Wine Quality Prediciton

William Lovejoy

3/4/2022

Data Loading and Exploration

Data was accquired from https://www.kaggle.com/yasserh/wine-quality-dataset. As always, load it in, and look at the head and types of data in the data set. I prefer using glimpse() as it lets me see the column name, data type, and some of the data. Lastly, I want to know what variables are the biggest determinants in percieved wine quality. So I want a general idea of how the wine quality scores are distributed

```
df <- read.csv("C:\\Users\\William Lovejoy\\Documents\\Codes\\R\\DataScience\
\Wine\\WineQT.csv")
head(df)
```

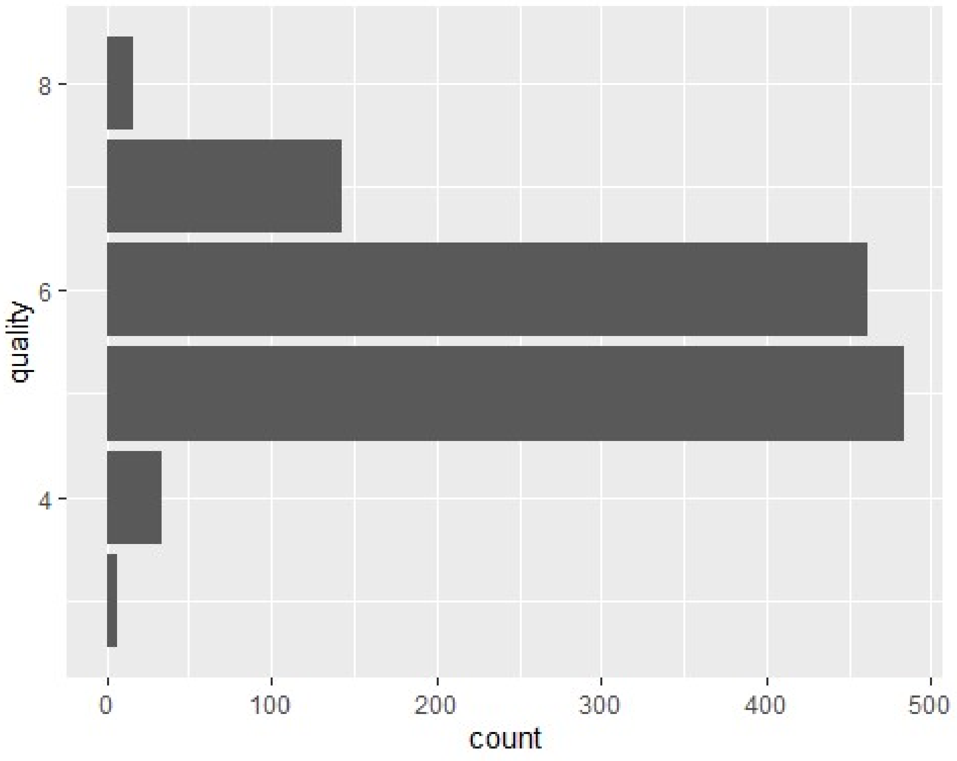
```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality Id
## 1       5  0
## 2       5  1
## 3       5  2
## 4       6  3
## 5       5  4
## 6       5  5
```

```
glimpse(df)
```

```
## Rows: 1,143
## Columns: 13
## $ fixed.acidity        <dbl> 7.4, 7.8, 7.8, 11.2, 7.4, 7.4, 7.9, 7.3, 7.8,
6.7~
## $ volatile.acidity     <dbl> 0.700, 0.880, 0.760, 0.280, 0.700, 0.660, 0.6
00, ~
```

```
## $ citric.acid          <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.06, 0.0
0, 0~
## $ residual.sugar       <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2, 2.0,
1.8,~
## $ chlorides            <dbl> 0.076, 0.098, 0.092, 0.075, 0.076, 0.075, 0.0
69, ~
## $ free.sulfur.dioxide  <dbl> 11, 25, 15, 17, 11, 13, 15, 15, 9, 15, 16, 9,
35,~
## $ total.sulfur.dioxide <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 65, 59, 2
9, 1~
## $ density              <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978, 0.997
8, 0~
## $ pH                   <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.30, 3.3
9, 3~
## $ sulphates            <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.46, 0.4
7, 0~
## $ alcohol              <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, 10.0, 9.5,
9.2~
## $ quality              <int> 5, 5, 5, 6, 5, 5, 5, 7, 7, 5, 5, 5, 7, 6, 5,
5, 5~
## $ Id                   <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 13, 16, 19
, 21~
```

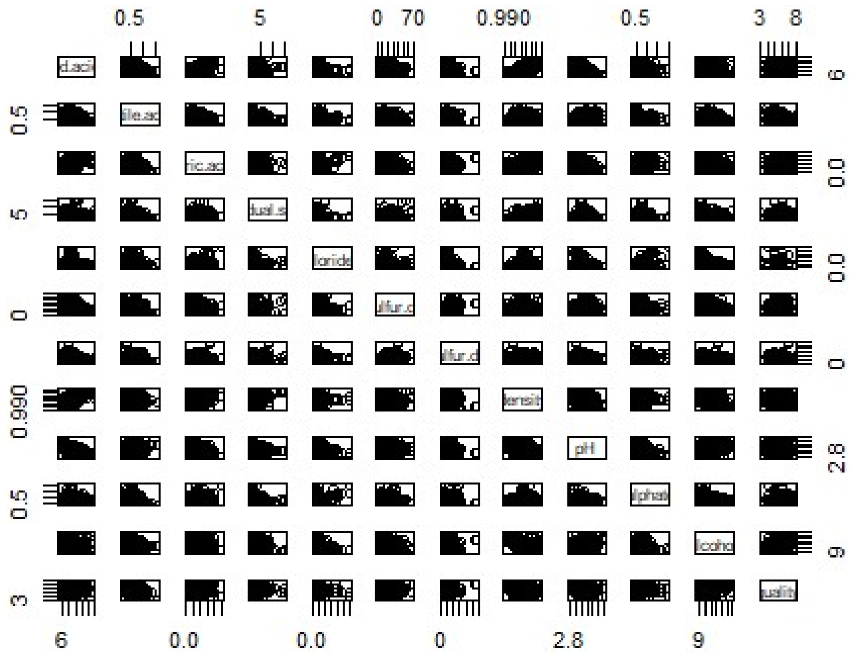
```
ggplot(df, aes(y = quality)) + geom_bar()
```



Mostly 5's and 6's. So let's look at overall relationship between the variables. To start with, we'll make a simple scatterplot of all the variables against each other (with the exception of the ID category)

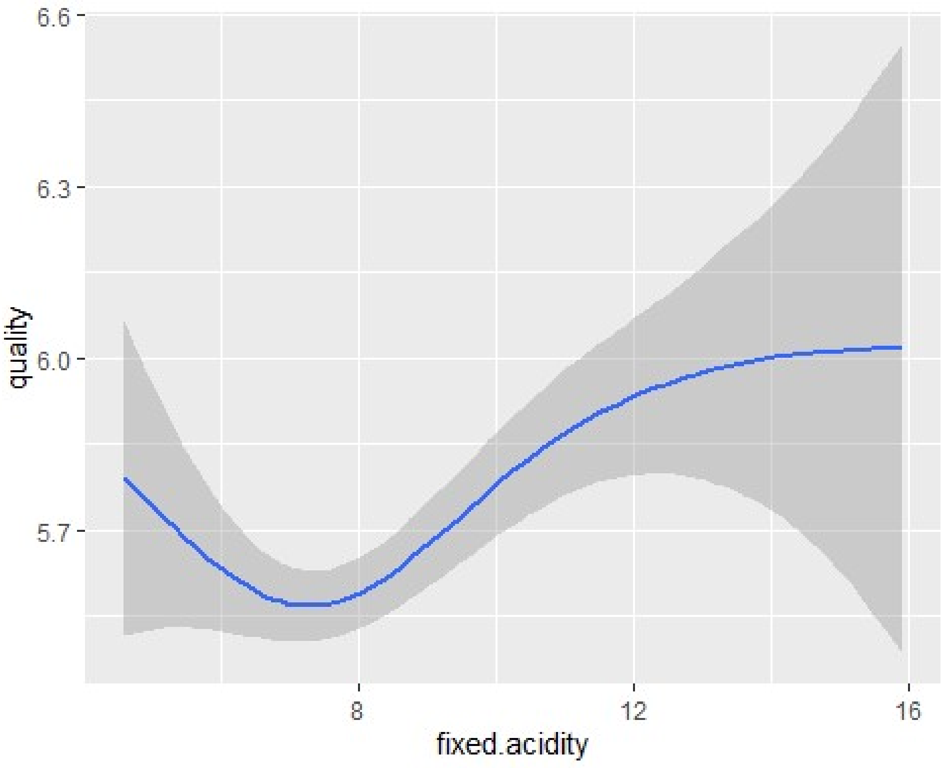
Plots

```
plot(df[,-c(13)])
```

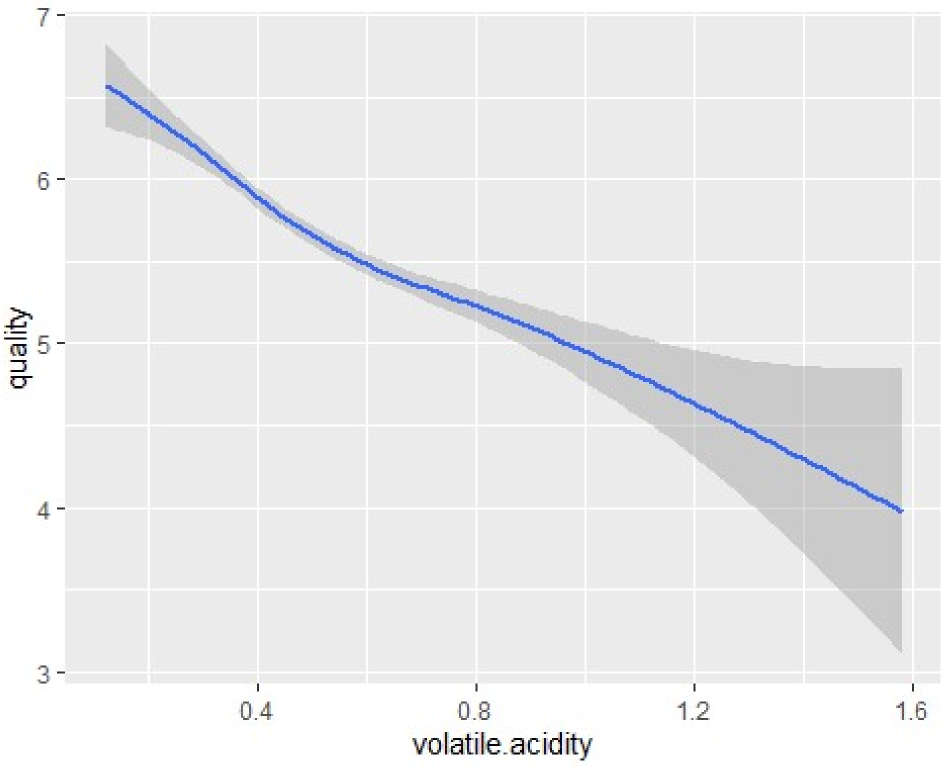


Quality seems to have a positive relationship with alcohol, and a negative relationship with volatile acidity. So lets look at those in more detail later. For now though, we can make quick smooth plots for all of the variables vs. quality.

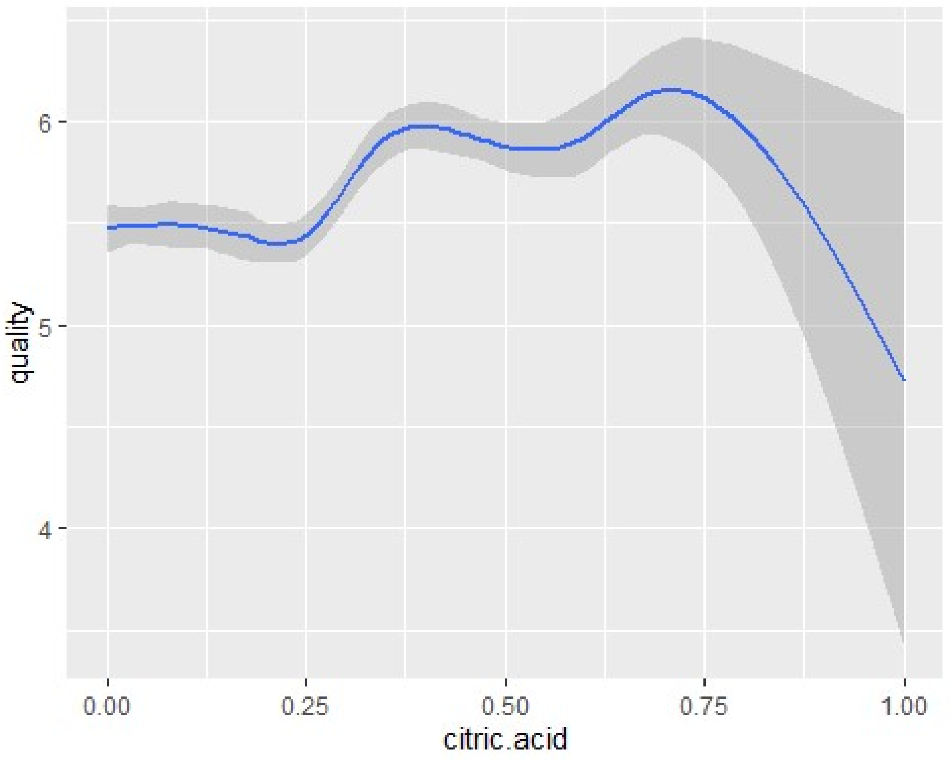
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



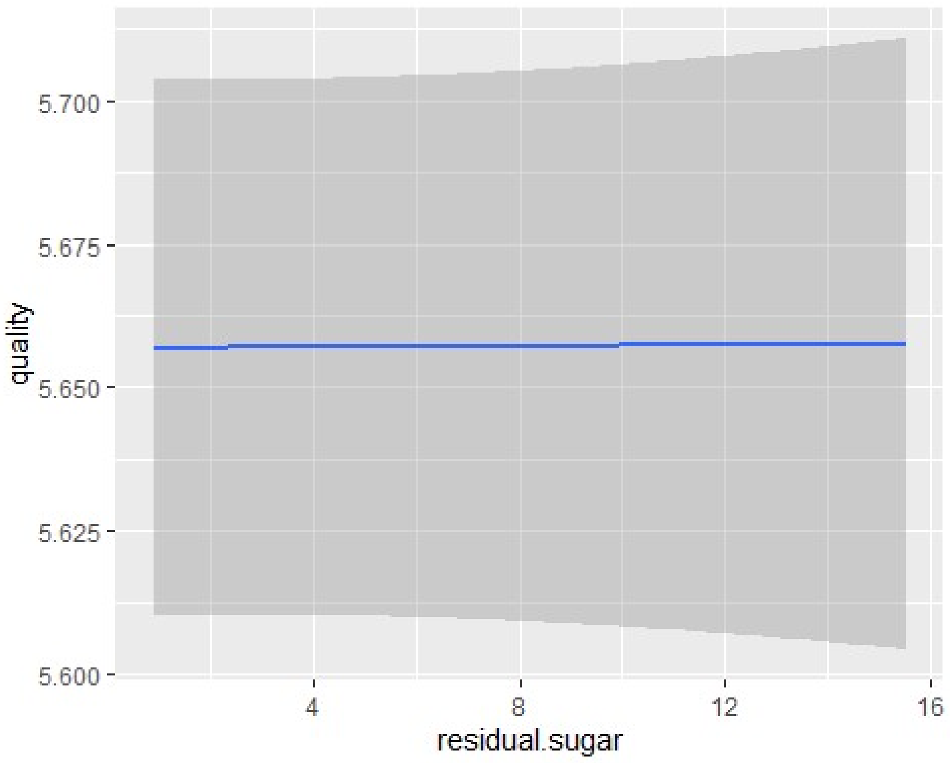
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



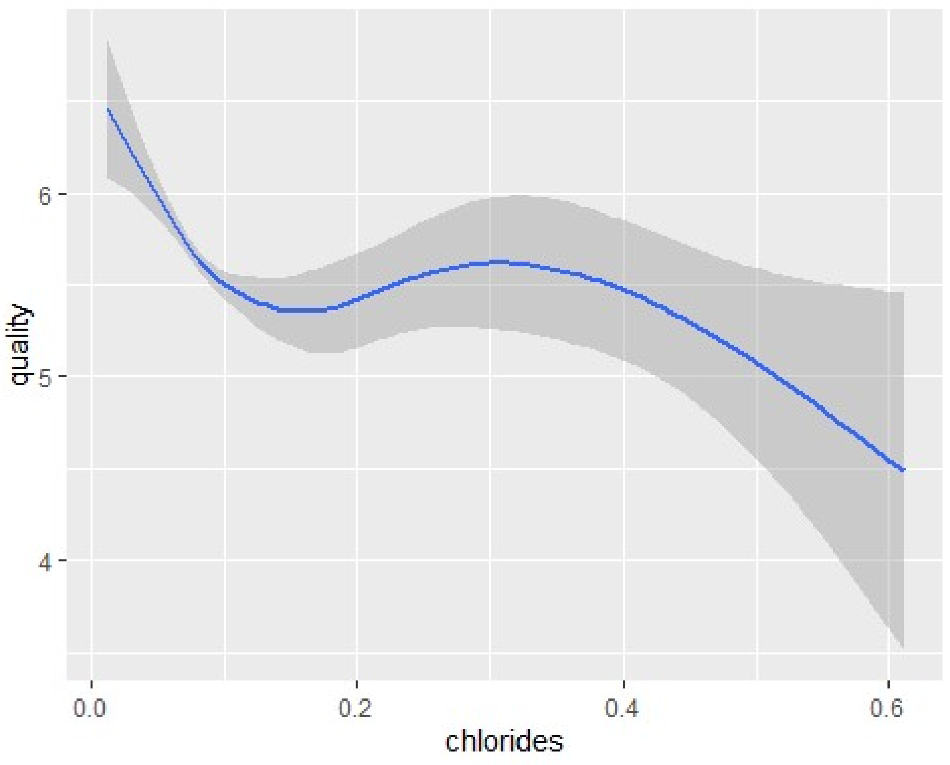
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



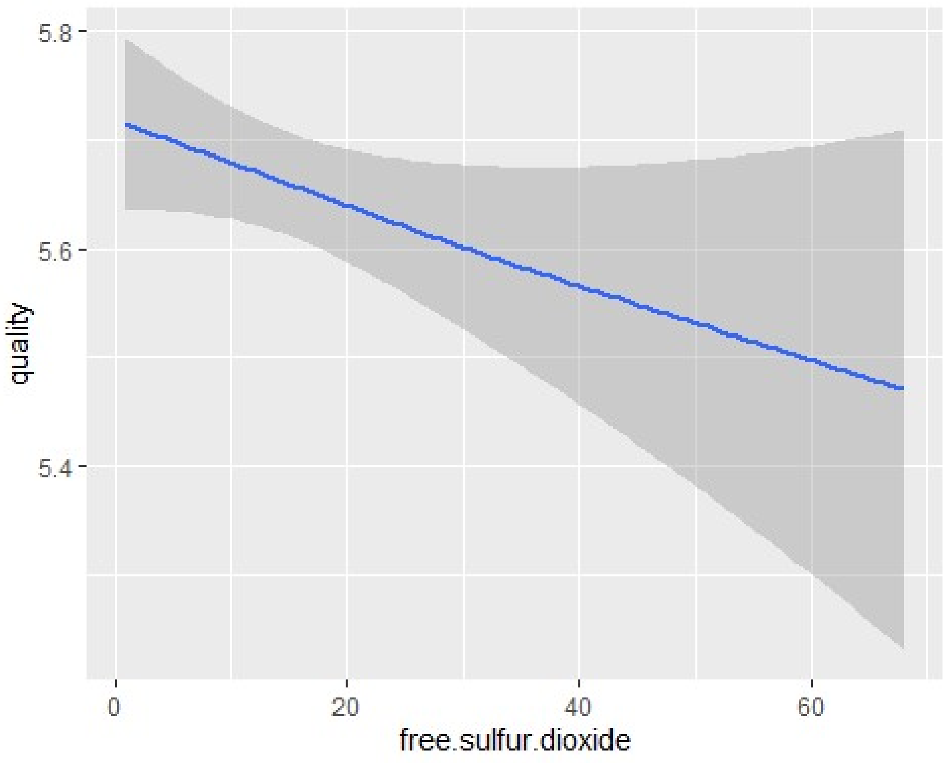
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



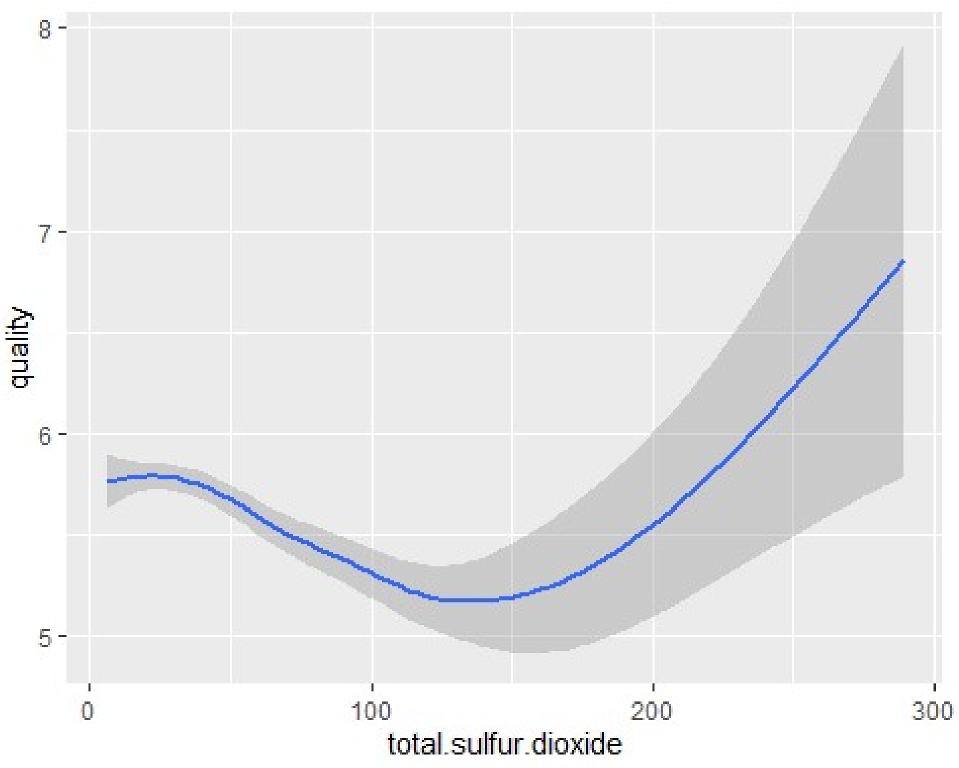
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



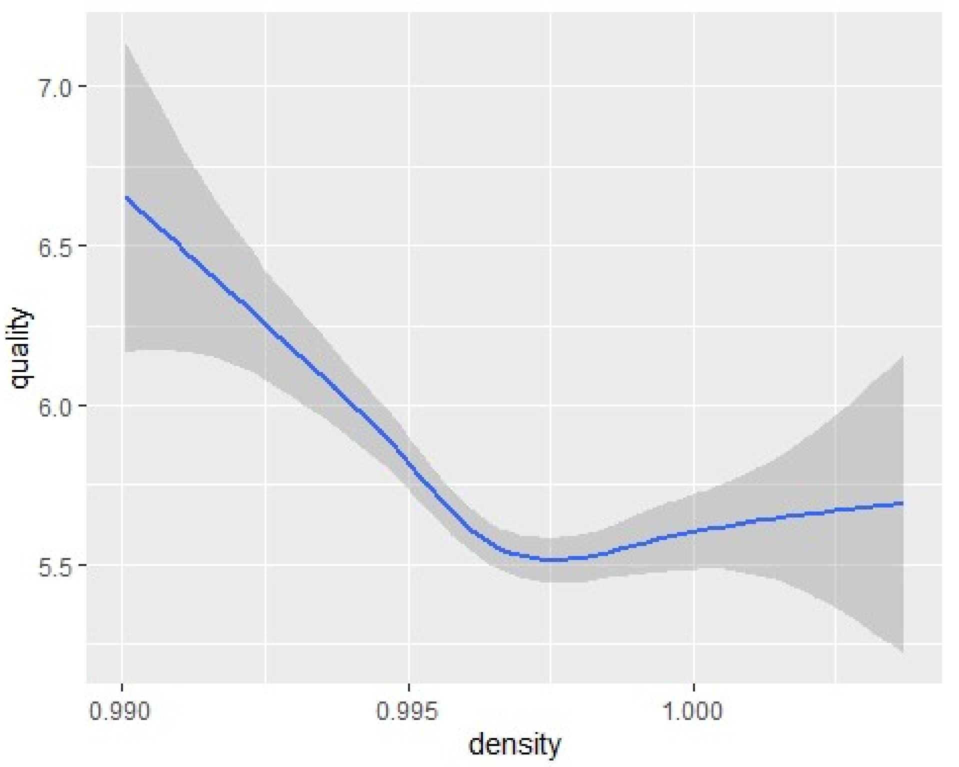
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



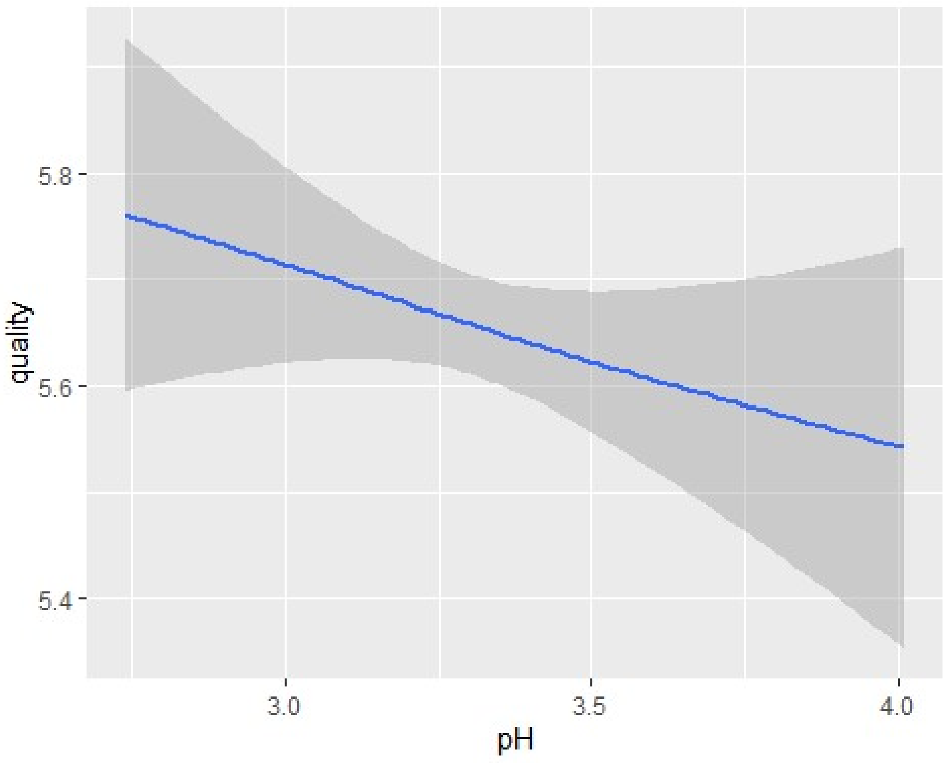
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



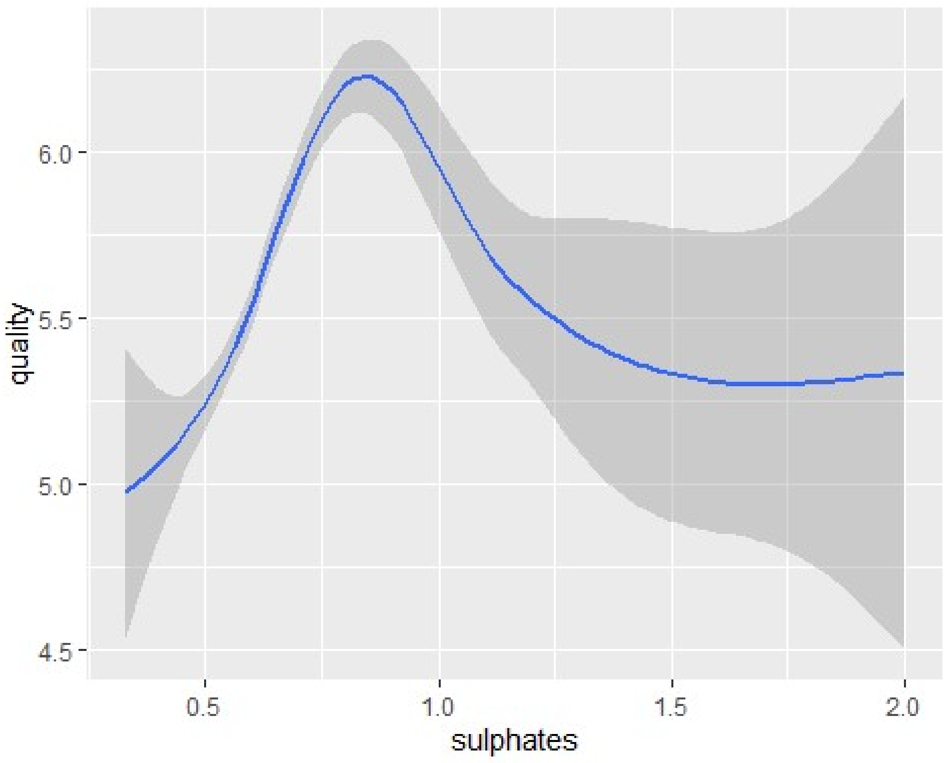
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



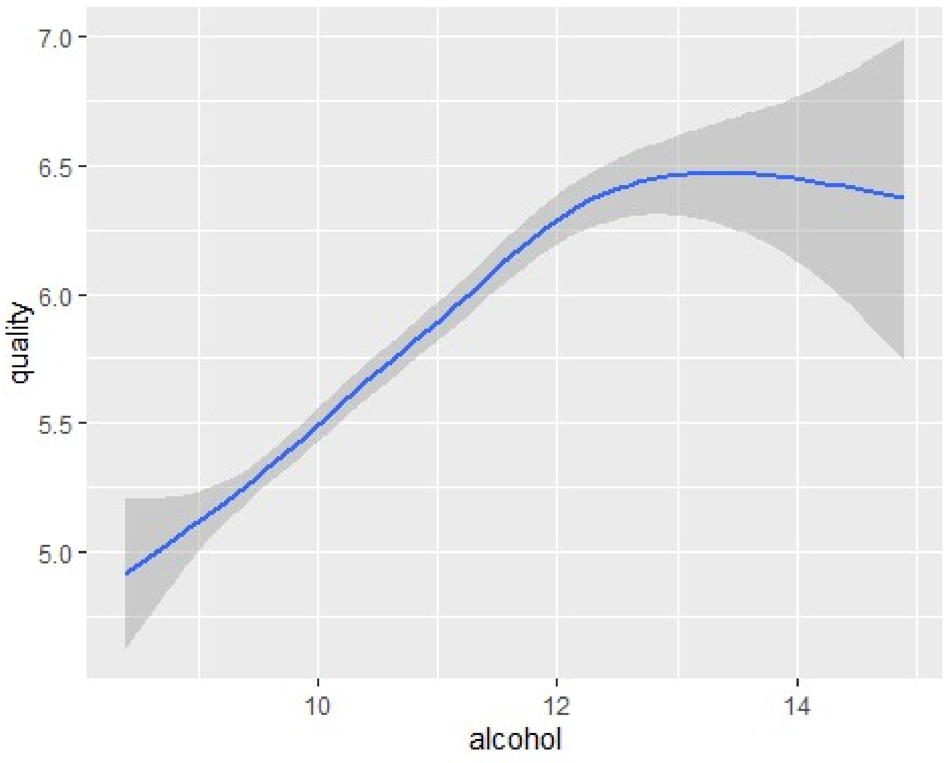
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



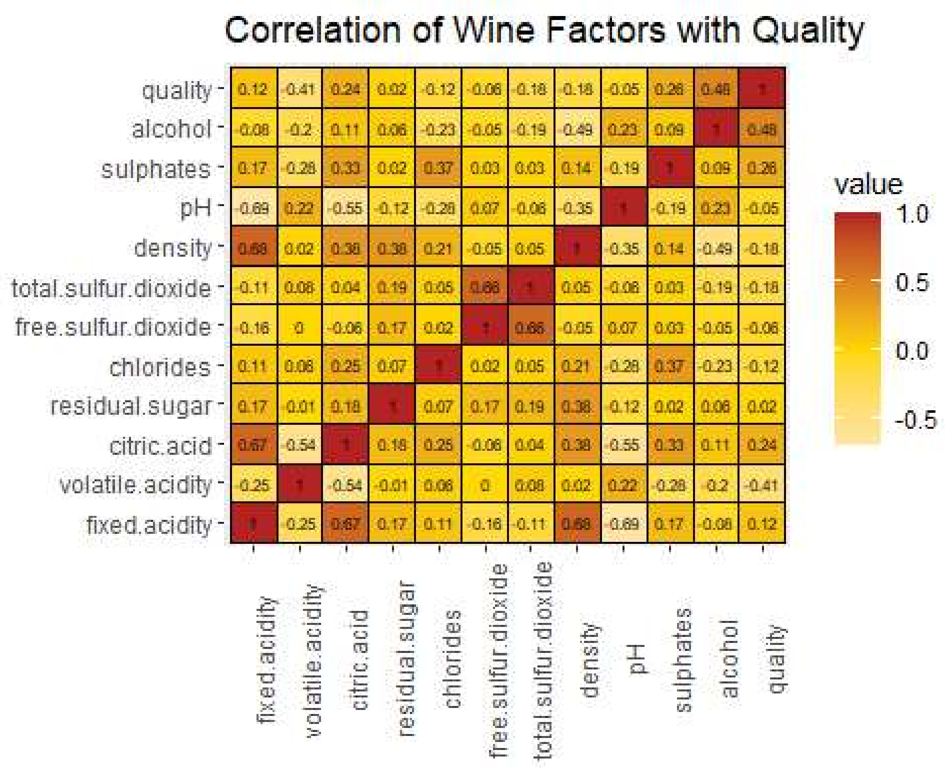
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



As expected, strong negative association with volatile acidity and a strong positive associatin with alcohol. There are some other variables that lead to a fairly positive association at least within certain ranges. Citric acid levels between 0.37 and 0.75 have higher quality scores, and sulphate values correlate positively with quality if they're under 0.8.

Now we that we have the basic exploration done, we can make a more complex chart. We'll start by calculating correlation valeus for all the variables except for ID, and then use melt() to reshape the data frame into just 3 columns (variable 1, variable 2, and correlation score)

```
correlated <- df[, -c(13)] %>% #column 13 is the ID numbers, so we'll drop th
at here.
  cor() %>%
  melt()

ggplot(correlated, aes(x = Var1, y = Var2, fill = value)) + scale_fill_gradie
nt2(low = "antiquewhite",

mid = "gold",

high = "firebrick") +
  geom_tile(color = "black") +
  geom_text(aes(label = round(value, 2)), size = 2) +
  theme(axis.text.x = element_text(angle = 90), axis.title = element_blank())
+
  labs(title = "Correlation of Wine Factors with Quality")
```



Just like before, but now a bit more clear. Quality is positively correlated with alcohol, sulphates, citric acid, and fixed acidity. It also has a noticeable negative correlation with volatile acidity.

Now that we know what sort of data is important, we can try and train a model to predict important variables. As this is my first foray into machine learning. The goal is to predict the quality of wine based on the listed variables using a Random Forest model. I'm using Random Forest due to it's accuracy, ease of use, and how well it fits with my data and goals (prediction with regression)

I set the seed so the data will be reproducible, and convert my quality scores into factors. I then split my data into it's train and testing groups (80/20), so everything will be ready to go.

```
set.seed(100) #Set the seed so that the data is reproducible
df$quality <- as.factor(df$quality) #Make the quality scores a factor
ind <- sample(2, nrow(df), replace = TRUE, prob = c(0.8, 0.2)) #Randomly assi
gn indexes with ~80% as 1 (training data)
train <- df[ind == 1,] #training data is everything with an index of 1
test <- df[ind == 2,] #testing data is everything with an index of 2
```

Now I run my random forest, and get an error rate of 34.89%. That means that my model has about 66.96% accuracy. I'll also make my prediction and confusion matrices here.

```
forest <- randomForest(data = train[,-c(13)], quality ~ ., proximity = TRUE)
#runs the data through the random forest model
print(forest)
```

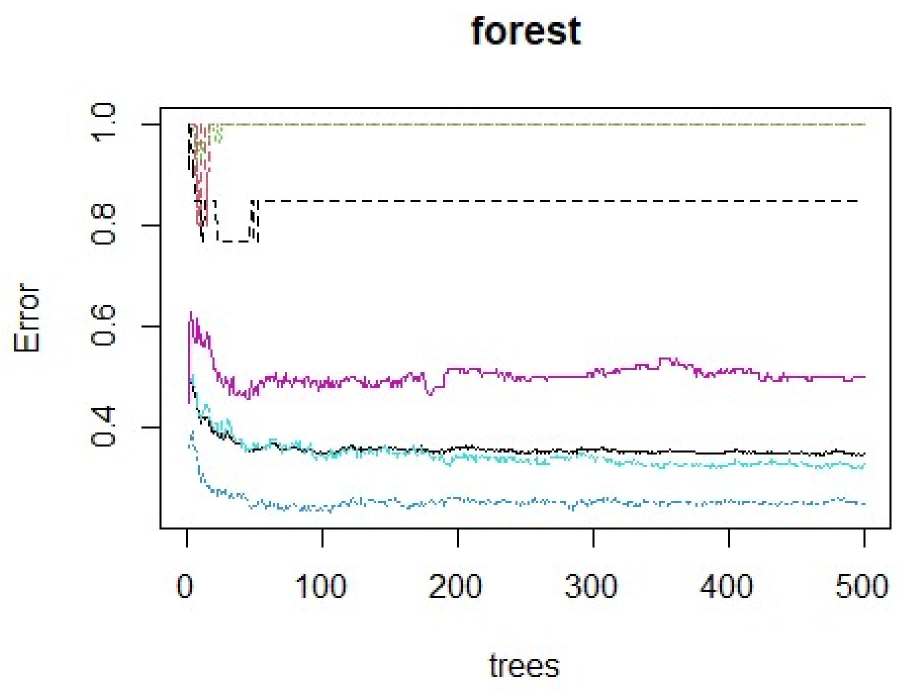
```
##
## Call:
##  randomForest(formula = quality ~ ., data = train[, -c(13)], proximity = T
RUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 34.89%
## Confusion matrix:
##   3 4   5   6  7 8 class.error
## 3 0 0   4   1  0 0   1.0000000
## 4 0 0  22   6  0 0   1.0000000
## 5 0 0 285  90  5 0   0.2500000
## 6 0 1  94 241 22 1   0.3286908
## 7 0 0   5  51 56 0   0.5000000
## 8 0 0   0   8  3 2   0.8461538
```

```
#Confusion and Prediction Matrices
p1 <- predict(forest, train)
print("Training Matrices")
```

```
## [1] "Training Matrices"
```

```
confusionMatrix(p1, train$quality)
```

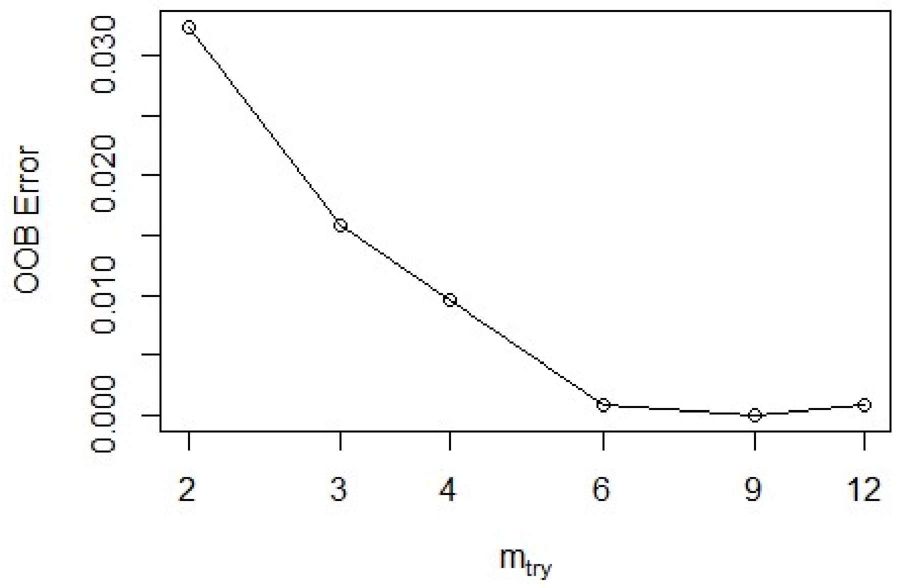
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8
##          3   5   0   0   0   0   0
##          4   0  28   0   0   0   0
##          5   0   0 380   0   0   0
##          6   0   0   0 359   0   0
##          7   0   0   0   0 112   0
##          8   0   0   0   0   0  13
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9959, 1)
##     No Information Rate : 0.4236
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          1.000000 1.00000   1.0000   1.0000   1.0000  1.00000
## Specificity          1.000000 1.00000   1.0000   1.0000   1.0000  1.00000
## Pos Pred Value       1.000000 1.00000   1.0000   1.0000   1.0000  1.00000
## Neg Pred Value       1.000000 1.00000   1.0000   1.0000   1.0000  1.00000
## Prevalence           0.005574 0.03122   0.4236   0.4002   0.1249  0.01449
## Detection Rate       0.005574 0.03122   0.4236   0.4002   0.1249  0.01449
## Detection Prevalence 0.005574 0.03122   0.4236   0.4002   0.1249  0.01449
## Balanced Accuracy    1.000000 1.00000   1.0000   1.0000   1.0000  1.00000

p2 <- predict(forest, test)
print("Testing Matrices")

## [1] "Testing Matrices"

confusionMatrix(p2, test$quality)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
##          4  0  0  0  0  0  0
##          5  1  3 84 27  1  0
##          6  0  2 19 72 14  1
##          7  0  0  0  4 16  2
##          8  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.6992
##                  95% CI : (0.6377, 0.7558)
##     No Information Rate : 0.4187
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5048
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000 0.00000   0.8155   0.6990  0.51613   0.0000
## Specificity          1.000000 1.00000   0.7762   0.7483  0.97209   1.0000
## Pos Pred Value            NaN      NaN   0.7241   0.6667  0.72727      NaN
## Neg Pred Value       0.995935 0.97967   0.8538   0.7754  0.93304   0.9878
## Prevalence           0.004065 0.02033   0.4187   0.4187  0.12602   0.0122
## Detection Rate       0.000000 0.00000   0.3415   0.2927  0.06504   0.0000
## Detection Prevalence 0.000000 0.00000   0.4715   0.4390  0.08943   0.0000
## Balanced Accuracy    0.500000 0.50000   0.7959   0.7236  0.74411   0.5000

plot(forest)
```



Our training accuracy was a 1, so everything trained correctly. Going ahead and running the test gives us an accuracy of 0.6992 or ~69.92%. This is a bit lower than I was expecting, so I'll try and fine tune the forest model. The goal here is to find the best value to input as mtry in my random forest model. This value determines how many variables it attempts at each for in the pathway

```
mtry <- tuneRF(df[, -c(13)], df$quality, ntreeTry=500,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
```

```
## mtry = 3  OOB error = 1.57%
## Searching left ...
## mtry = 2     OOB error = 3.24%
## -1.055556 0.01
## Searching right ...
## mtry = 4     OOB error = 0.96%
## 0.3888889 0.01
## mtry = 6     OOB error = 0.09%
## 0.9090909 0.01
## mtry = 9     OOB error = 0%
## 1 0.01
## mtry = 12    OOB error = 0.09%
## -Inf 0.01
```



```
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry)

##         mtry     OOBError
## 2.OOB      2 0.0323709536
## 3.OOB      3 0.0157480315
## 4.OOB      4 0.0096237970
## 6.OOB      6 0.0008748906
## 9.OOB      9 0.0000000000
## 12.OOB    12 0.0008748906

print(best.m)

## [1] 9
```

mtry = 9 seems to be golden. It has an OOB of 0. So we'll run our random forest again with an mtry of 9. We'll also look at how important each variable is when it comes to calculating the quality of wine. A higher MeanDecreaseGini means that the variable is more important when calculating the quality of a wine. Alcohol is the most important variable, followed by volatile acidity, sulphates, and total sulphur dioxide. This reflects our tile plot from the beginning.

```
forest2 <- randomForest(data = train[,-c(13)], quality ~ ., proximity = TRUE,
mtry = 9) #runs the data through the random forest model
print(forest2)
```

```
##
## Call:
##  randomForest(formula = quality ~ ., data = train[, -c(13)], proximity = T
RUE,      mtry = 9)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 36.23%
## Confusion matrix:
##   3 4   5   6  7 8 class.error
## 3 0 0   3   2  0 0   1.0000000
## 4 0 0  20   8  0 0   1.0000000
## 5 0 0 284  89  7 0   0.2526316
## 6 0 1  97 233 27 1   0.3509749
## 7 0 0   8  51 53 0   0.5267857
## 8 0 0   0   6  5 2   0.8461538
```

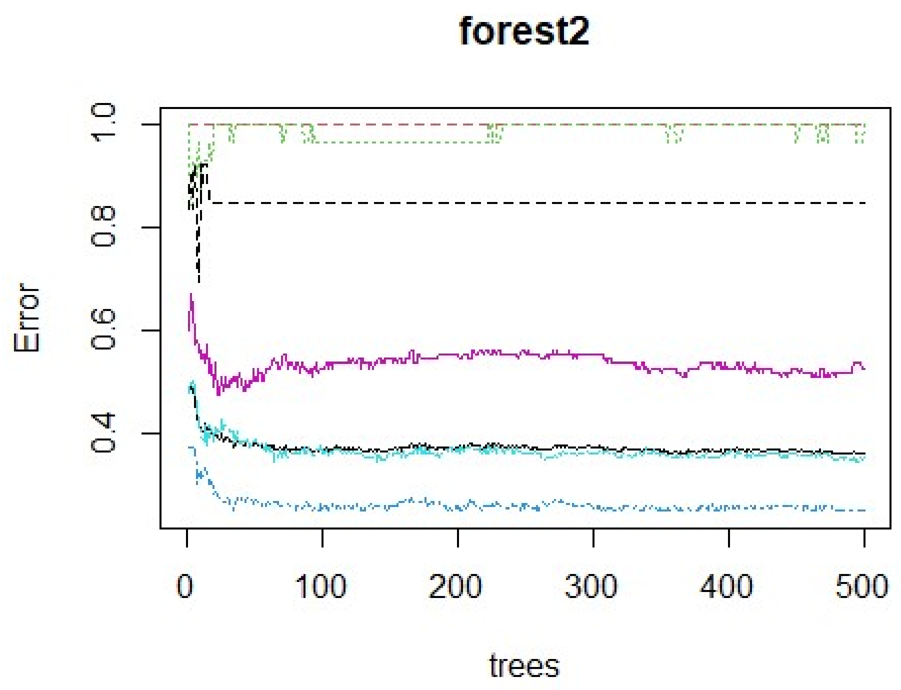
```
#Matrices
p12 <- predict(forest2, train)
confusionMatrix(p12, train$quality)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8
##          3   5   0   0   0   0   0
##          4   0  28   0   0   0   0
##          5   0   0 380   0   0   0
##          6   0   0   0 359   0   0
##          7   0   0   0   0 112   0
##          8   0   0   0   0   0  13
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9959, 1)
##     No Information Rate : 0.4236
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          1.000000  1.00000   1.0000   1.0000   1.0000  1.00000
## Specificity          1.000000  1.00000   1.0000   1.0000   1.0000  1.00000
## Pos Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000  1.00000
## Neg Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000  1.00000
## Prevalence           0.005574  0.03122   0.4236   0.4002   0.1249  0.01449
## Detection Rate       0.005574  0.03122   0.4236   0.4002   0.1249  0.01449
## Detection Prevalence 0.005574  0.03122   0.4236   0.4002   0.1249  0.01449
## Balanced Accuracy    1.000000  1.00000   1.0000   1.0000   1.0000  1.00000
```

```
p22 <- predict(forest2, test)
confusionMatrix(p22, test$quality)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
##          4  0  0  0  0  0  0
##          5  1  3 84 26  1  0
##          6  0  2 19 73 11  1
##          7  0  0  0  3 19  2
##          8  0  0  0  1  0  0
##
## Overall Statistics
##
##                Accuracy : 0.7154
##                  95% CI : (0.6547, 0.771)
##     No Information Rate : 0.4187
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5347
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000  0.00000   0.8155   0.7087  0.61290 0.000000
## Specificity          1.000000  1.00000   0.7832   0.7692  0.97674 0.995885
## Pos Pred Value            NaN      NaN   0.7304   0.6887  0.79167 0.000000
## Neg Pred Value       0.995935  0.97967   0.8550   0.7857  0.94595 0.987755
## Prevalence           0.004065  0.02033   0.4187   0.4187  0.12602 0.012195
## Detection Rate       0.000000  0.00000   0.3415   0.2967  0.07724 0.000000
## Detection Prevalence 0.000000  0.00000   0.4675   0.4309  0.09756 0.004065
## Balanced Accuracy    0.500000  0.50000   0.7994   0.7390  0.79482 0.497942
```

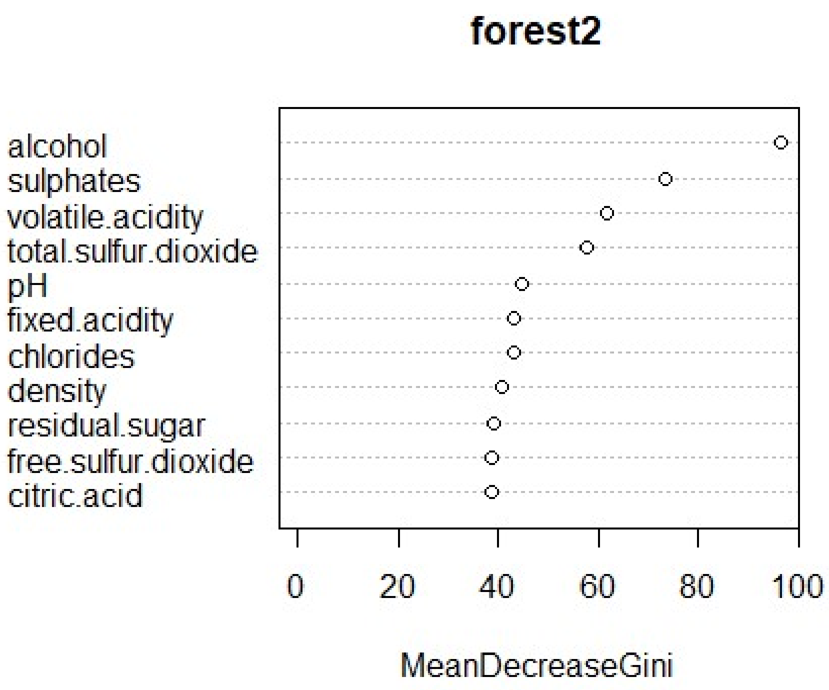
```
plot(forest2)
```

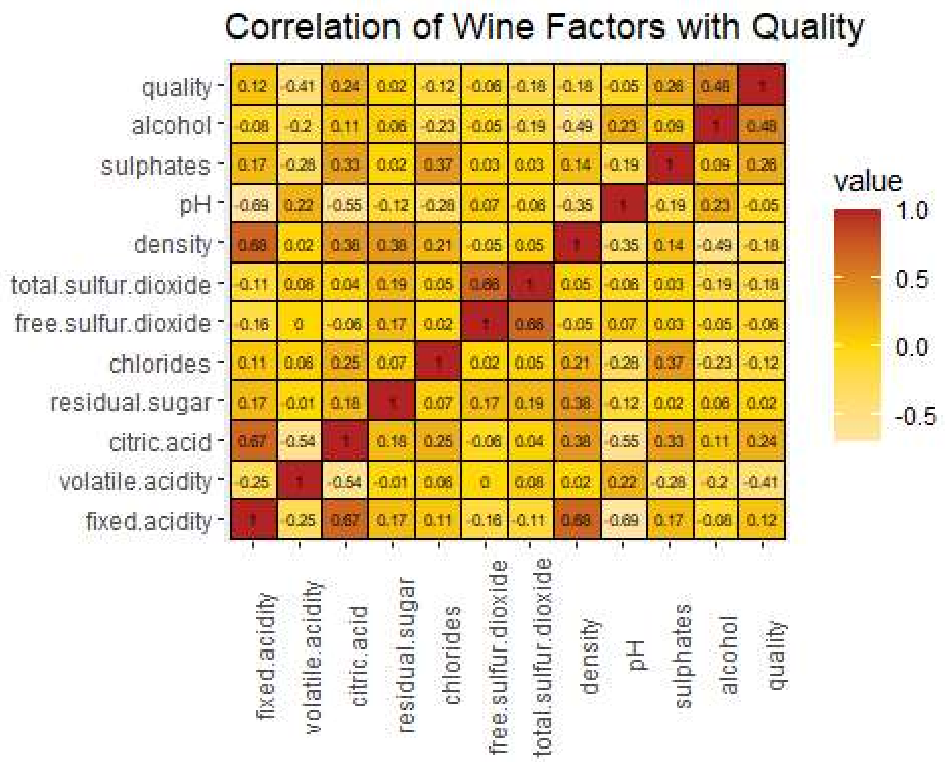


```
importance(forest2)
```

```
##                      MeanDecreaseGini
## fixed.acidity                43.24782
## volatile.acidity             61.61125
## citric.acid                  38.67895
## residual.sugar               38.97394
## chlorides                    43.10834
## free.sulfur.dioxide          38.78258
## total.sulfur.dioxide         57.61463
## density                      40.74653
## pH                           44.68111
## sulphates                    73.52353
## alcohol                      96.31902
```

```
varImpPlot(forest2)
```





	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	quality
quality	0.12	-0.41	0.24	0.02	-0.12	-0.06	-0.18	-0.18	-0.05	0.26	0.48	1
alcohol	-0.08	-0.2	0.11	0.06	-0.23	-0.05	-0.19	-0.49	0.23	0.09	1	0.48
sulphates	0.17	-0.26	0.33	0.02	0.37	0.03	0.03	0.14	-0.19	1	0.09	0.26
pH	-0.69	0.22	-0.55	-0.12	-0.28	0.07	-0.08	-0.35	1	-0.19	0.23	-0.05
density	0.68	0.02	0.38	0.38	0.21	-0.05	0.05	1	-0.35	0.14	-0.49	-0.18
total.sulfur.dioxide	-0.11	0.08	0.04	0.19	0.05	0.66	1	0.05	-0.08	0.03	-0.19	-0.18
free.sulfur.dioxide	-0.16	0	-0.06	0.17	0.02	1	0.66	-0.05	0.07	0.03	-0.05	-0.06
chlorides	0.11	0.06	0.25	0.07	1	0.02	0.05	0.21	-0.28	0.37	-0.23	-0.12
residual.sugar	0.17	-0.01	0.18	1	0.07	0.17	0.19	0.38	-0.12	0.02	0.06	0.02
citric.acid	0.67	-0.54	1	0.18	0.25	-0.06	0.04	0.38	-0.55	0.33	0.11	0.24
volatile.acidity	-0.25	1	-0.54	-0.01	0.06	0	0.08	0.02	0.22	-0.26	-0.2	-0.41
fixed.acidity	1	-0.25	0.67	0.17	0.11	-0.16	-0.11	0.68	-0.69	0.17	-0.08	0.12

It's important to note that while volatile acidity has a strong impact on wine quality score, it's a negative impact. The plot of variable importance does not show which variable we want more of, just which ones have the largest impact. By comparing these variables to the tile plot from before, we can infer that people like their wine to be stronger, with low to moderate amounts of sulphates and low levels of volatile acidity.