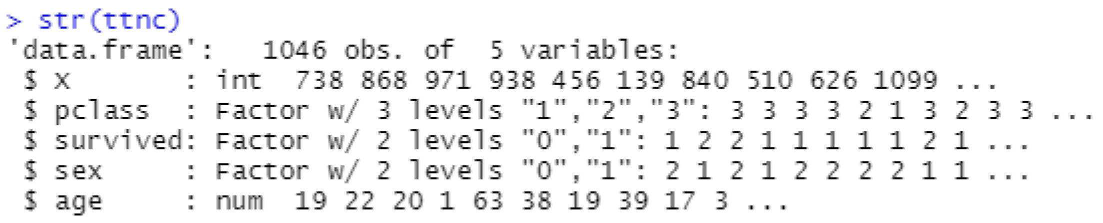
Titanic Survival Report

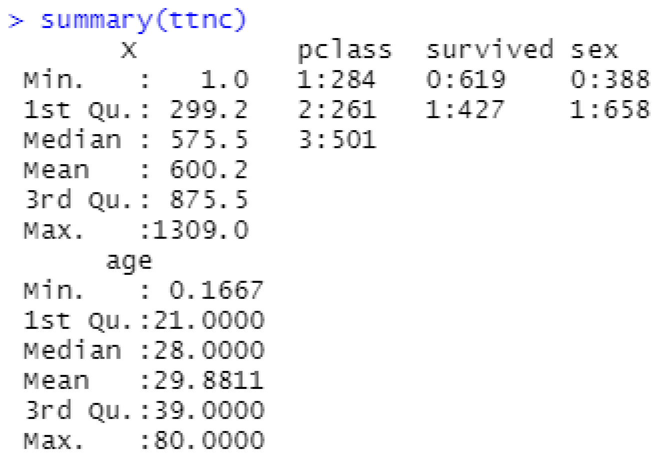
Data Explorations Functions (*performed 4 basic and 2 on models)

Before building models we want to explore the data to gain a better understanding of the variables and how they may be related; we do this with built-in R functions. First was using the str function to just get an idea of the structure of the data:



```
> str(ttnc)
'data.frame':	1046 obs. of  5 variables:
 $ X       : int  738 868 971 938 456 139 840 510 626 1099 ...
 $ pclass  : Factor w/ 3 levels "1","2","3": 3 3 3 3 2 1 3 2 3 3 ...
 $ survived: Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 1 2 1 ...
 $ sex     : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 2 2 1 1 ...
 $ age     : num  19 22 20 1 63 38 19 39 17 3 ...
```

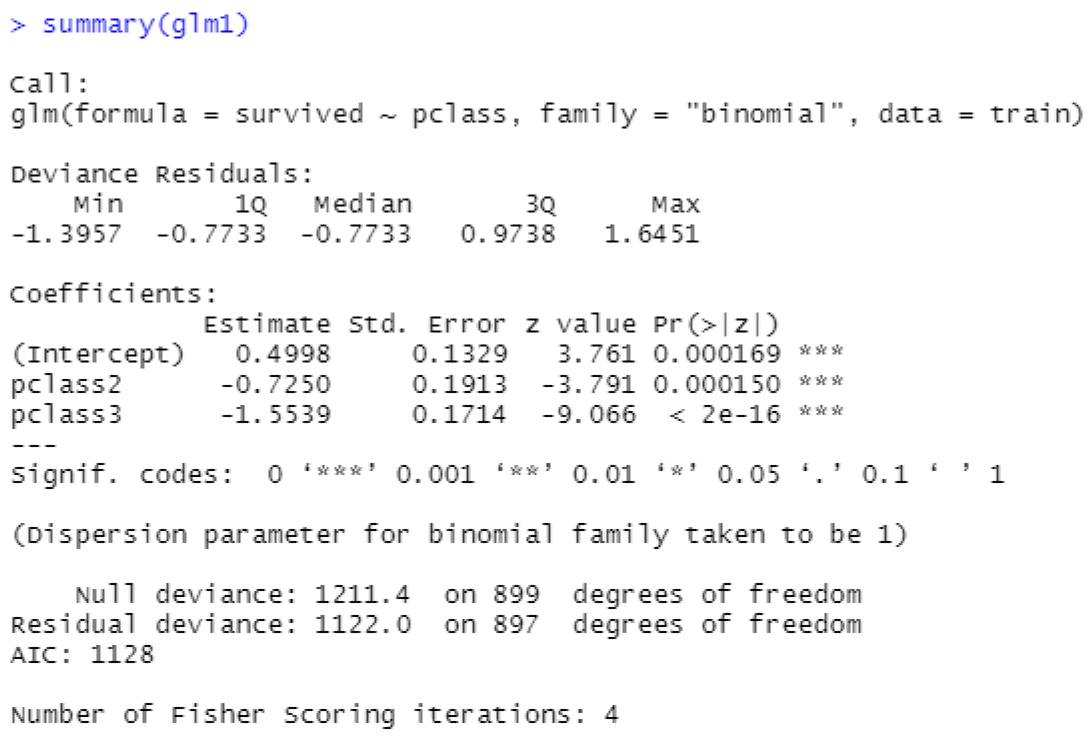
We see that there are 1046 instances and 5 columns, X, pclass, survived, sex, and age. The column X is hard to understand but perhaps it might be simply a passenger ID of sorts; it's not too important as we are working with the other variables. Next, pclass is the passenger class and is in three levels, first to third. The survived variable is if a passenger survived or not, sex is gender, 0 for female and 1 for male, and finally, age is simple the age of a passenger. Not to get ahead of myself, but these factors were originally numerical so I changed them to factors to make them easier to work with our classification algorithms (except for age). Let's dive deeper by looking at the summary statistics for each column with the summary function:



```
> summary(ttnc)
       X          pclass  survived sex    
 Min.   :   1.0   1:284   0:619    0:388  
 1st Qu.: 299.2   2:261   1:427    1:658  
 Median : 575.5   3:501                   
 Mean   : 600.2                           
 3rd Qu.: 875.5                           
 Max.   :1309.0                           
      age         
 Min.   : 0.1667  
 1st Qu.:21.0000  
 Median :28.0000  
 Mean   :29.8811  
 3rd Qu.:39.0000  
 Max.   :80.0000  
```

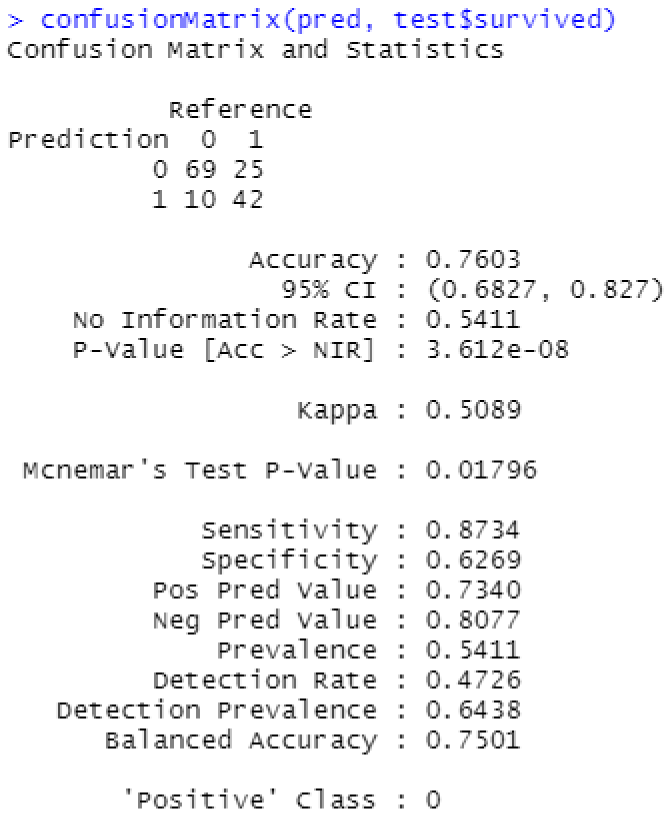
We can see that this a large data set, which is good, and that the factor variables are generally balanced. A few uneven ratios I would keep in mind is that there are much more passengers in third class compared to the other two classes, there are slightly more people who perished, and there are much more males than females. The contrast isn't too large so it's nothing to worry about but good to know when understanding some of the results. For the age variable, we see the values range from .1667 to 80 and that the median age is 28 and the mean 29.8811. Most of the passengers are middle-aged and this shows with these metrics. Moreover, with the median and mean so close to one another, we can say this is a symmetrical distribution with little skew.

There wasn't much that simple data exploration functions were giving me about just the data so I decided to perform some exploration on the performance in the model (after creating the model). First I wanted to see how the logistic regression model performed with the data using the summary() function:



```
> summary(glm1)

Call:
glm(formula = survived ~ pclass, family = "binomial", data = train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.3957  -0.7733  -0.7733   0.9738   1.6451  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)   0.4998     0.1329   3.761 0.000169 ***
pclass2      -0.7250     0.1913  -3.791 0.000150 ***
pclass3      -1.5539     0.1714  -9.066  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1211.4  on 899  degrees of freedom
Residual deviance: 1122.0  on 897  degrees of freedom
AIC: 1128

Number of Fisher Scoring iterations: 4
```

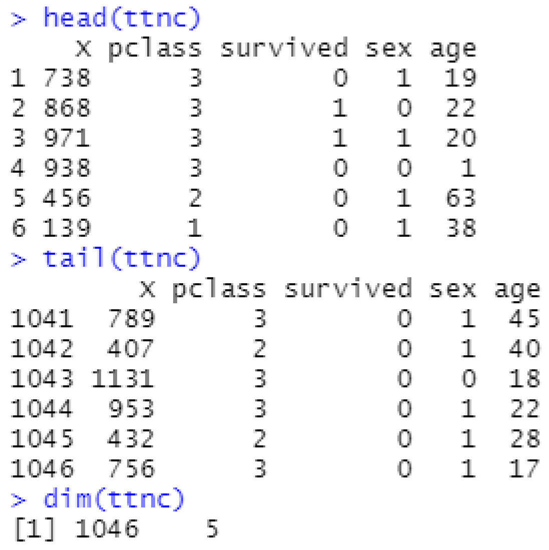
We see with the pclass dummy variables that if a person is in second class instead of first, the log odds of their survival decreases by 0.7250 and for class 3 it’s 1.5539, much greater. Looking at the deviances, there wasn’t much of a drop from the null to residual deviance so pclass may not be a good predictor, or maybe not just by itself. The naive bayes classifier considers more variables so let’s see if it performed much better. The summary function doesn’t produce the same output and simply printing the model gives a large number of conditional probabilities so let’s use the confusionMatrix() function instead:



```
> confusionMatrix(pred, test$survived)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 69 25
         1 10 42

               Accuracy : 0.7603
                 95% CI : (0.6827, 0.827)
    No Information Rate : 0.5411
    P-Value [Acc > NIR] : 3.612e-08

                  Kappa : 0.5089

 Mcnemar's Test P-Value : 0.01796

            Sensitivity : 0.8734
            Specificity : 0.6269
         Pos Pred Value : 0.7340
         Neg Pred Value : 0.8077
             Prevalence : 0.5411
         Detection Rate : 0.4726
   Detection Prevalence : 0.6438
      Balanced Accuracy : 0.7501

       'Positive' Class : 0
```

Though this output can’t be directly compared with logistic regression we can still learn a few things. We print the accuracy, specificity, and sensitivity elsewhere so let's focus on another value we don’t as much, the kappa. Kappa is a statistic that attempts to adjust accuracy by accounting for the possibility of a correct prediction by chance alone. Here the kappa is 0.5089, and as a kappa score from .4 to .6 means moderate agreement, this means the agreement between the predictions and actual values is moderately good, not perfect but not bad.

I also decided to use the head/tail functions on the data as well as the dim function just in case the model exploration doesn't count for data exploration functions. Here is the output and some discussion:

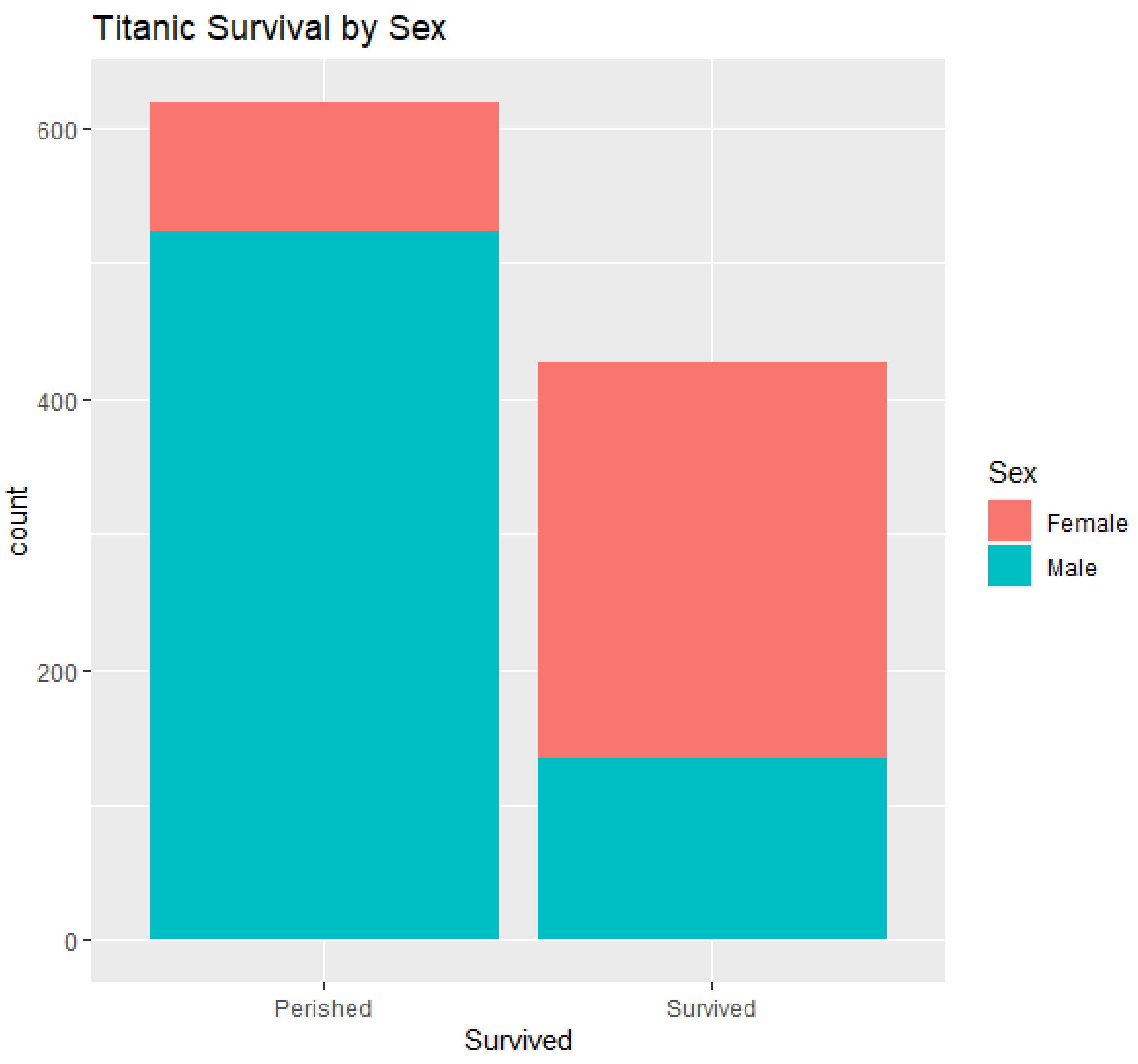


```
> head(ttnc)
    X pclass survived sex age
1 738      3        0   1  19
2 868      3        1   0  22
3 971      3        1   1  20
4 938      3        0   0   1
5 456      2        0   1  63
6 139      1        0   1  38
> tail(ttnc)
        X pclass survived sex age
1041  789      3        0   1  45
1042  407      2        0   1  40
1043 1131      3        0   0  18
1044  953      3        0   1  22
1045  432      2        0   1  28
1046  756      3        0   1  17
> dim(ttnc)
[1] 1046    5
```

The head and tail functions give a general idea of the instances in the data set. Here we see values for each column for each row and it gives us a good chance to see if there are any major mistakes/errors at the heads and tails of the data with unrealistic magnitude; luckily we don't see any here. Dim() is another function that gives us the dimensions of the data set; we see that it is 1046 instances by 5 rows.

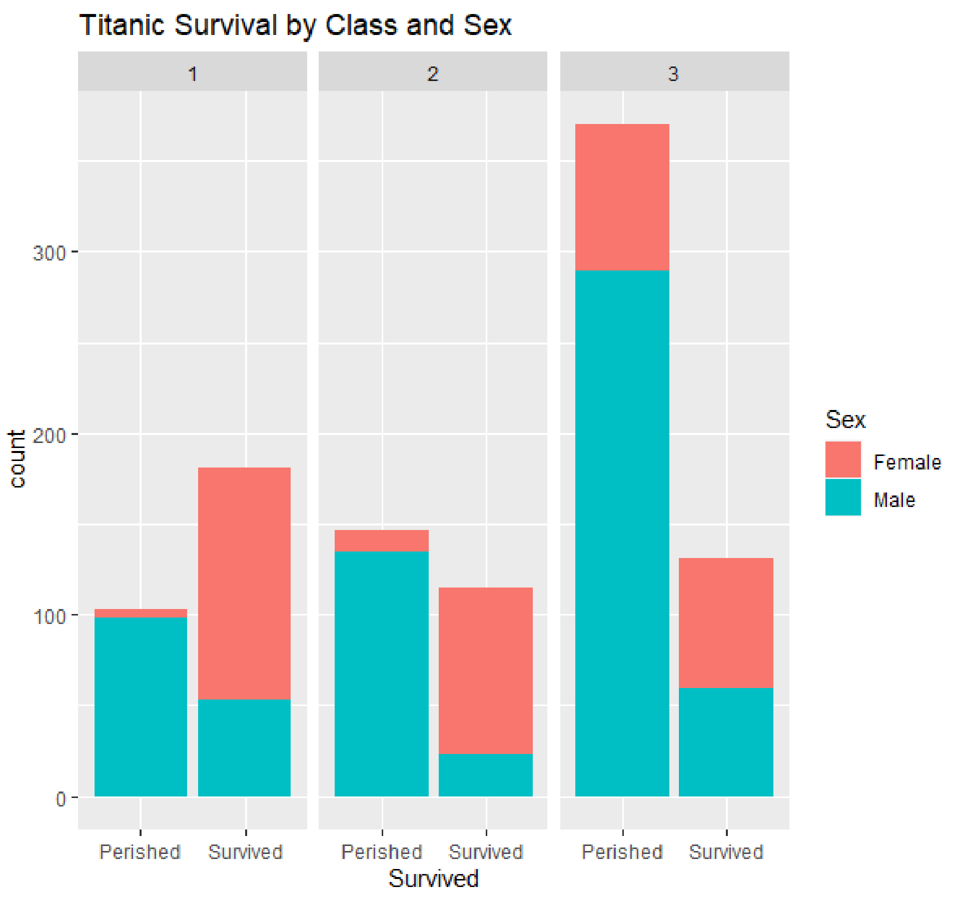
Data Exploration Graphs (4 graphs)

This project was a great opportunity to delve into the ggplot2 package to produce some beautiful visualizations. I was specifically drawn to the aesthetic colors and the ways to show information of combinations of variables within one graph very easily. Let’s see this in practice with the first graph, a simple graph that shows the counts of survival information but breaks it down even more by showing the proportion of how many people were male or female for each factor of survival:



With the ‘fill’ argument of ggplot2 we are able to glean even more information out of a simple bar; we see that a huge majority of the perished were male and that a slight majority of the survivors were female. This may be due to the fact that women and children were most likely taken to safety first amidst the crisis. We can also simply observe the counts of total perished and survived with the magnitude of the bars and sadly it looks that much more people died than survived. Just a side note, this graph was a great opportunity to learn how to manipulate the labels and legend as the original data was strictly numerical.

Next, I decided to delve even deeper by adding another feature to the combination: pclass, or passenger class. The facet_wrap argument allows for a multi-panel graph by passenger class, and by breaking it down this way, we are able to see the sex breakdown for each class rather than for just everyone:



With this visualization, we are better able to see not only how the survival counts vary by class but also how each class treats its women. Let's first break it down by class and then we'll look at any shared trends overall.

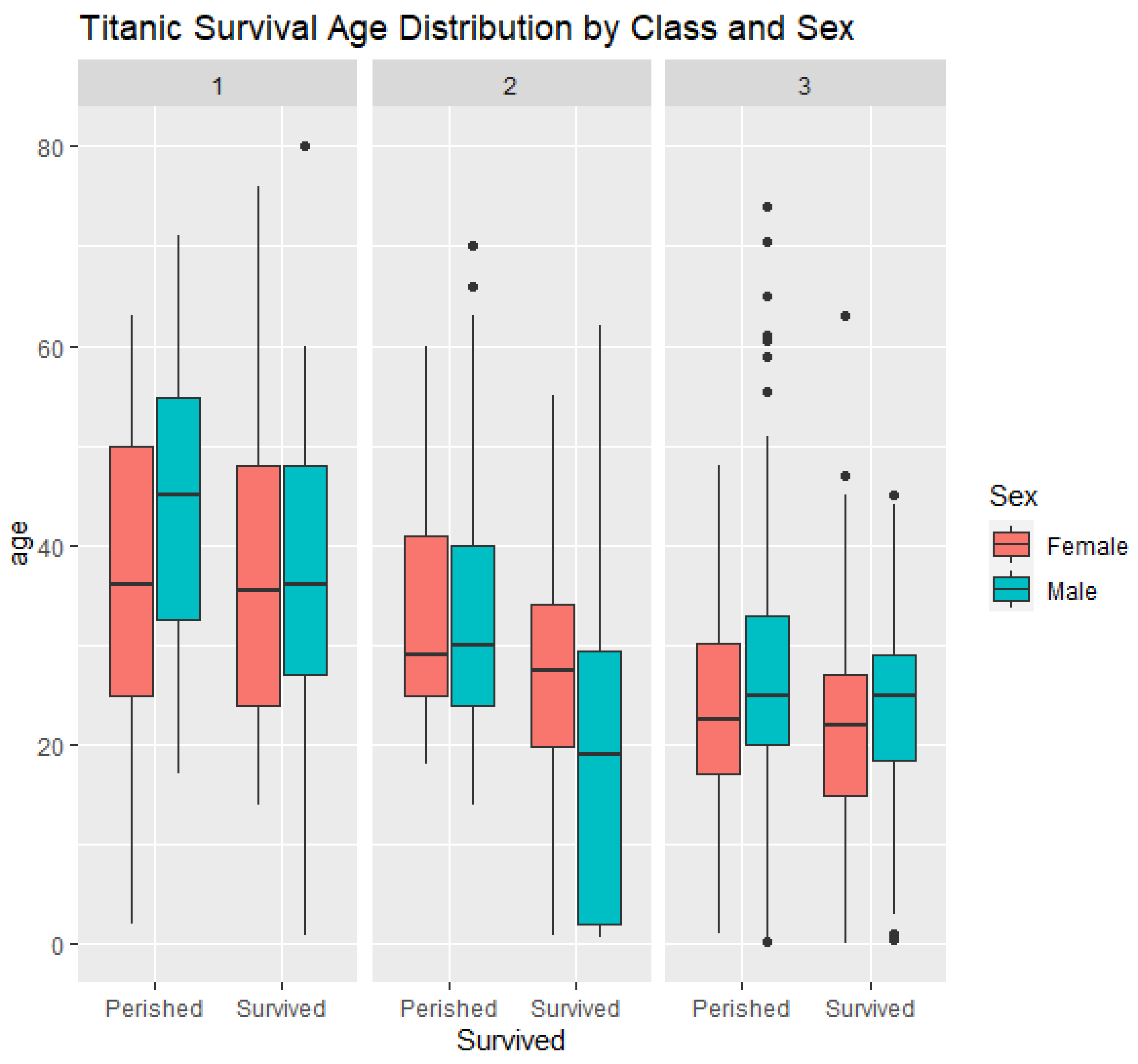
Passenger class 1, which is made of the wealthiest people, is interesting in that it is the only class where more people survived than perished; this may be due to the fact that they were treated more important because of their status and they could perhaps afford to have safer escape measures. Another interesting observation is that the gender disparity in survival is much larger here where almost all the perished were men, only a sliver women, and almost 60% of the survivors were women.

In the second class, we see something more in line with the overall trend that more people perished than survived, though the difference isn't large; they weren't as well off as first-class but definitely much better than the third class which we will observe next. We see a similar gender disparity in survival here as well with much more of the survivors being women and perished male.

Finally, in the third passenger class, we see that much of the overall survival trend has its weight from here; this class has the most number of people, carrying the most influence on the overall population data. We see a great difference in survival here with almost three times as many people dying than surviving. This may be due to not many people prioritizing the lives of "lower class" people and perhaps even that these people may have had to help the others get off first before them. Again, even though about an equal number of females die and survive, we see that a large proportion of the perished were men and more of them who survived were female than male but the proportions are very close. This speaks to the socioeconomic inequality that these people sadly had to face.

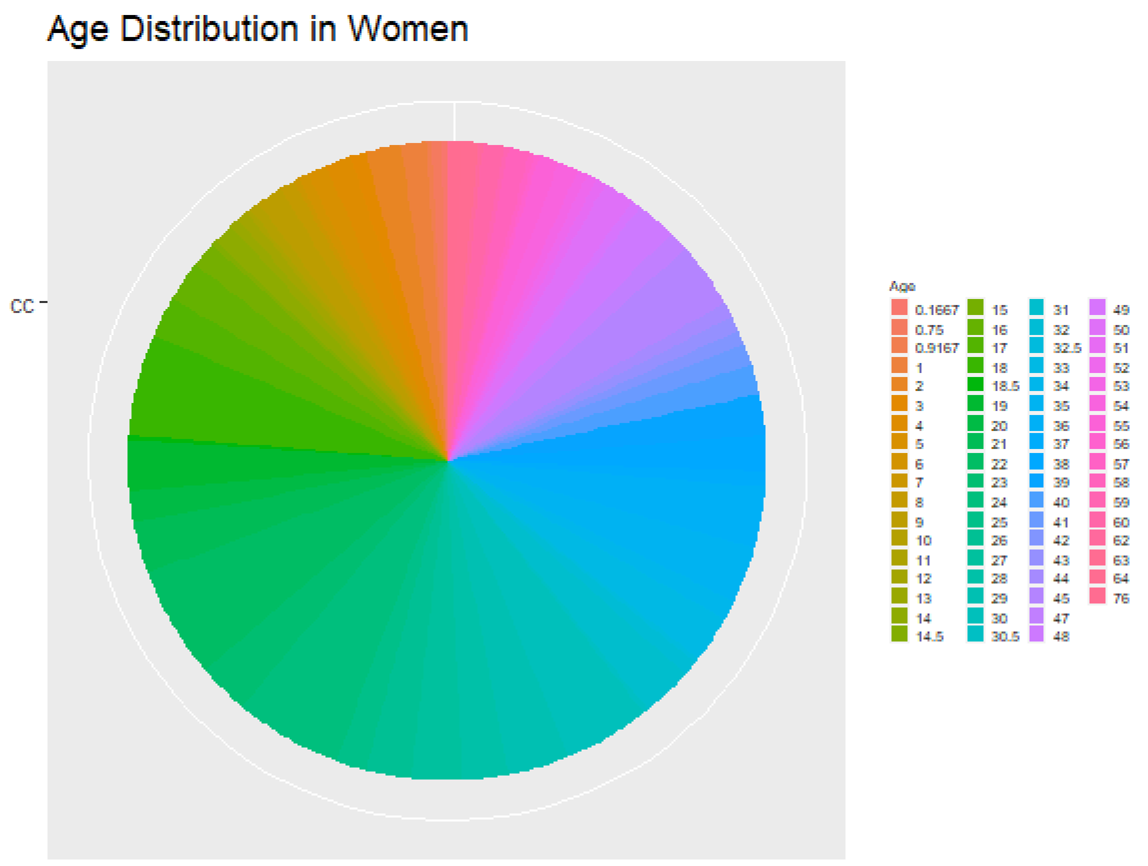
So the biggest takeaways from the part of exploration are that the largest passenger class aboard the titanic was third class and that they were the worst off whereas the first class was best off.

Next, I decided to explore the last predictor in our model, age. It was a little difficult at first to figure out how to include it with all the other variables to create a meaningful plot both because of the number of variables as well as age being a continuous variable whereas the others were discrete, however, I realized I could use this to focus on the distribution of ages in survival:



There are many aspects to explore in this one graph but let’s look at a few key ideas. First, we see that the perished are generally older than the survived in all three classes. However, if we look a little closer in regards to sex, though that observation holds true for men, as the median value of men who passed away are high for each class, though closer for the third class, the women’s median age is very close whether they died or survived. This may be due to the fact that all women were treated with care no matter the age so they died or survived similarly. Coming back to the third class, we see the median ages for both genders for perished and survival is very close; perhaps this is the case because they were treated equally badly because of their class. Overall, we also see that the first class is generally older and the second and third are younger and younger.

Finally, there wasn't another graph I could think of that gave much different information from the previous in-depth graph so I decided to explore a format we haven't worked with much, the pie chart. All of the factors, except age, were discrete factors so most of the variations I tried with the pie chart weren't very unique or useful; instead, I decided to be a little creative and explore the age distribution within just the women. Though other types of graphs are probably more useful, I learned a lot about subsetting, legend manipulation, pie chart creation, and aesthetically the graph looks really cool!

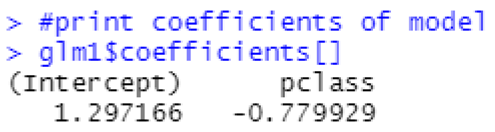


We see that a majority of women are quite young, from about 15 to 45, and the second biggest group being middle-aged to slightly old, from 45-75. Small children make a very small part of total women.

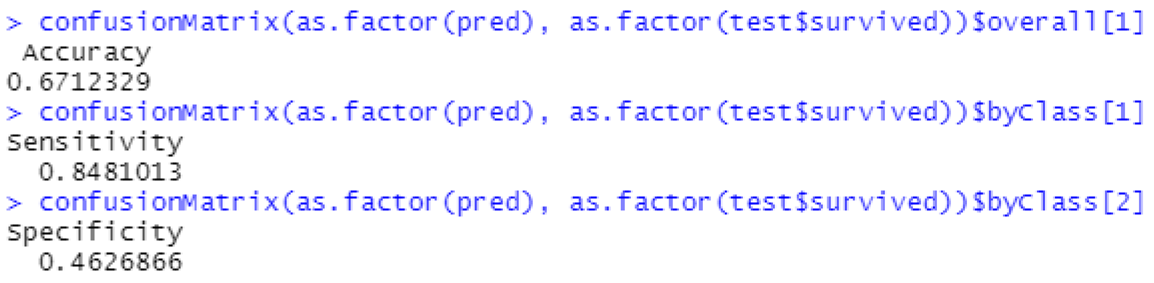
Logistic Regression Comparison

Logistic regression was deceivingly simple to implement using the R function glm() but once I started from scratch for C++ I was able to better understand how complicated the pieces at play are. My first attempt at implementation had me in a complicated mess of vectors, incorrect matrix multiplication, and various other errors with libraries I tried to use; instead I finished naive bayes first and came back to it. When I started from scratch the second time through, I learned that arrays may help with some runtime issues and after sitting down and writing out the math, I realized the matrix multiplication was quite simple and really involved multiplying one array but another. So overall, C++ was more complicated to understand but the process was actually quite simple.

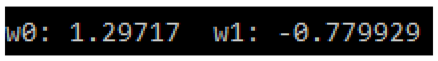
I received the same results with C++ as I did with R. The intercept was 1.29717 and the pclass coefficient -0.779929. I also had the same metrics for accuracy, sensitivity, and specificity: 0.671233, 0.848101, and 0.462687 respectfully. The output for both is below:



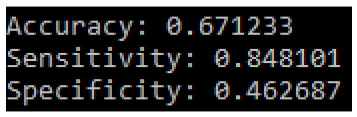
```
> #print coefficients of model
> glm1$coefficients[]
(Intercept)      pclass
   1.297166   -0.779929
```



```
> confusionMatrix(as.factor(pred), as.factor(test$survived))$overall[1]
 Accuracy
0.6712329
> confusionMatrix(as.factor(pred), as.factor(test$survived))$byClass[1]
Sensitivity
  0.8481013
> confusionMatrix(as.factor(pred), as.factor(test$survived))$byClass[2]
Specificity
  0.4626866
```

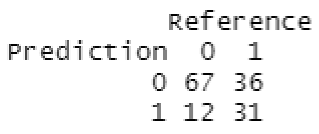


```
w0: 1.29717  w1: -0.779929
```

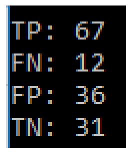


```
Accuracy: 0.671233
Sensitivity: 0.848101
Specificity: 0.462687
```

The confusion matrix values matched up as well when double checking:

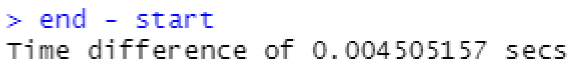


```
          Reference
Prediction  0  1
         0 67 36
         1 12 31
```

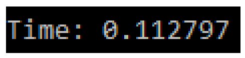


```
TP: 67
FN: 12
FP: 36
TN: 31
```

As for the run times, logistic regression ran much faster using R's built in optimization methods than in C++: 0.004505157 vs 0.112797 seconds. Time was measured using the Sys.time() in R and the chrono library in C++. It was only around the glm() function in R and around sigmoid/error/weight iterations in C++. Output below:



```
> end - start
Time difference of 0.004505157 secs
```

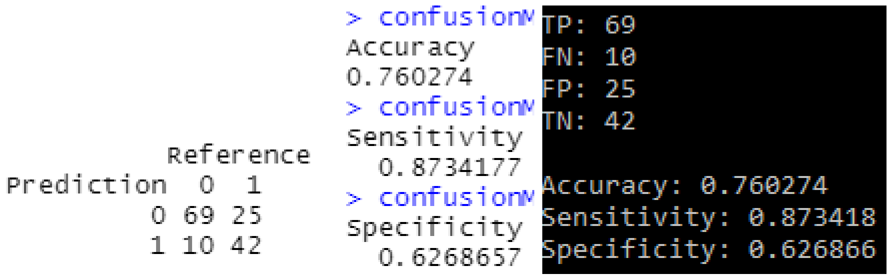


```
Time: 0.112797
```

Naive Bayes Comparison

The naive bayes algorithm was similarly simple to implement in R with the naiveBayes() function. The C++ implementation was more complicated but relatively much simpler than logistic regression as it was mostly just probability calculations. It wasn't too hard to understand but just had to be careful with indexing and remembering the orientation of the likelihood matrices; writing stuff out helped here as well.

I received the same results with C++ as I did with R here as well: with an accuracy, sensitivity, and specificity of 0.760274, 0.873418, and 0.626866 for both. I also double checked my prediction values with a confusion matrix.

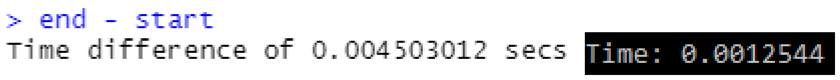


```
          Reference
Prediction  0  1
         0 69 25
         1 10 42
```

```
> confusionM
Accuracy
0.760274
> confusionM
Sensitivity
  0.8734177
> confusionM
Specificity
  0.6268657
```

```
TP: 69
FN: 10
FP: 25
TN: 42

Accuracy: 0.760274
Sensitivity: 0.873418
Specificity: 0.626866
```

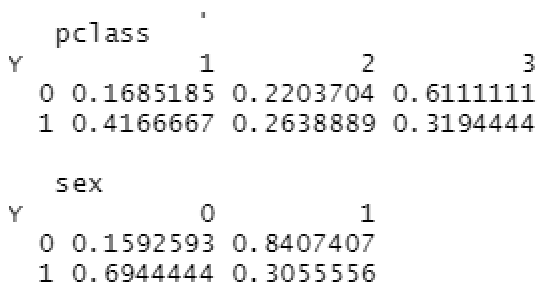
Time was measured with the same function here as well. For the C++ implementation, chrono was started with the calculation of the apriori probabilities and finished at the calculation of variance for the last variable, age, as that is the last piece before prediction. Interestingly, in this case, C++ was actually much fast than R with 0.0012544 vs 0.004503012 seconds.



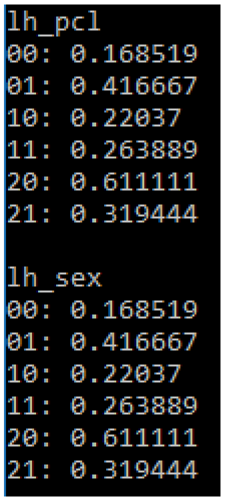
```
> end - start
Time difference of 0.004503012 secs
```

```
Time: 0.0012544
```

I also printed the conditional probabilities to verify between R and C++:



```
   pclass
Y          1         2         3
  0 0.1685185 0.2203704 0.6111111
  1 0.4166667 0.2638889 0.3194444

   sex
Y          0         1
  0 0.1592593 0.8407407
  1 0.6944444 0.3055556
```



```
lh_pcl
00: 0.168519
01: 0.416667
10: 0.22037
11: 0.263889
20: 0.611111
21: 0.319444

lh_sex
00: 0.168519
01: 0.416667
10: 0.22037
11: 0.263889
20: 0.611111
21: 0.319444
```