Technical Appendix

Tasheena Narraidoo, Michael Shi, Reynaldo Pena

12/04/16

Contents

Code Book

Reading our datasets

We have 2 datasets, one for red wine and one for white wine. We will also combine these 2 datasets. We will give a summary statistics of the variables for each data set but we will interpret the summary statistics that is relevant to our analysis in more detail in our next section (Exploring Our Data and Data Pre-processing).

Our Variables

Each dataset has the same variable names. They each have 12 variables. Here is a description of our variables

	Variable	Unit	Some_Description
1	fixed acidity	g(tartaric acid)/dm3	Fixed acidity makes wines taste sour.
2	volatile acidity	g(acetic acid)/dm3	Can "spoil" the wine, but low levels may add to its complexity.
3	citric acid	g/dm3	Adds acidity to the wine.
4	residual sugar	g/dm3	Determines sweetness of wine.
5	chlorides	g(sodium chloride)/dm3	Amount of salt in the wine.
6	free sulfur dioxide	mg/dm3	Prevents microbial growth and the oxidation of wine.
7	total sulfur dioxide	mg/dm3	Amount of free and bound forms of SO2.
8	density	g/cm3	Sweet wines have high density and dry wines have low density.
9	pH	pH	Acidity of wine.
10	sulphates	g(potassium sulphate)/dm3	Additive that contributes to S02 levels.
11	alcohol	vol.%	Alcohol content of wine.
12	quality	0-10	Rating of the wine [0(worst) – 10(best)].

Table 1: Variable Description

White Wine

We have 4,898 observations for white wine.

```
nrow(whitewine)
```

```
## [1] 4898
```

fixed.acidity takes values from 3.80 to 14.20. Its unit of measurement is g(tartaric acid)/dm3. The mean fixed.acidity is 6.85. It is a continuous input variable in assessing wine quality.

volatile.acidity takes values from 0.080 to 1.1. Its unit of measurement is g(acetic acid)/dm3. The mean volatile.acidity is 0.278. It is a continuous input variable in assessing wine quality.

citric.acid takes values from 0 to 1.66. Its unit of measurement is g/dm3. The mean citric.acid is 0.334. It is a continuous input variable in assessing wine quality.

residual.sugar takes values from 0.6 to 65.8. Its unit of measurement is g/dm3. The mean residual.sugar is 6.39. It is a continuous input variable in assessing wine quality.

chlorides takes values from 0.009 to 0.346. Its unit of measurement is g(sodium chloride)/dm3. The mean chlorides is 0.0458. It is a continuous input variable in assessing wine quality.

free.sulfur.dioxide takes value from 2.0 to 289.0. Its unit of measurement is mg/dm3. The mean free.sulfur.dioxide is 35.3. It is a continuous input variable in assessing wine quality.

total.sulfur.dioxide takes values from 9 to 440. Its unit of measurement is mg/dm3. The mean total.sulfur.dioxide is 138. It is a continuous input variable in assessing wine quality.

density takes values from .987 to 1.039. Its unit of measurement is g/cm3. The mean density is 0.994. It is a continuous input variable in assessing wine quality.

pH takes values form 2.72 to 3.82. The mean pH is 3.19. It is a continuous input variable in assessing wine quality.

sulphates take values from .22 to 1.08. Its unit of measurement is g(potassium sulphate)/dm3. The mean sulphates is 0.49. It is a continuous input variable in assessing wine quality.

alcohol takes values from 8 to 14.2. Its unit of measurement is vol.%. The mean alcohol is 10.5 %. It is a continuous input variable in assessing wine quality.

quality (which is within the [0,10] range) takes value from 3 to 9. It is the value attributed to the quality of wine, 0 being the lowest quality and 10, the highest.

	min	Q1	median	Q3	max	mean	sd	n	missing
fixed.acidity	3.80	6.30	6.80	7.30	14.20	6.85	0.84	4898	0
volatile.acidity	0.08	0.21	0.26	0.32	1.10	0.28	0.10	4898	0
citric.acid	0.00	0.27	0.32	0.39	1.66	0.33	0.12	4898	0
residual.sugar	0.60	1.70	5.20	9.90	65.80	6.39	5.07	4898	0
chlorides	0.01	0.04	0.04	0.05	0.35	0.05	0.02	4898	0
free.sulfur.dioxide	2.00	23.00	34.00	46.00	289.00	35.31	17.01	4898	0
total.sulfur.dioxide	9.00	108.00	134.00	167.00	440.00	138.36	42.50	4898	0
density	0.99	0.99	0.99	1.00	1.04	0.99	0.00	4898	0
pH	2.72	3.09	3.18	3.28	3.82	3.19	0.15	4898	0
sulphates	0.22	0.41	0.47	0.55	1.08	0.49	0.11	4898	0
alcohol	8.00	9.50	10.40	11.40	14.20	10.51	1.23	4898	0
quality	3.00	5.00	6.00	6.00	9.00	5.88	0.89	4898	0

Table 2: White wine summary statistics

From the above table, we notice that range is much larger compared to the IQR. For instance, the range for fixed acidity is 10.4 g/dm3 while its IQR is 1 g/dm3. In all cases except for quality(where it is less) and density (where it is equal), the mean is greater than the median. These observations indicate that there are outliers in the dataset.

Since we will use this specific dataset for Random Forest, we don't need to worry about the outliers as it is not sensitive to outliers.

However, we will use it for SVM. We therefore need to perform an outlier analysis when we use the dataset for SVM as it is sensitive to outliers.

Red Wine

We have 1599 red wine observations

```
nrow(redwine)
```

```
## [1] 1599
```

fixed.acidity takes values from 4.6 to 15.9. Its unit of measurement is g(tartaric acid)/dm3. The mean fixed.acidity is 8.32. It is a continuous input variable in assessing wine quality.

volatile.acidity takes values from 0.12 to 1.58. Its unit of measurement is g(acetic acid)/dm3. The mean volatile.acidity is 0.528. It is a continuous input variable in assessing wine quality.

citric.acid takes values from 0.09 to 1. Its unit of measurement is g/dm3. The mean citric.acid is 0.271. It is a continuous input variable in assessing wine quality.

residual.sugar takes values from 0.9 to 15.5. Its unit of measurement is g/dm3. The mean residual.sugar is 2.54. It is a continuous input variable in assessing wine quality.

chlorides takes values from 0.012 to 0.611. Its unit of measurement is g(sodium chloride)/dm3. The mean chlorides is 0.0875. It is a continuous input variable in assessing wine quality.

free.sulfur.dioxide takes value from 1 to 72. Its unit of measurement is mg/dm3. The mean free.sulfur.dioxide is 15.9. It is a continuous input variable in assessing wine quality.

total.sulfur.dioxide takes values from 6 to 289. Its unit of measurement is mg/dm3. The mean total.sulfur.dioxide is 46.5. It is a continuous input variable in assessing wine quality.

density takes values from 0.99 to 1. Its unit of measurement is g/cm3. The mean density is 0.997. It is a continuous input variable in assessing wine quality.

pH takes values form 2.74 to 4.01. The mean pH is 3.31. It is a continuous input variable in assessing wine quality.

sulphates take values from 0.33 to 2. Its unit of measurement is g(potassium sulphate)/dm3. The mean sulphates is 0.658. It is a continuous input variable in assessing wine quality.

alcohol takes values from 8.4 to 14.9. Its unit of measurement is vol.%. The mean alcohol is 10.4 %. It is a continuous input variable in assessing wine quality.

quality (which is within the [0,10] range) takes value from 3 to 8. It is the value attributed to the quality of wine, 0 being the lowest quality and 10, the highest.

Here is a summary for red wine:

	min	Q1	median	Q3	max	mean	sd	n	missing
fixed.acidity	4.60	7.10	7.90	9.20	15.90	8.32	1.74	1599	0
volatile.acidity	0.12	0.39	0.52	0.64	1.58	0.53	0.18	1599	0
citric.acid	0.00	0.09	0.26	0.42	1.00	0.27	0.19	1599	0
residual.sugar	0.90	1.90	2.20	2.60	15.50	2.54	1.41	1599	0
chlorides	0.01	0.07	0.08	0.09	0.61	0.09	0.05	1599	0
free.sulfur.dioxide	1.00	7.00	14.00	21.00	72.00	15.87	10.46	1599	0
total.sulfur.dioxide	6.00	22.00	38.00	62.00	289.00	46.47	32.90	1599	0
density	0.99	1.00	1.00	1.00	1.00	1.00	0.00	1599	0
pH	2.74	3.21	3.31	3.40	4.01	3.31	0.15	1599	0
sulphates	0.33	0.55	0.62	0.73	2.00	0.66	0.17	1599	0
alcohol	8.40	9.50	10.20	11.10	14.90	10.42	1.07	1599	0
quality	3.00	5.00	6.00	6.00	8.00	5.64	0.81	1599	0

Table 3: Red wine summary statistics

We see similar trend for red wine as we saw for white wine. From the above table, we notice that range is much larger compared to the IQR. For instance, the range for fixed acidity is 11.3 g/dm3 while its IQR is 2.1 g/dm3. In all cases except for quality(where it is less), pH (where it is equal) and density (where it is equal), the mean is greater than the median. These observations indicate that there are outliers in the dataset.

Since we will use this specific dataset for Random Forest, we don't need to worry about the outliers as it is not sensitive to outliers.

However, we will use it for SVM. We therefore need to perform an outlier analysis when we use the dataset for SVM as it is sensitive to outliers.

Combined data set

We will now combine the 2 datasets. We will use the combined data sets for our cluster analysis.

```
nrow(redwine) #1599
```

```
FALSE [1] 1599
```

```
nrow(whitewine) # 4898
```

```
FALSE [1] 4898
```

```
redwine[,"type"]  <- c("red")
whitewine[,"type"]  <- c("white")

wine <- join(redwine, whitewine,type = "full")
```

```
FALSE Joining by: fixed.acidity, volatile.acidity, citric.acid, residual.sugar, chlorides, free.sulfur.
```

We are adding a new variable called 'type' which takes values "red" or "white" to denote the type of wine according.

```
tally(~type, data = wine)
```

```
## type
##   red white
##  1599  4898
```

We have a total of 6497 observations for the combined dataset.

```
nrow(whitewine)
```

```
## [1] 4898
```

fixed.acidity takes values from 3.80 to 15.9. Its unit of measurement is g(tartaric acid)/dm3. The mean fixed.acidity is 7.22. It is a continuous input variable in assessing wine quality.

volatile.acidity takes values from 0.080 to 1.58. Its unit of measurement is g(acetic acid)/dm3. The mean volatile.acidity is 0.34. It is a continuous input variable in assessing wine quality.

citric.acid takes values from 0 to 1.66. Its unit of measurement is g/dm3. The mean citric.acid is 0.319. It is a continuous input variable in assessing wine quality.

residual.sugar takes values from 0.6 to 65.8. Its unit of measurement is g/dm3. The mean residual.sugar is 5.44. It is a continuous input variable in assessing wine quality.

chlorides takes values from 0.009 to 0.611. Its unit of measurement is g(sodium chloride)/dm3. The mean chlorides is 0.056. It is a continuous input variable in assessing wine quality.

free.sulfur.dioxide takes value from 1 to 289.0. Its unit of measurement is mg/dm3. The mean free.sulfur.dioxide is 30.5. It is a continuous input variable in assessing wine quality.

total.sulfur.dioxide takes values from 6 to 440. Its unit of measurement is mg/dm3. The mean total.sulfur.dioxide is 116. It is a continuous input variable in assessing wine quality.

density takes values from .987 to 1.04. Its unit of measurement is g/cm3. The mean density is 0.995. It is a continuous input variable in assessing wine quality.

pH takes values form 2.72 to 4.01. The mean pH is 3.22. It is a continuous input variable in assessing wine quality.

sulphates take values from .22 to 2. Its unit of measurement is g(potassium sulphate)/dm3. The mean sulphates is 0.531. It is a continuous input variable in assessing wine quality.

alcohol takes values from 8 to 14.9. Its unit of measurement is vol.%. The mean alcohol is 10.5 %. It is a continuous input variable in assessing wine quality.

quality (which is within the [0,10] range) takes value from 3 to 9. It is the value attributed to the quality of wine, 0 being the lowest quality and 10, the highest.

Here is a summary for the combined wines:

	min	Q1	median	Q3	max	mean	sd	n	missing
fixed.acidity	3.80	6.40	7.00	7.70	15.90	7.22	1.30	6497	0
volatile.acidity	0.08	0.23	0.29	0.40	1.58	0.34	0.16	6497	0
citric.acid	0.00	0.25	0.31	0.39	1.66	0.32	0.15	6497	0
residual.sugar	0.60	1.80	3.00	8.10	65.80	5.44	4.76	6497	0
chlorides	0.01	0.04	0.05	0.07	0.61	0.06	0.04	6497	0
free.sulfur.dioxide	1.00	17.00	29.00	41.00	289.00	30.53	17.75	6497	0
total.sulfur.dioxide	6.00	77.00	118.00	156.00	440.00	115.74	56.52	6497	0
density	0.99	0.99	0.99	1.00	1.04	0.99	0.00	6497	0
pH	2.72	3.11	3.21	3.32	4.01	3.22	0.16	6497	0
sulphates	0.22	0.43	0.51	0.60	2.00	0.53	0.15	6497	0
alcohol	8.00	9.50	10.30	11.30	14.90	10.49	1.19	6497	0
quality	3.00	5.00	6.00	6.00	9.00	5.82	0.87	6497	0

Table 3: Combined wine summary statistics

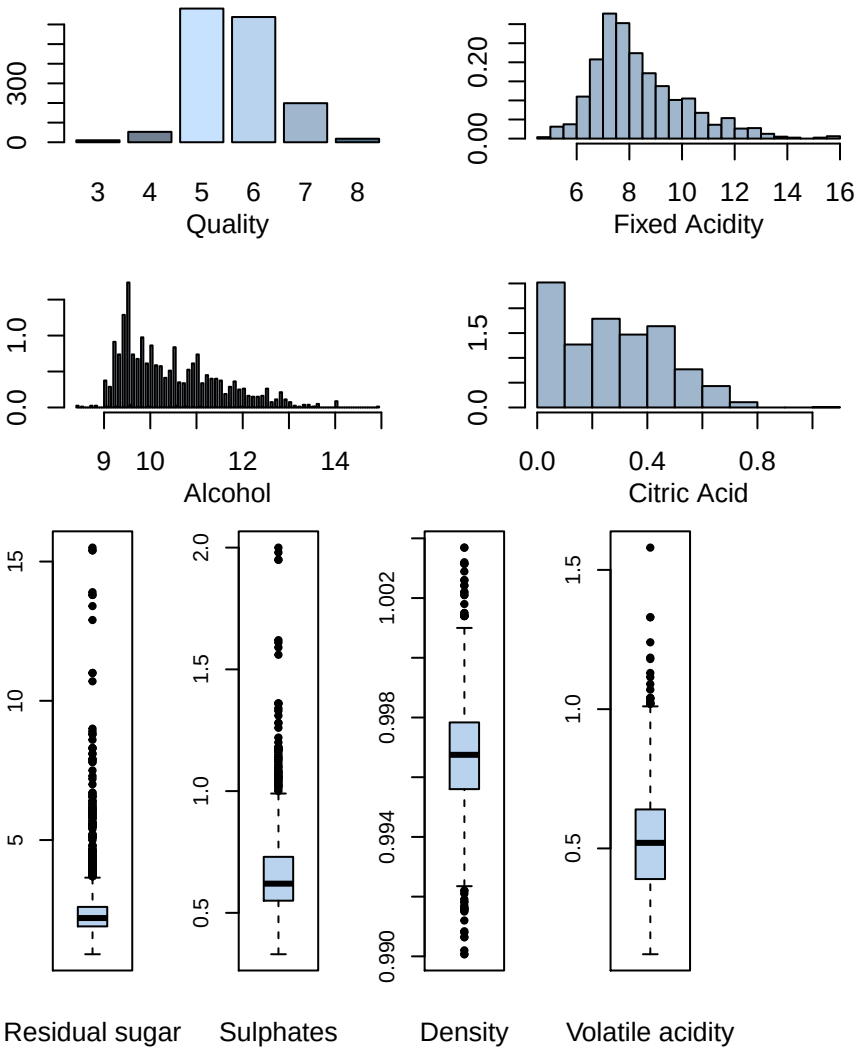
We see similar patterns when we combine the two datasets. From the above table, we notice that range is much larger compared to the IQR as we saw in the individual datasets. Again, in most cases the mean is greater than the median. These observations indicate that there are outliers in the combined dataset as well.

We will perform an outlier analysis which we will apply for cluster analysis as it is sensitive to outliers. We won't need to worry about the outliers for the PCA analysis.

Exploring Our Data and Data Pre-processing

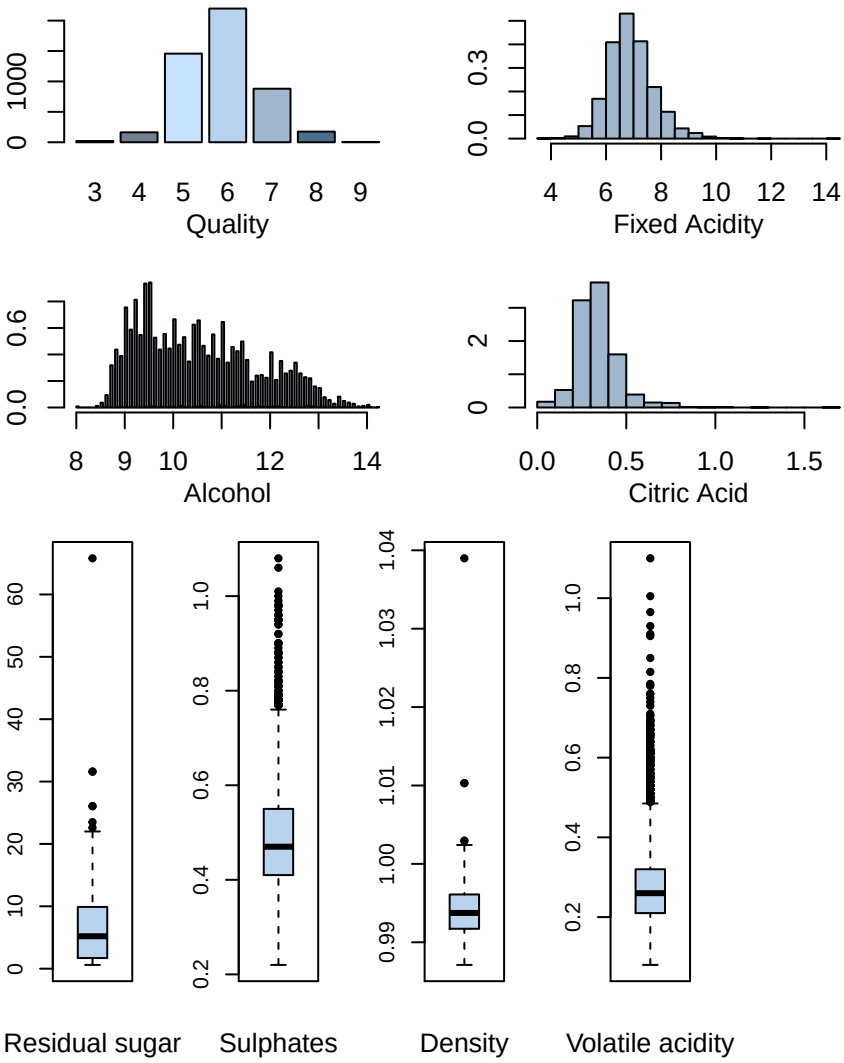
Exploring our variables

Red Wine



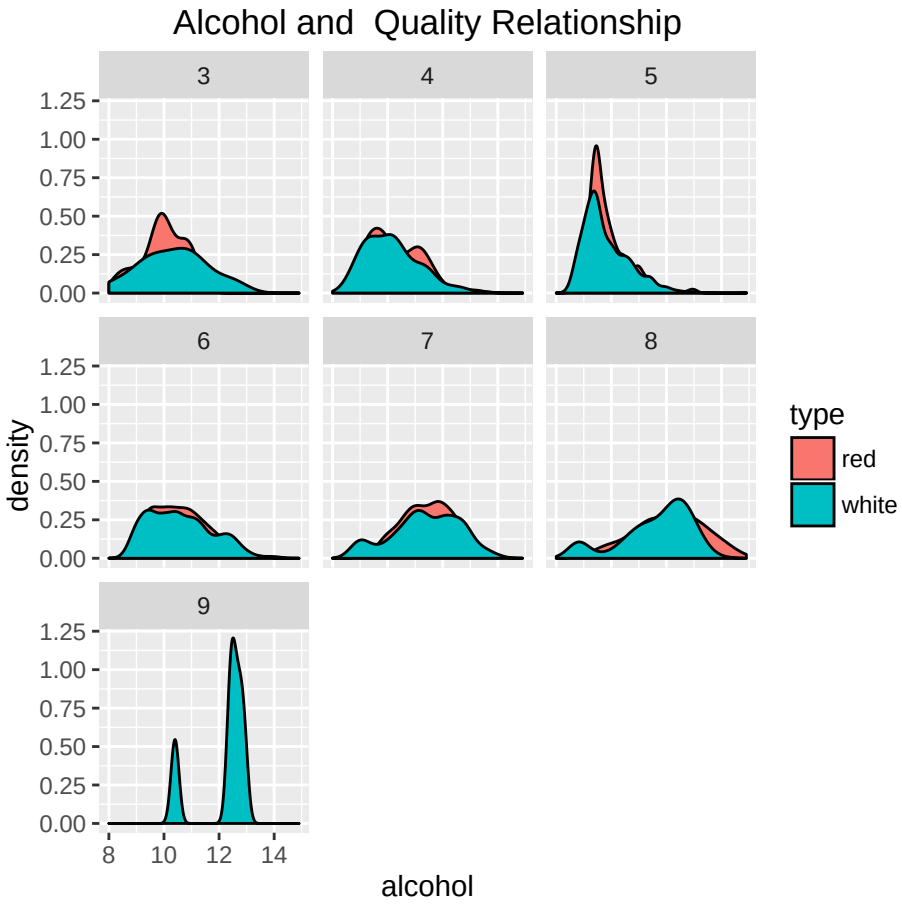
The above histograms, and boxplots reinforces the idea that we got earlier from our descriptive statistics summary - we have outliers for red wine. Quality has most values concentrated in the categories 5, 6 and 7. The distributions for fixed acidity, volatile acidity and citric acid are not symmetric. Residual sugar is positively skewed. Free sulphur dioxide and density have only a few outliers. Alcohol's distribution is different from the rest. It is worth some further exploration.

White Wine



The above histograms, and boxplots reinforces the idea that we got earlier from our descriptive statistics summary - we have outliers for white wine. Quality has most values concentrated in the categories 5, 6 and 7. The distributions for fixed acidity, volatile acidity and citric acid are not symmetric. Residual sugar is positively skewed. Free sulphur dioxide and density have only a few outliers. Alcohol's distribution is different from the rest. It is worth some further exploration.

We go on to explore alcohol against our response variable.



Interesting Finding

It would seem that red wine with higher alcohol content have lower ratings while white wine with higher alcohol content have higher rating.

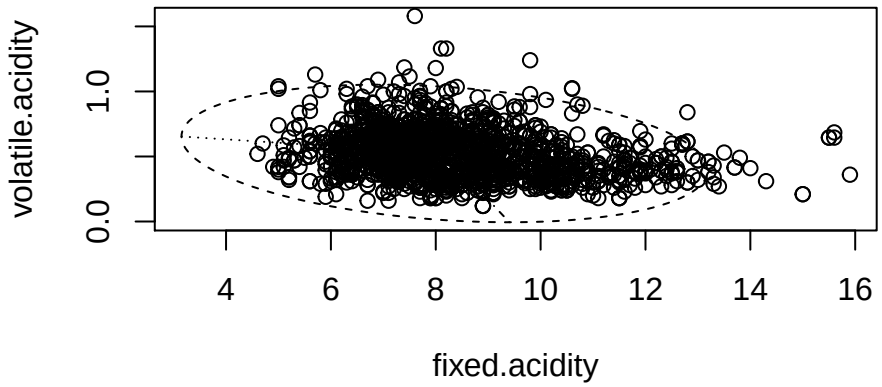
Outlier Analysis

Red wine

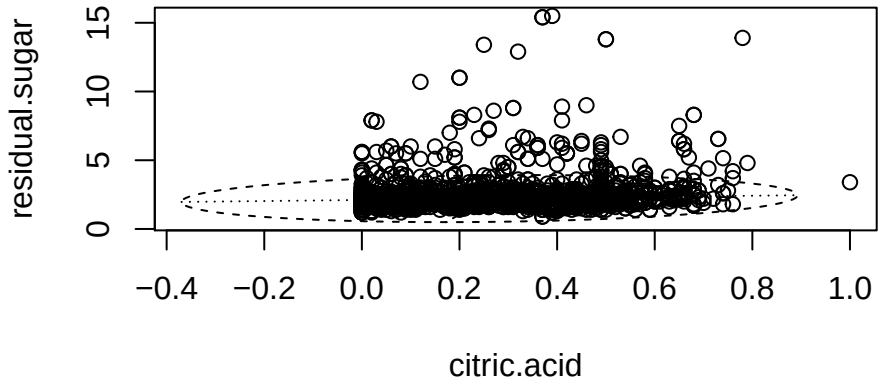
We will use the data from this analysis for SVM

Below we looked at some bivariate data and based on the following plots we choose a cut-off for our outliers for red wine.

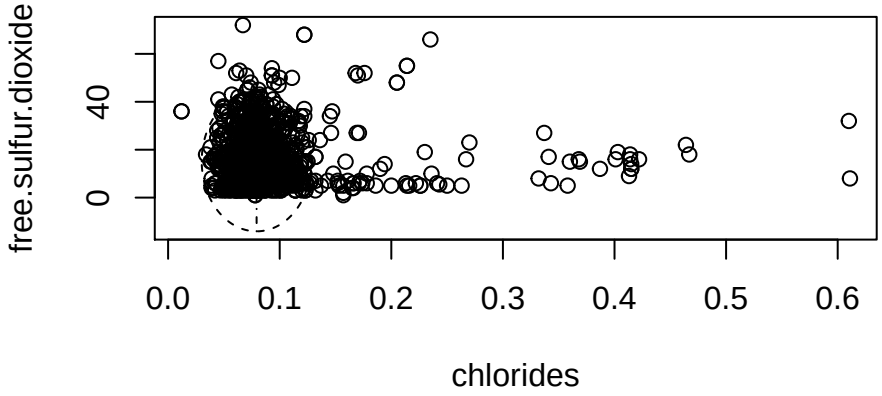
```
RedWine <- redwine
twoVars = RedWine[c("fixed.acidity", "volatile.acidity")]
bvbox(twoVars, mtitle = "", xlab = "fixed.acidity", ylab = "volatile.acidity")
```



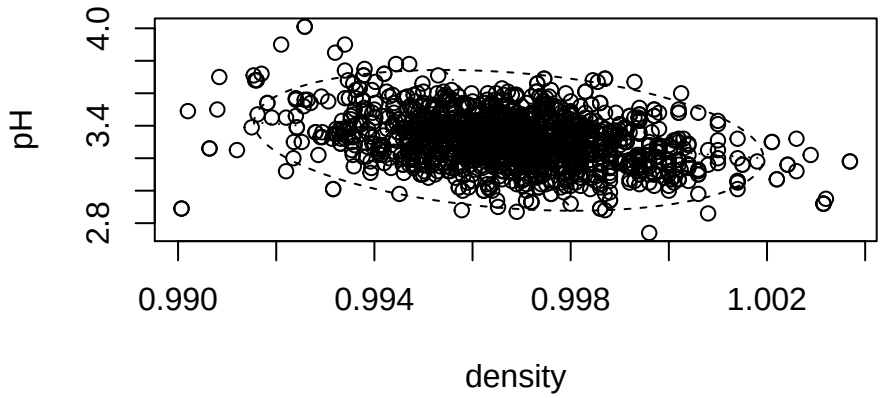

```
twoVars = RedWine[c("citric.acid", "residual.sugar")]
bvbox(twoVars, mtitle = "", xlab = "citric.acid", ylab = "residual.sugar")
```



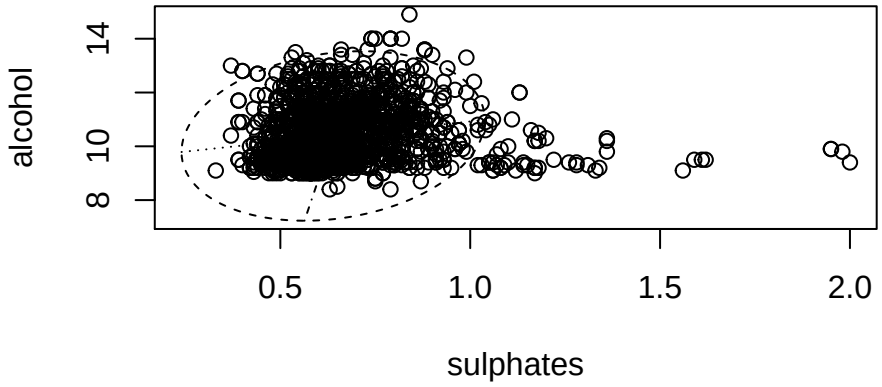

```
twoVars = RedWine[c("chlorides", "free.sulfur.dioxide")]
bvbox(twoVars, mtitle = "", xlab = "chlorides", ylab = "free.sulfur.dioxide")
```



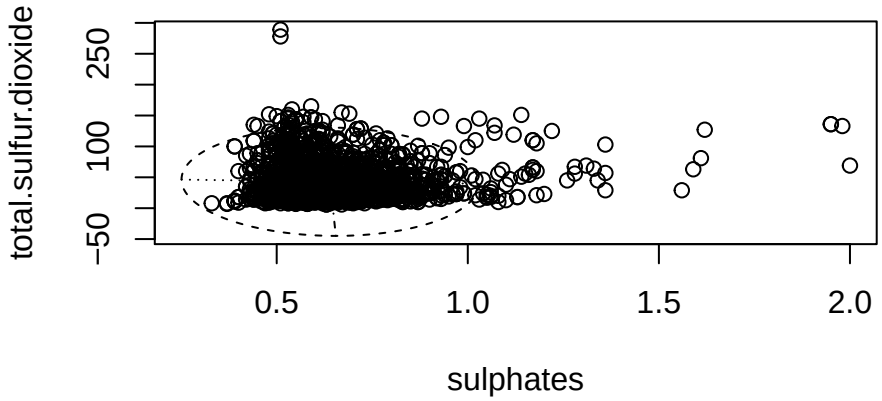

```
twoVars = RedWine[c("density", "pH")]
bvbox(twoVars, mtitle = "", xlab = "density", ylab = "pH")
```




```
twoVars = RedWine[c("sulphates", "alcohol")]
bvbox(twoVars, mtitle = "", xlab = "sulphates", ylab = "alcohol")
```




```
twoVars = RedWine[c("sulphates", "total.sulfur.dioxide")]
bvbox(twoVars, mtitle = "", xlab = "sulphates", ylab = "total.sulfur.dioxide")
```



You can find the cuff-offs from the following code chunk. They were based on the data points from the above plots based on their distance from the majority of the data points.

```
outliers = with(RedWine, c(which(volatile.acidity>1.4),
                           which(fixed.acidity>14.5),
                           which(citric.acid>1),
                           which(residual.sugar>10),
                           which(chlorides>0.5),
```

```
                            which(free.sulfur.dioxide>60),
                            which (total.sulfur.dioxide>200),
                            which(density<.9905), which(sulphates>1.4)))

RedWine2 = RedWine[-outliers,]
```

```
nrow(redwine)
```

```
## [1] 1599
```

```
nrow(RedWine2)
```

```
## [1] 1563
```

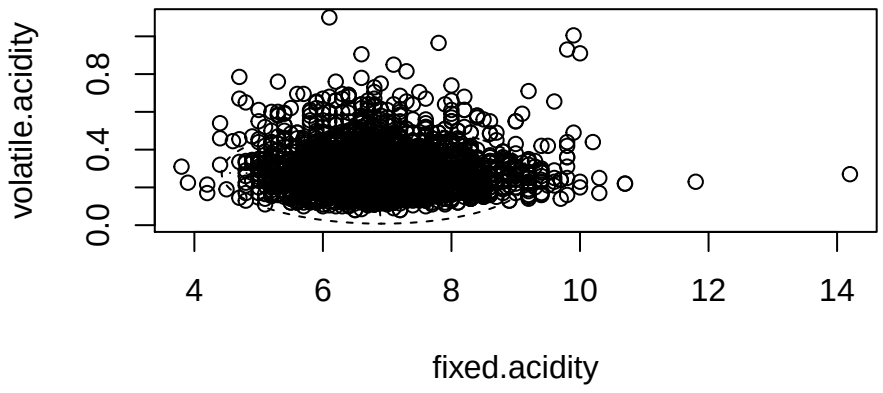
Given the above boxplots, we eliminate major outliers below. In total, we eliminated 36 data points and are left with 1563.

White wine

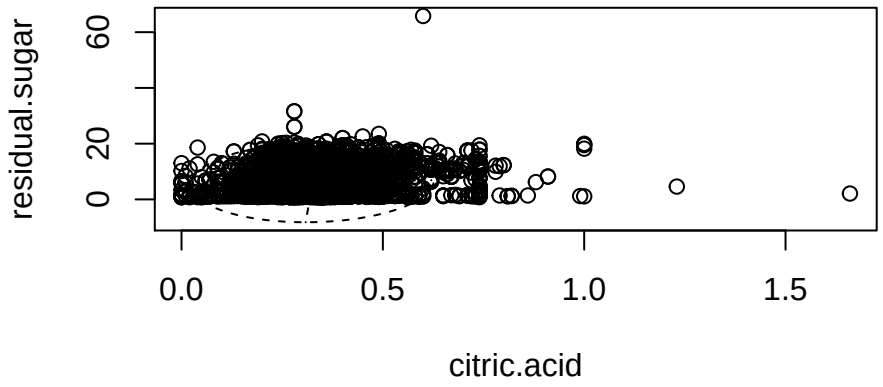
We will use the data from this analysis for SVM

We perform the same analysis as above. Below we looked at some bivariate data and based on the following plots we choose a cut-off for our outliers for white wine.

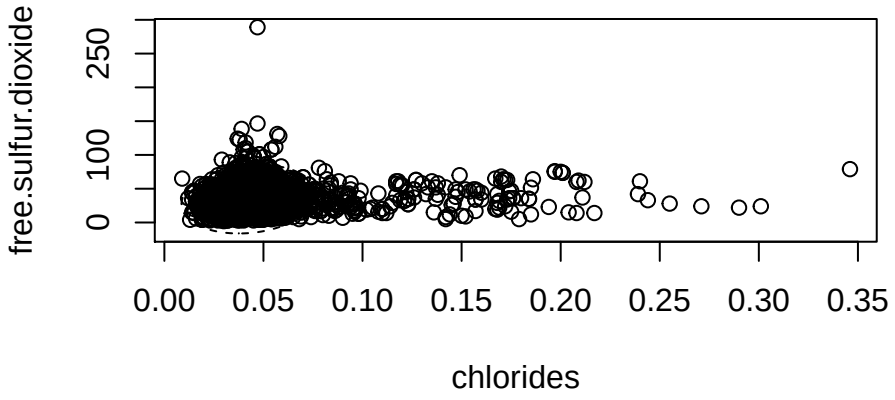
```
WhiteWine <- whitewine
twoVars = WhiteWine[c("fixed.acidity", "volatile.acidity")]
bvbox(twoVars, mtitle = "", xlab = "fixed.acidity", ylab = "volatile.acidity")
```



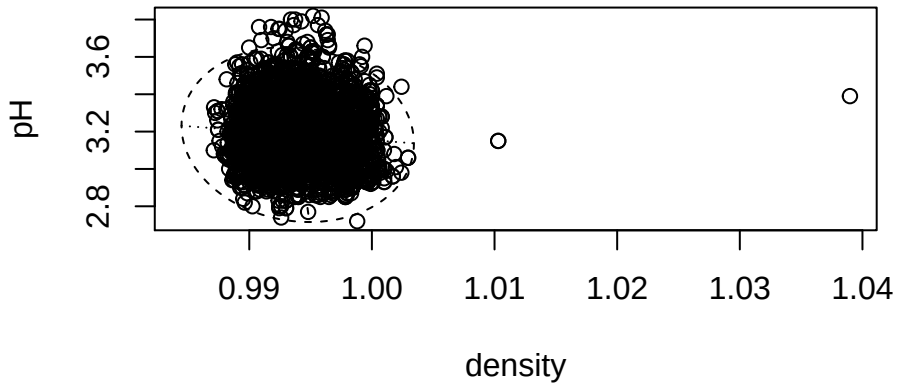

```
twoVars = WhiteWine[c("citric.acid", "residual.sugar")]
bvbox(twoVars, mtitle = "", xlab = "citric.acid", ylab = "residual.sugar")
```



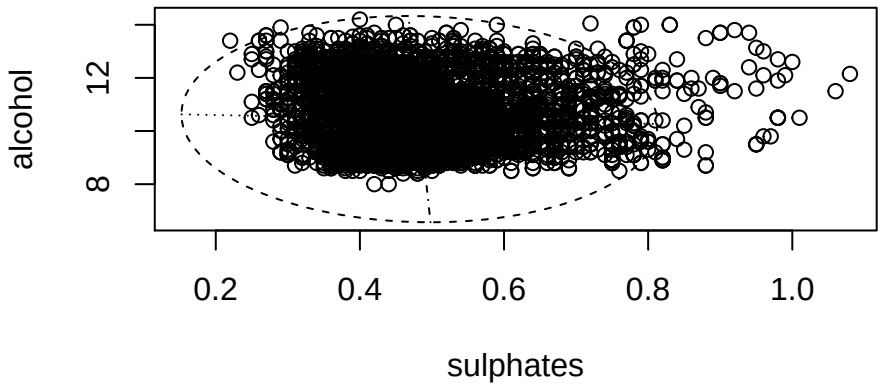

```
twoVars = WhiteWine[c("chlorides", "free.sulfur.dioxide")]
bvbox(twoVars, mtitle = "", xlab = "chlorides", ylab = "free.sulfur.dioxide")
```




```
twoVars = WhiteWine[c("density", "pH")]
bvbox(twoVars, mtitle = "", xlab = "density", ylab = "pH")
```




```
twoVars = WhiteWine[c("sulphates", "alcohol")]
bvbox(twoVars, mtitle = "", xlab = "sulphates", ylab = "alcohol")
```



The cuff-offs can be obtained from the following code chunk. They were based on the data points from the above plots based on their distance from the majority of the data points.

```
outliers2 = with(WhiteWine, c(which(volatile.acidity>1),
                              which(fixed.acidity>11.5),
                              which(citric.acid>1.2),
                              which(residual.sugar>40),
                              which (chlorides>0.23),
                              which(free.sulfur.dioxide>200),
                              which(density>1.01),
                              which(sulphates>1)))

WhiteWine2 = WhiteWine[-outliers2,]
```

```
nrow(whitewine)
```

```
## [1] 4898
```

```
nrow(WhiteWine2)
```

```
## [1] 4877
```

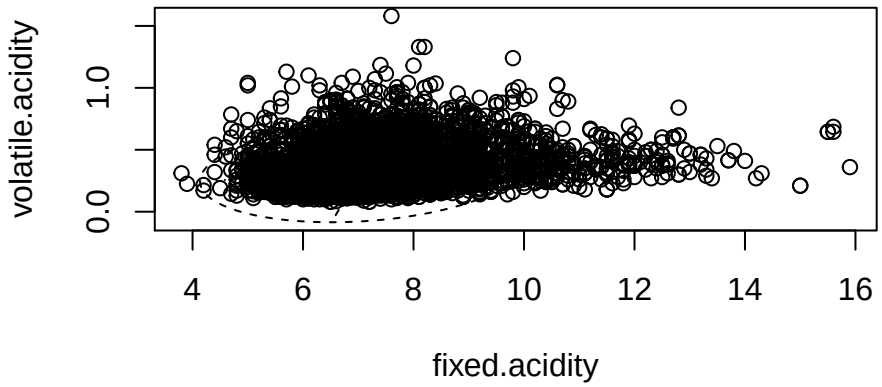
Now we eliminate the outliers from the White Wine data set. Our new data set WhiteWine2 has 21 less variables than the original.

Combined wines

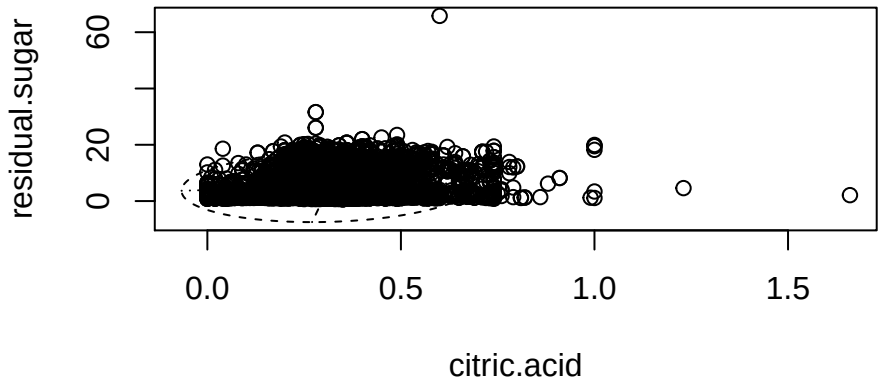
We will use the data from this analysis for clustering.

We add a new variable to our datasets called color to differentiate between red and white wine. This will help us for our cluster analysis. Again we explore bivariate plots to decide on the outlier cuff-offs

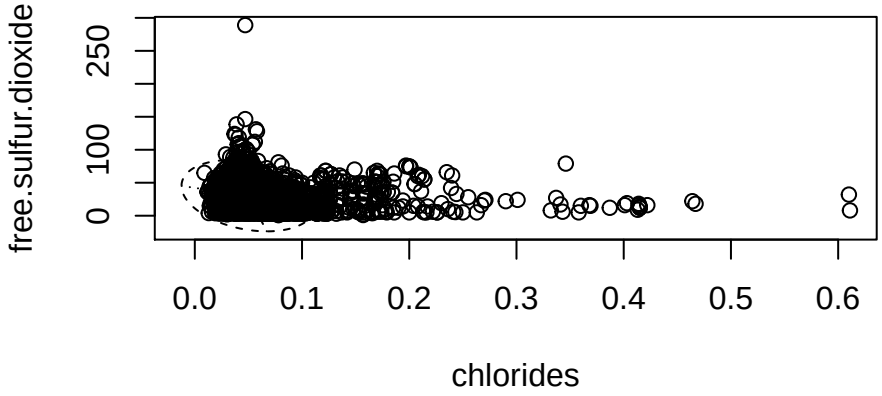
```
redwine2 <- redwine
whitewine2 <- whitewine

redwine2$color = 0
whitewine2$color = 1
AllWines = rbind(redwine2, whitewine2)
twoVars = AllWines[c("fixed.acidity", "volatile.acidity")]
bvbox(twoVars, mtitle = "AllWines", xlab = "fixed.acidity", ylab = "volatile.acidity")
```



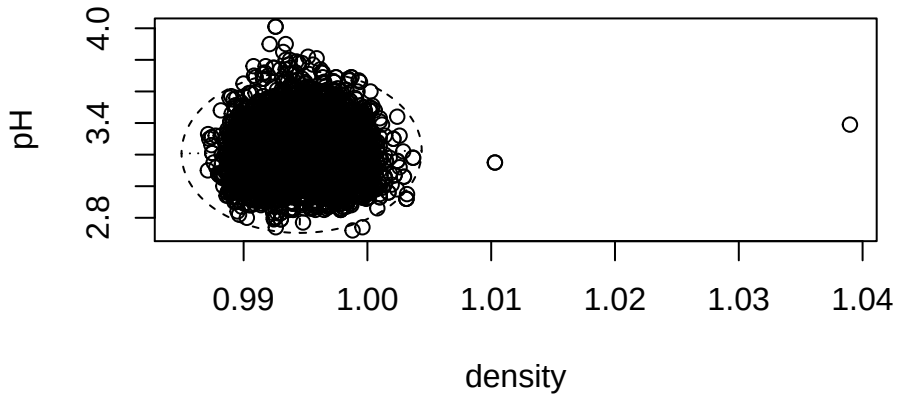

```
twoVars = AllWines[c("citric.acid", "residual.sugar")]
bvbox(twoVars, mtitle = "AllWines", xlab = "citric.acid", ylab = "residual.sugar")
```



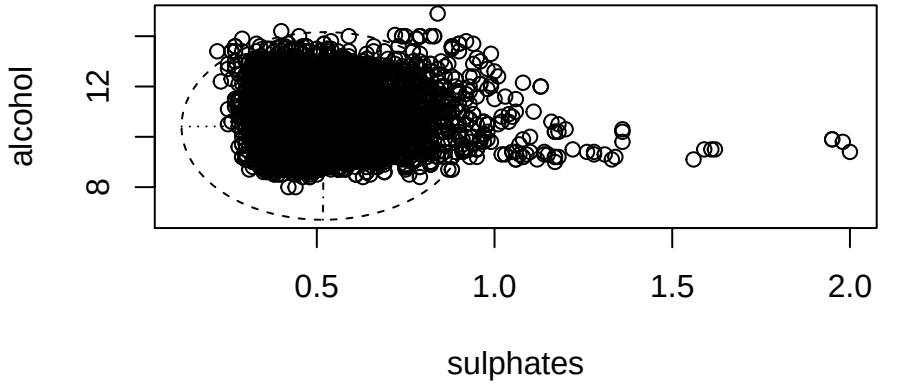

```
twoVars = AllWines[c("chlorides", "free.sulfur.dioxide")]
bvbox(twoVars, mtitle = "AllWines", xlab = "chlorides", ylab = "free.sulfur.dioxide")
```




```
twoVars = AllWines[c("density", "pH")]
bvbox(twoVars, mtitle = "AllWines", xlab = "density", ylab = "pH")
```




```
twoVars = AllWines[c("sulphates", "alcohol")]
bvbox(twoVars, mtitle = "AllWines", xlab = "sulphates", ylab = "alcohol")
```



We then eliminate outliers from this joint distribution. The cuff-offs can be obtained from the following code chunk. They were based on the data points from the above plots based on their distance from the majority of the data points.

```
outliers3 = with(AllWines, c(which(volatile.acidity>1),
                             which(fixed.acidity>11.5),
                             which(citric.acid>1.2),
                             which(residual.sugar>40),
                             which (chlorides>0.5),
                             which(free.sulfur.dioxide>200),
                             which(density>1.01)))

AllWines2 = AllWines[-outliers3,]
AllWinesB <- AllWines2
```

We have eliminated 129 points from the combined set. This may seem a lot compared to the individual sets but this may stem form merging their variability.

```
nrow(wine)
```

```
## [1] 6497
```

```
nrow(AllWines2)
```

```
## [1] 6368
```

Principal Component Analysis

PCA

Principal components analysis is a statistical approach that uses an orthogonal transformation to convert a set of correlated variables into another set of uncorrelated variables or principal components. We decided to use principal components analysis on our wine data set with the goal of dimensionality reduction. We wanted to find a way to express the same information represented by the original 12 variables in fewer. We will combine the wine for our PCA analysis. We don't have to worry our outliers here because PCA is not sensitive to outliers. We use scaling because we have different measurement units.

Combined wine - by type

```
cor(wine[,-13])
```

```
##                       fixed.acidity volatile.acidity citric.acid
## fixed.acidity                1.0000           0.2190      0.3244
## volatile.acidity             0.2190           1.0000     -0.3780
## citric.acid                  0.3244          -0.3780      1.0000
## residual.sugar              -0.1120          -0.1960      0.1425
## chlorides                    0.2982           0.3771      0.0390
## free.sulfur.dioxide         -0.2827          -0.3526      0.1331
## total.sulfur.dioxide        -0.3291          -0.4145      0.1952
## density                      0.4589           0.2713      0.0962
## pH                          -0.2527           0.2615     -0.3298
## sulphates                    0.2996           0.2260      0.0562
## alcohol                     -0.0955          -0.0376     -0.0105
## quality                     -0.0767          -0.2657      0.0855
##                       residual.sugar chlorides free.sulfur.dioxide
## fixed.acidity                 -0.112    0.2982             -0.2827
## volatile.acidity              -0.196    0.3771             -0.3526
## citric.acid                    0.142    0.0390              0.1331
## residual.sugar                 1.000   -0.1289              0.4029
## chlorides                     -0.129    1.0000             -0.1950
## free.sulfur.dioxide            0.403   -0.1950              1.0000
## total.sulfur.dioxide           0.495   -0.2796              0.7209
## density                        0.553    0.3626              0.0257
## pH                            -0.267    0.0447             -0.1459
## sulphates                     -0.186    0.3956             -0.1885
## alcohol                       -0.359   -0.2569             -0.1798
## quality                       -0.037   -0.2007              0.0555
##                       total.sulfur.dioxide density      pH sulphates
## fixed.acidity                      -0.3291  0.4589 -0.2527   0.29957
## volatile.acidity                   -0.4145  0.2713  0.2615   0.22598
```

```
## citric.acid                          0.1952  0.0962 -0.3298   0.05620
## residual.sugar                       0.4955  0.5525 -0.2673  -0.18593
## chlorides                           -0.2796  0.3626  0.0447   0.39559
## free.sulfur.dioxide                  0.7209  0.0257 -0.1459  -0.18846
## total.sulfur.dioxide                 1.0000  0.0324 -0.2384  -0.27573
## density                              0.0324  1.0000  0.0117   0.25948
## pH                                  -0.2384  0.0117  1.0000   0.19212
## sulphates                           -0.2757  0.2595  0.1921   1.00000
## alcohol                             -0.2657 -0.6867  0.1212  -0.00303
## quality                             -0.0414 -0.3059  0.0195   0.03849
##                        alcohol quality
## fixed.acidity        -0.09545 -0.0767
## volatile.acidity     -0.03764 -0.2657
## citric.acid          -0.01049  0.0855
## residual.sugar       -0.35941 -0.0370
## chlorides            -0.25692 -0.2007
## free.sulfur.dioxide  -0.17984  0.0555
## total.sulfur.dioxide -0.26574 -0.0414
## density              -0.68675 -0.3059
## pH                    0.12125  0.0195
## sulphates            -0.00303  0.0385
## alcohol               1.00000  0.4443
## quality               0.44432  1.0000
```

Looking at the correlations between variables in our data, it looks like there are a number of correlations above .3, indicating that there is some sort of underlying structure in the data and fulfills the assumptions required for principal components analysis.

```
PCWine=prcomp(wine[,-13],scale=TRUE)
PCWine$rotation
```

```
##                          PC1    PC2     PC3     PC4      PC5     PC6
## fixed.acidity        -0.2569  0.262 -0.4675  0.1440 -0.16536  0.0300
## volatile.acidity     -0.3949  0.105  0.2797  0.0801 -0.14777 -0.3827
## citric.acid           0.1465  0.144 -0.5881 -0.0555  0.23462  0.3622
## residual.sugar        0.3189  0.343  0.0755 -0.1125 -0.50792 -0.0633
## chlorides            -0.3134  0.270 -0.0468 -0.1653  0.39390 -0.4254
## free.sulfur.dioxide   0.4227  0.111  0.0990 -0.3033  0.24845 -0.2832
## total.sulfur.dioxide  0.4744  0.144  0.1013 -0.1322  0.22397 -0.1068
## density              -0.0924  0.555  0.0516 -0.1506 -0.33036  0.1546
## pH                   -0.2081 -0.153  0.4068 -0.4715  0.00146  0.5609
## sulphates            -0.2999  0.120 -0.1687 -0.5880  0.19325 -0.0201
## alcohol              -0.0589 -0.493 -0.2129 -0.0800 -0.11602 -0.1695
## quality               0.0875 -0.297 -0.2958 -0.4724 -0.45913 -0.2779
##                          PC7      PC8      PC9    PC10     PC11      PC12
## fixed.acidity        -0.3934  0.00116  0.42417 -0.2724 -0.27693 -0.335093
## volatile.acidity     -0.4451  0.31008 -0.12323  0.4939  0.14080 -0.082421
## citric.acid          -0.0477  0.44496 -0.24623  0.3304  0.22928  0.001347
## residual.sugar        0.0958  0.08194 -0.48802 -0.2072  0.00514 -0.451215
## chlorides             0.4733  0.37553 -0.04405 -0.2389 -0.19340 -0.043278
## free.sulfur.dioxide  -0.3627  0.12010  0.30140 -0.3034  0.48616 -0.000905
## total.sulfur.dioxide -0.2348  0.01128  0.00181  0.2948 -0.72016  0.064063
## density              -0.0133  0.04294  0.07108 -0.0768 -0.00332  0.715667
```

```
## pH                   -0.0793  0.36228  0.13666 -0.1124 -0.13908 -0.206763
## sulphates            -0.1702 -0.59222 -0.29740  0.0855  0.04722 -0.078200
## alcohol              -0.3389  0.22604 -0.41706 -0.4161 -0.19129  0.332012
## quality               0.2732  0.09305  0.35665  0.3078 -0.01808  0.008288
```

We run our PCA on correlations over covariances because our variables are on different scales and we do not want to weight variables with higher covariances differently.

```
summary(PCWine) #prop variance explained by each component
```

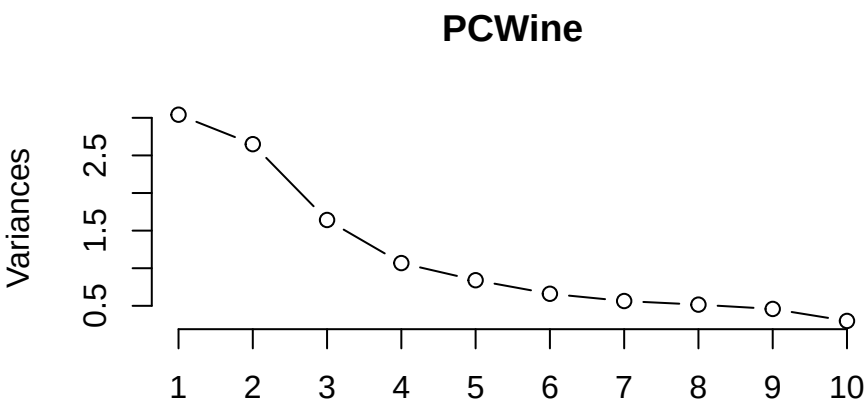
```
## Importance of components:
##                           PC1   PC2   PC3    PC4   PC5   PC6   PC7   PC8
## Standard deviation     1.744 1.628 1.281 1.0337 0.917 0.813 0.751 0.718
## Proportion of Variance 0.253 0.221 0.137 0.0891 0.070 0.055 0.047 0.043
## Cumulative Proportion  0.253 0.474 0.611 0.7001 0.770 0.825 0.872 0.915
##                           PC9   PC10  PC11    PC12
## Standard deviation     0.6770 0.5468 0.477 0.18107
## Proportion of Variance 0.0382 0.0249 0.019 0.00273
## Cumulative Proportion  0.9534 0.9783 0.997 1.00000
```

Now we would like to choose how many principal components to keep. Looking at the variance explained explained by each principal component, we can see we need 4 principal components to explain 70% of the variation in the data. According to the eigenvalue >0.7 rule, we need 5 principal components.

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	3.04	25.35	25.35
Dim.2	2.65	22.08	47.43
Dim.3	1.64	13.68	61.11
Dim.4	1.07	8.91	70.01
Dim.5	0.84	7.00	77.02
Dim.6	0.66	5.50	82.52

Table 2: Eigenvalue for first 6 components

```
screeplot(PCWine, type="l") #it is an L
```



Looking at the scree plot, there is a slight elbow at 4 principal components.

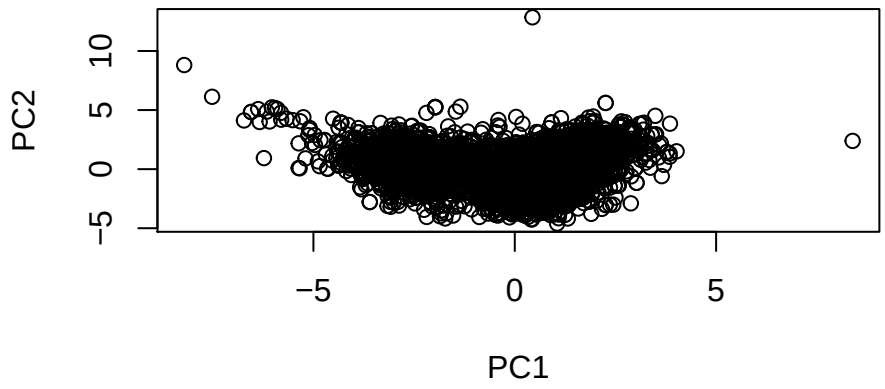
Since our overall goal is to reduce the number of variables and 2 of the 3 guidelines recommend 4 principal components, we will use 4 principal components.

```r
cor(wine[,-13],PCWine$x) #loadings
```

```
##                         PC1    PC2     PC3     PC4      PC5     PC6
## fixed.acidity        -0.448  0.426 -0.5989  0.1488 -0.15160  0.0244
## volatile.acidity     -0.689  0.171  0.3583  0.0828 -0.13548 -0.3110
## citric.acid           0.255  0.235 -0.7535 -0.0574  0.21510  0.2944
## residual.sugar        0.556  0.558  0.0967 -0.1163 -0.46566 -0.0515
## chlorides            -0.547  0.439 -0.0599 -0.1709  0.36112 -0.3457
## free.sulfur.dioxide   0.737  0.181  0.1268 -0.3135  0.22778 -0.2301
## total.sulfur.dioxide  0.827  0.234  0.1298 -0.1367  0.20533 -0.0868
## density              -0.161  0.903  0.0661 -0.1557 -0.30287  0.1256
## pH                   -0.363 -0.249  0.5212 -0.4874  0.00134  0.4558
## sulphates            -0.523  0.195 -0.2161 -0.6079  0.17717 -0.0164
## alcohol              -0.103 -0.802 -0.2728 -0.0827 -0.10637 -0.1377
## quality               0.153 -0.483 -0.3790 -0.4884 -0.42092 -0.2258
##                          PC7      PC8      PC9    PC10     PC11      PC12
## fixed.acidity        -0.29542  0.00083  0.28718 -0.1490 -0.13211 -0.060674
## volatile.acidity     -0.33423  0.22273 -0.08343  0.2701  0.06717 -0.014924
## citric.acid          -0.03582  0.31963 -0.16671  0.1806  0.10938  0.000244
## residual.sugar        0.07191  0.05886 -0.33041 -0.1133  0.00245 -0.081700
## chlorides             0.35539  0.26975 -0.02982 -0.1306 -0.09226 -0.007836
## free.sulfur.dioxide  -0.27236  0.08627  0.20406 -0.1659  0.23193 -0.000164
## total.sulfur.dioxide -0.17632  0.00810  0.00123  0.1612 -0.34356  0.011600
## density              -0.00998  0.03085  0.04812 -0.0420 -0.00159  0.129583
## pH                   -0.05956  0.26023  0.09252 -0.0615 -0.06635 -0.037438
## sulphates            -0.12783 -0.42540 -0.20135  0.0467  0.02253 -0.014159
## alcohol              -0.25448  0.16237 -0.28236 -0.2275 -0.09126  0.060116
## quality               0.20512  0.06684  0.24146  0.1683 -0.00863  0.001501
```

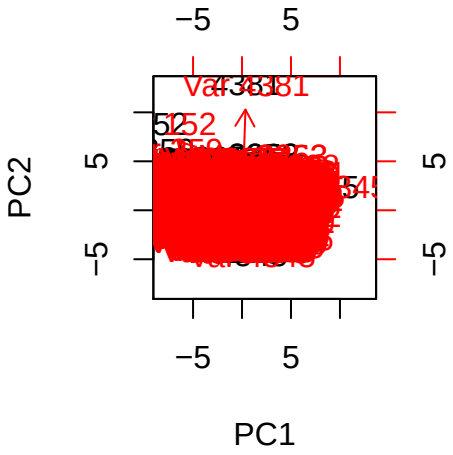
Looking at the loadings, the first PC represents a negative relationship between the variables fixed and volatile acidity, chlorides, pH, and sulphates and the variables residual sugar and sulfur dioxide. The second PC mainly capures the negative relationship between the varibles density, fixed acidity, residual sugar, and chlorides and the variables alcohol and quality. The third PC represents the negative relationship between the variables citric acid and fixed acidity and the variable pH. The 4th PC represents the positive relationship between pH, sulphates, and quality.

```r
plot(PCWine$x)
```

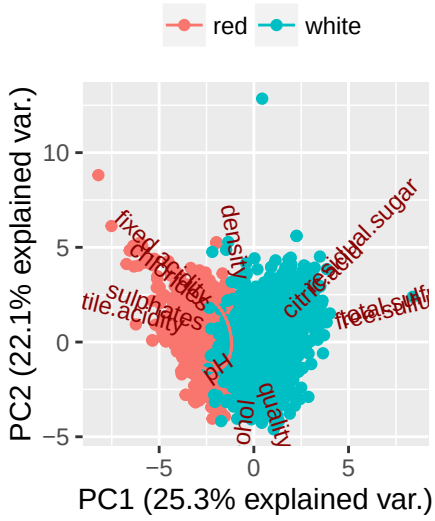


```
biplot(PCWine$x[,1:2],PCWine$x[,1:2])
```

```
## Warning in arrows(0, 0, y[, 1L] * 0.8, y[, 2L] * 0.8, col = col[2L], length
## = arrow.len): zero-length arrow is of indeterminate angle and so skipped

## Warning in arrows(0, 0, y[, 1L] * 0.8, y[, 2L] * 0.8, col = col[2L], length
## = arrow.len): zero-length arrow is of indeterminate angle and so skipped
```




```
g <- ggbiplot(PCWine, choices=1:2, obs.scale = 1, var.scale = 1, groups = wine$type,
              ellipse = TRUE,
               circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
print(g)
```

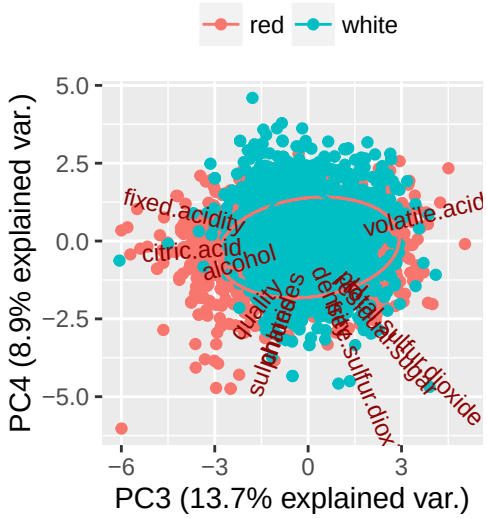


Looking at the biplot for all the points, we can see a clear separation of groups when looking at the first 2 principal components. The red wines have a lower value for PC1, which would indicate higher levels of fixed acidity, volatile acidity, chlorides, and sulphates and lower levels of residual sugar and free and total sulfur dioxide.

```
g <- ggbiplot(PCWine, choices=2:3, obs.scale = 1, var.scale = 1, groups = wine$type,
              ellipse = TRUE,
               circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
print(g)
```

Looking at the biplot for the second and third principal components, we can see that there is much less clear separation. It does look like red wines have a higher spread in PC3, which indicates a greater variance in pH levels. There doesn't look to be much difference in red and white wines for PC2.

```
g <- ggbiplot(PCWine, choices=3:4, obs.scale = 1, var.scale = 1, groups = wine$type,
              ellipse = TRUE,
               circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
print(g)
```



Looking at PC3 and PC4, we see again that there is a larger spread in PC3 for red wines. There does not look to be any significant difference between redwine and white wines in PC4.

Combined wine - by quality

```
allQ <- wine
allQ$qualityCut <- cut(allQ$quality, c(0,5,7,9))
```

```
cor(allQ[,-13][,-13])
```

```
##                       fixed.acidity volatile.acidity citric.acid
## fixed.acidity                1.0000           0.2190      0.3244
## volatile.acidity             0.2190           1.0000     -0.3780
## citric.acid                  0.3244          -0.3780      1.0000
## residual.sugar              -0.1120          -0.1960      0.1425
## chlorides                    0.2982           0.3771      0.0390
## free.sulfur.dioxide         -0.2827          -0.3526      0.1331
## total.sulfur.dioxide        -0.3291          -0.4145      0.1952
## density                      0.4589           0.2713      0.0962
## pH                          -0.2527           0.2615     -0.3298
## sulphates                    0.2996           0.2260      0.0562
## alcohol                     -0.0955          -0.0376     -0.0105
## quality                     -0.0767          -0.2657      0.0855
##                       residual.sugar chlorides free.sulfur.dioxide
## fixed.acidity                -0.112    0.2982             -0.2827
## volatile.acidity             -0.196    0.3771             -0.3526
## citric.acid                   0.142    0.0390              0.1331
## residual.sugar                1.000   -0.1289              0.4029
## chlorides                    -0.129    1.0000             -0.1950
## free.sulfur.dioxide           0.403   -0.1950              1.0000
## total.sulfur.dioxide          0.495   -0.2796              0.7209
## density                       0.553    0.3626              0.0257
## pH                           -0.267    0.0447             -0.1459
## sulphates                    -0.186    0.3956             -0.1885
## alcohol                      -0.359   -0.2569             -0.1798
## quality                      -0.037   -0.2007              0.0555
##                       total.sulfur.dioxide density      pH sulphates
## fixed.acidity                      -0.3291  0.4589 -0.2527   0.29957
## volatile.acidity                   -0.4145  0.2713  0.2615   0.22598
## citric.acid                         0.1952  0.0962 -0.3298   0.05620
## residual.sugar                      0.4955  0.5525 -0.2673  -0.18593
## chlorides                          -0.2796  0.3626  0.0447   0.39559
## free.sulfur.dioxide                 0.7209  0.0257 -0.1459  -0.18846
## total.sulfur.dioxide                1.0000  0.0324 -0.2384  -0.27573
## density                             0.0324  1.0000  0.0117   0.25948
## pH                                 -0.2384  0.0117  1.0000   0.19212
## sulphates                          -0.2757  0.2595  0.1921   1.00000
## alcohol                            -0.2657 -0.6867  0.1212  -0.00303
## quality                            -0.0414 -0.3059  0.0195   0.03849
##                       alcohol quality
## fixed.acidity        -0.09545 -0.0767
```

```
## volatile.acidity     -0.03764 -0.2657
## citric.acid          -0.01049  0.0855
## residual.sugar       -0.35941 -0.0370
## chlorides            -0.25692 -0.2007
## free.sulfur.dioxide  -0.17984  0.0555
## total.sulfur.dioxide -0.26574 -0.0414
## density              -0.68675 -0.3059
## pH                    0.12125  0.0195
## sulphates            -0.00303  0.0385
## alcohol               1.00000  0.4443
## quality               0.44432  1.0000
```

Looking at the correlations between variables in our data, it looks like there are a number of correlations above .3, indicating that there is some sort of underlying structure in the data and fulfills the assumptions required for principal components analysis.

```
PCWineQ=prcomp(allQ[,-13][,-13],scale=TRUE)
PCWineQ$rotation
```

```
##                          PC1    PC2     PC3     PC4      PC5     PC6
## fixed.acidity        -0.2569  0.262 -0.4675  0.1440 -0.16536  0.0300
## volatile.acidity     -0.3949  0.105  0.2797  0.0801 -0.14777 -0.3827
## citric.acid           0.1465  0.144 -0.5881 -0.0555  0.23462  0.3622
## residual.sugar        0.3189  0.343  0.0755 -0.1125 -0.50792 -0.0633
## chlorides            -0.3134  0.270 -0.0468 -0.1653  0.39390 -0.4254
## free.sulfur.dioxide   0.4227  0.111  0.0990 -0.3033  0.24845 -0.2832
## total.sulfur.dioxide  0.4744  0.144  0.1013 -0.1322  0.22397 -0.1068
## density              -0.0924  0.555  0.0516 -0.1506 -0.33036  0.1546
## pH                   -0.2081 -0.153  0.4068 -0.4715  0.00146  0.5609
## sulphates            -0.2999  0.120 -0.1687 -0.5880  0.19325 -0.0201
## alcohol              -0.0589 -0.493 -0.2129 -0.0800 -0.11602 -0.1695
## quality               0.0875 -0.297 -0.2958 -0.4724 -0.45913 -0.2779
##                          PC7      PC8      PC9    PC10     PC11      PC12
## fixed.acidity        -0.3934  0.00116  0.42417 -0.2724 -0.27693 -0.335093
## volatile.acidity     -0.4451  0.31008 -0.12323  0.4939  0.14080 -0.082421
## citric.acid          -0.0477  0.44496 -0.24623  0.3304  0.22928  0.001347
## residual.sugar        0.0958  0.08194 -0.48802 -0.2072  0.00514 -0.451215
## chlorides             0.4733  0.37553 -0.04405 -0.2389 -0.19340 -0.043278
## free.sulfur.dioxide  -0.3627  0.12010  0.30140 -0.3034  0.48616 -0.000905
## total.sulfur.dioxide -0.2348  0.01128  0.00181  0.2948 -0.72016  0.064063
## density              -0.0133  0.04294  0.07108 -0.0768 -0.00332  0.715667
## pH                   -0.0793  0.36228  0.13666 -0.1124 -0.13908 -0.206763
## sulphates            -0.1702 -0.59222 -0.29740  0.0855  0.04722 -0.078200
## alcohol              -0.3389  0.22604 -0.41706 -0.4161 -0.19129  0.332012
## quality               0.2732  0.09305  0.35665  0.3078 -0.01808  0.008288
```

We run our PCA on correlations over covariances because our variables are on different scales and we do not want to weight variables with higher covariances differently.

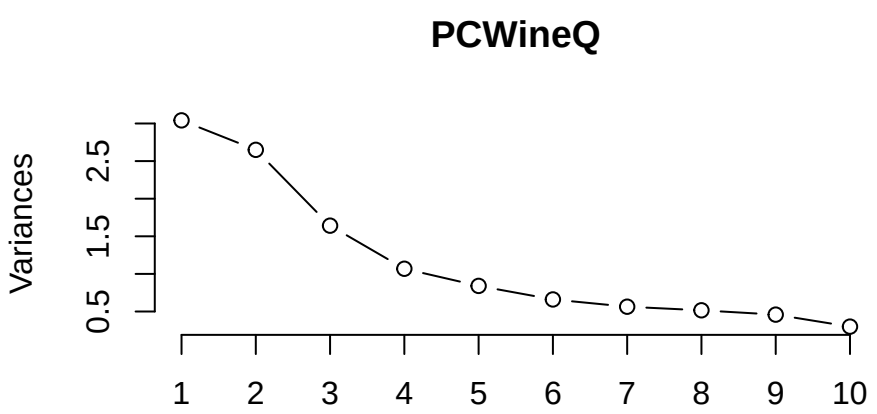
```
summary(PCWineQ) #prop variance explained by each component
```

```
## Importance of components:
```

```
##                           PC1   PC2   PC3    PC4   PC5   PC6   PC7   PC8
## Standard deviation     1.744 1.628 1.281 1.0337 0.917 0.813 0.751 0.718
## Proportion of Variance 0.253 0.221 0.137 0.0891 0.070 0.055 0.047 0.043
## Cumulative Proportion  0.253 0.474 0.611 0.7001 0.770 0.825 0.872 0.915
##                           PC9   PC10  PC11    PC12
## Standard deviation     0.6770 0.5468 0.477 0.18107
## Proportion of Variance 0.0382 0.0249 0.019 0.00273
## Cumulative Proportion  0.9534 0.9783 0.997 1.00000
```

Now we would like to choose how many principal components to keep. Looking at the variance explained explained by each principal component, we can see we need 4 principal components to explain 70% of the variation in the data. According to the eigenvalue >0.7 rule, we need 5 principal components.

```
screeplot(PCWineQ, type="l") #it is an L
```



Looking at the scree plot, there is a slight elbow at 4 principal components.

Since our overall goal is to reduce the number of variables and 2 of the 3 guidelines recommend 4 principal components, we will use 4 principal components.

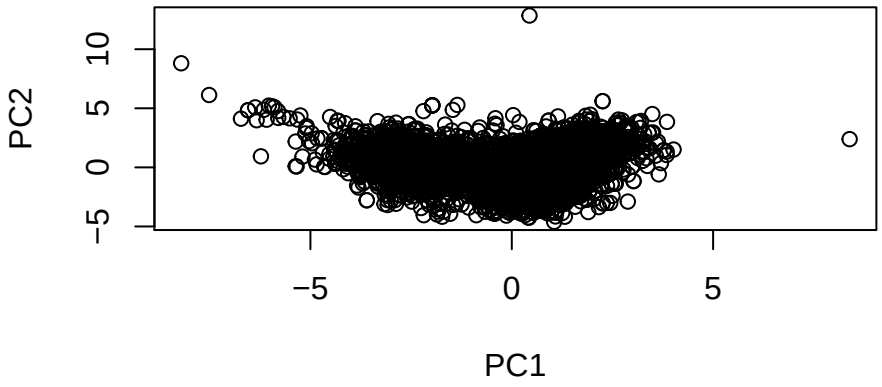
```
cor(allQ[,-13][,-13],PCWineQ$x) #loadings
```

```
##                         PC1    PC2     PC3     PC4      PC5     PC6
## fixed.acidity        -0.448  0.426 -0.5989  0.1488 -0.15160  0.0244
## volatile.acidity     -0.689  0.171  0.3583  0.0828 -0.13548 -0.3110
## citric.acid           0.255  0.235 -0.7535 -0.0574  0.21510  0.2944
## residual.sugar        0.556  0.558  0.0967 -0.1163 -0.46566 -0.0515
## chlorides            -0.547  0.439 -0.0599 -0.1709  0.36112 -0.3457
## free.sulfur.dioxide   0.737  0.181  0.1268 -0.3135  0.22778 -0.2301
## total.sulfur.dioxide  0.827  0.234  0.1298 -0.1367  0.20533 -0.0868
## density              -0.161  0.903  0.0661 -0.1557 -0.30287  0.1256
## pH                   -0.363 -0.249  0.5212 -0.4874  0.00134  0.4558
## sulphates            -0.523  0.195 -0.2161 -0.6079  0.17717 -0.0164
## alcohol              -0.103 -0.802 -0.2728 -0.0827 -0.10637 -0.1377
## quality               0.153 -0.483 -0.3790 -0.4884 -0.42092 -0.2258
##                          PC7      PC8      PC9    PC10     PC11      PC12
## fixed.acidity        -0.29542  0.00083  0.28718 -0.1490 -0.13211 -0.060674
## volatile.acidity     -0.33423  0.22273 -0.08343  0.2701  0.06717 -0.014924
## citric.acid          -0.03582  0.31963 -0.16671  0.1806  0.10938  0.000244
## residual.sugar        0.07191  0.05886 -0.33041 -0.1133  0.00245 -0.081700
## chlorides             0.35539  0.26975 -0.02982 -0.1306 -0.09226 -0.007836
```

```
## free.sulfur.dioxide  -0.27236  0.08627  0.20406 -0.1659  0.23193 -0.000164
## total.sulfur.dioxide -0.17632  0.00810  0.00123  0.1612 -0.34356  0.011600
## density              -0.00998  0.03085  0.04812 -0.0420 -0.00159  0.129583
## pH                   -0.05956  0.26023  0.09252 -0.0615 -0.06635 -0.037438
## sulphates            -0.12783 -0.42540 -0.20135  0.0467  0.02253 -0.014159
## alcohol              -0.25448  0.16237 -0.28236 -0.2275 -0.09126  0.060116
## quality               0.20512  0.06684  0.24146  0.1683 -0.00863  0.001501
```

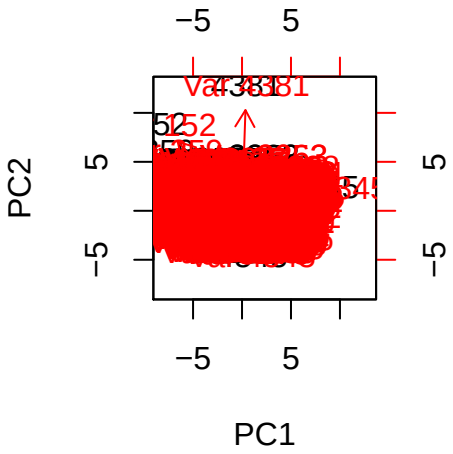
Looking at the loadings, the first PC represents a negative relationship between the variables fixed and volatile acidity, chlorides, pH, and sulphates and the variables residual sugar and sulfur dioxide. The second PC mainly capures the negative relationship between the varibles density, fixed acidity, residual sugar, and chlorides and the variables alcohol and quality. The third PC represents the negative relationship between the variables citric acid and fixed acidity and the variable pH. The 4th PC represents the positive relationship between pH, sulphates, and quality.

```
plot(PCWineQ$x)
```

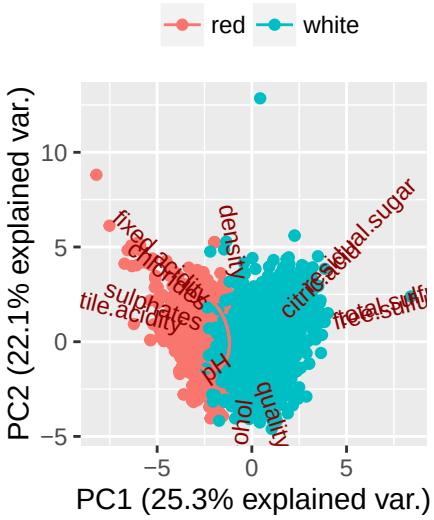



```
biplot(PCWineQ$x[,1:2],PCWineQ$x[,1:2])
```

```
## Warning in arrows(0, 0, y[, 1L] * 0.8, y[, 2L] * 0.8, col = col[2L], length
## = arrow.len): zero-length arrow is of indeterminate angle and so skipped

## Warning in arrows(0, 0, y[, 1L] * 0.8, y[, 2L] * 0.8, col = col[2L], length
## = arrow.len): zero-length arrow is of indeterminate angle and so skipped
```



```
g <- ggbiplot(PCWineQ, choices=1:2, obs.scale = 1, var.scale = 1, groups = allQ$type,
              ellipse = TRUE,
               circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
print(g)
```

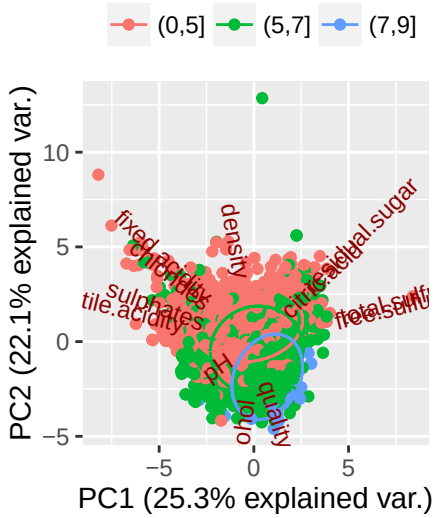


Looking at the biplot for all the points, we can see a clear separation of groups when looking at the first 2 principal components. The red wines have a lower value for PC1, which would indicate higher levels of fixed acidity, volatile acidity, chlorides, and sulphates and lower levels of residual sugar and free and total sulfur dioxide.

```
g <- ggbiplot(PCWineQ, choices=2:3, obs.scale = 1, var.scale = 1, groups = allQ$type,
              ellipse = TRUE,
               circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
print(g)
```

Looking at PC3 and PC4, we see again that there is a larger spread in PC3 for red wines. There does not look to be any significant difference between redwine and white wines in PC4.

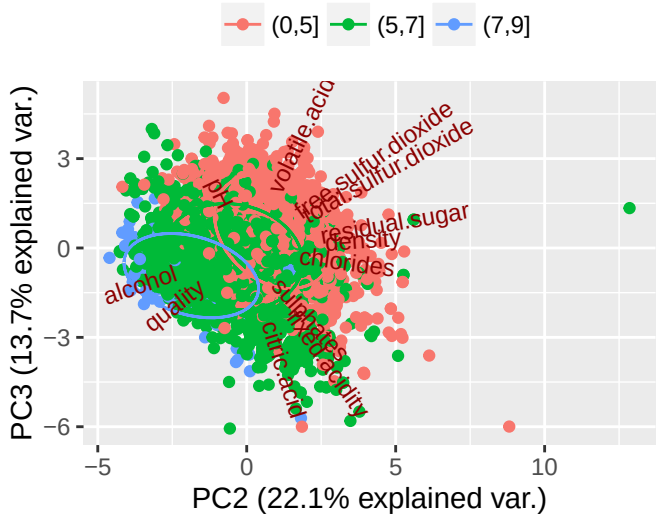
```
g <- ggbiplot(PCWineQ, choices=1:2, obs.scale = 1, var.scale = 1, groups = allQ$qualityCut,
              ellipse = TRUE,
               circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
print(g)
```



Looking at the biplot between the first 2 PCs grouped based on quality, we can see that all groups have similar values in PC1. However, there is a clear trend in PC2 where higher quality wines have lower values of PC2.

```
g <- ggbiplot(PCWineQ, choices=2:3, obs.scale = 1, var.scale = 1, groups = allQ$qualityCut,
              ellipse = TRUE,
               circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
```

```
                legend.position = 'top')
print(g)
```



Looking at the plot between PC2 and PC3, we can see once again that higher quality wines have lower values of PC2 and lower quality wines have higher values of PC2.

So, PCA allows us to reduce the dimensionality of our data.

Cluster Analysis

Red and White Wines Combined

We are interested in determining whether there are any variables that have a strong influence on the quality of the wine. To this end, we use cluster analysis to analyze both the red and white wines together.

The Cluster Method:

Cluster analysis attempts to group objects into clusters in such a way that two objects in one group are more similar than objects in another one. The classification rules used to group the objects are subjective and are determined based on the statistician's goals. Below, we present the best cluster grouping we found.

How it works: First select a way to measure the distance (dissimilarity) between all pairs of observations (i.e. a similarity measure). Get the dissimilarity by looking at how the observations compare in the variables measured. The most common similarity measures are Euclidean, statistical (standardized), and Mahalanobis. Then, decide which clustering algorithm to use based on the chosen distance measure. There are two main clustering techniques: Hierarchical clustering and K-means (nonhierarchical) clustering. Hierarchical clustering starts with each observation in its own cluster, and then slowly merges them with the closest observations to form larger and larger clusters. Finally, use a dendogram to gauge the best cluster number to partition the data into (look at big height jumps and evenly distributed clusters). In K-means clustering, the number of clusters are know beforehand. K-means can be used to verify or improve upon the hierarchical clustering results.

Assumptions:

Correlations among the variables and multicollinearity can alter the solution, so eliminate any highly correlated variables. The data must be numerical, or else use another distance measure such as Pearson's or Spearman's.

It is very important to chose the correct similarity measure. The sample size must also be large enough to represent underlying structure. Cluster analysis is very sensitive to outliers, and these must be examined. If the outlier is not part of a relevant group, it can be removed.

Our wines data was ideal to use clustering because of the following reasons:

- Large enough data set of almost 6,500 observations.
- There were no highly correlated variables except for total.sulfur.dioxide and density. They were eliminated from the analysis.
- There were also very few outliers (130) compared to a set of almost 6,500 observations.
- Our data was all numerical except for color. We effectively incorporated color into our analysis by assigning Red Wine = 0 and White Wine = 1. This variable also made cluster formation clearer.
- There we two clear groups from the start, red and white wine, which made clustering the data and comparing the clusters easier.

Our Analysis:

Custer analysis is very sensitive to outliers. Thus, we created bivaraite boxplots on randomly chosen pairs of variables and eliminated outliers in our data pre-processing.

We have already performed an outlier analysis for the combined red and white dataset which we will use for clustering. First notice that there are over 6,000 entries (6368 to be exact) in our combined data set after acounting for outliers, which is more than enough for our analysis to represent the underlying structure of wines. We examine the correlation between the varibles to eliminate any highly correlated variables.

We are removing the outliers as discussed previously.

```
cor(AllWinesB)
```

```
##                       fixed.acidity volatile.acidity citric.acid
## fixed.acidity                1.0000           0.2165      0.2616
## volatile.acidity             0.2165           1.0000     -0.4124
## citric.acid                  0.2616          -0.4124      1.0000
## residual.sugar              -0.0954          -0.2022      0.1632
## chlorides                    0.2996           0.3878      0.0055
## free.sulfur.dioxide         -0.2616          -0.3520      0.1677
## total.sulfur.dioxide        -0.2895          -0.4119      0.2394
## density                      0.4287           0.2639      0.0559
## pH                          -0.2399           0.2637     -0.3184
## sulphates                    0.2692           0.2345      0.0165
## alcohol                     -0.1123          -0.0433     -0.0155
## quality                     -0.0934          -0.2537      0.0801
## color                       -0.4569          -0.6616      0.2441
##                       residual.sugar chlorides free.sulfur.dioxide
## fixed.acidity                -0.0954    0.2996             -0.2616
## volatile.acidity             -0.2022    0.3878             -0.3520
## citric.acid                   0.1632    0.0055              0.1677
## residual.sugar                1.0000   -0.1273              0.4163
## chlorides                    -0.1273    1.0000             -0.1909
## free.sulfur.dioxide           0.4163   -0.1909              1.0000
## total.sulfur.dioxide          0.4999   -0.2723              0.7158
## density                       0.5662    0.3673              0.0613
## pH                           -0.2813    0.0626             -0.1629
```

```
## sulphates                    -0.1845    0.3722             -0.1834
## alcohol                      -0.3709   -0.2674             -0.1869
## quality                      -0.0419   -0.2025              0.0606
## color                         0.3454   -0.5168              0.4649
##                       total.sulfur.dioxide density      pH sulphates
## fixed.acidity                      -0.2895  0.4287 -0.2399   0.26922
## volatile.acidity                   -0.4119  0.2639  0.2637   0.23446
## citric.acid                         0.2394  0.0559 -0.3184   0.01649
## residual.sugar                      0.4999  0.5662 -0.2813  -0.18451
## chlorides                          -0.2723  0.3673  0.0626   0.37222
## free.sulfur.dioxide                 0.7158  0.0613 -0.1629  -0.18337
## total.sulfur.dioxide                1.0000  0.0721 -0.2579  -0.26318
## density                             0.0721  1.0000  0.0299   0.24337
## pH                                 -0.2579  0.0299  1.0000   0.21841
## sulphates                          -0.2632  0.2434  0.2184   1.00000
## alcohol                            -0.2757 -0.7232  0.1197  -0.00252
## quality                            -0.0456 -0.3244  0.0283   0.03730
## color                               0.6901 -0.3733 -0.3555  -0.47924
##                        alcohol quality   color
## fixed.acidity        -0.11229 -0.0934 -0.4569
## volatile.acidity     -0.04334 -0.2537 -0.6616
## citric.acid          -0.01547  0.0801  0.2441
## residual.sugar       -0.37093 -0.0419  0.3454
## chlorides            -0.26739 -0.2025 -0.5168
## free.sulfur.dioxide  -0.18687  0.0606  0.4649
## total.sulfur.dioxide -0.27569 -0.0456  0.6901
## density              -0.72319 -0.3244 -0.3733
## pH                    0.11969  0.0283 -0.3555
## sulphates            -0.00252  0.0373 -0.4792
## alcohol               1.00000  0.4534  0.0331
## quality               0.45345  1.0000  0.1180
## color                 0.03313  0.1180  1.0000
```

There are only two pairs of highly correlate variables (>0.7): free.sulfur.dioxide and total.sulfur.dioxid, alcohol and density. We eliminate total.sulfur.dioxide and density. Note that alcohol is the most correlated variable with quality (0.45). Also, color is very low correlated with quality (0.11) byt highly with total.sulfur.dioxide and volatile.acidity.

```
AllWines3= AllWinesB[-c(7,8)]
#pairs(AllWines3) this pairs plot takes a bit of time to complie
```

It looks like there are no outliers present in our data.

Now we need to determine whether to standarize our variables.

```
summary(AllWines3)
```

```
##  fixed.acidity   volatile.acidity  citric.acid    residual.sugar
##  Min.   : 3.80   Min.   :0.080    Min.   :0.000   Min.   : 0.60
##  1st Qu.: 6.40   1st Qu.:0.230    1st Qu.:0.240   1st Qu.: 1.80
##  Median : 7.00   Median :0.290    Median :0.310   Median : 3.10
##  Mean   : 7.13   Mean   :0.335    Mean   :0.315   Mean   : 5.48
##  3rd Qu.: 7.60   3rd Qu.:0.400    3rd Qu.:0.390   3rd Qu.: 8.20
```

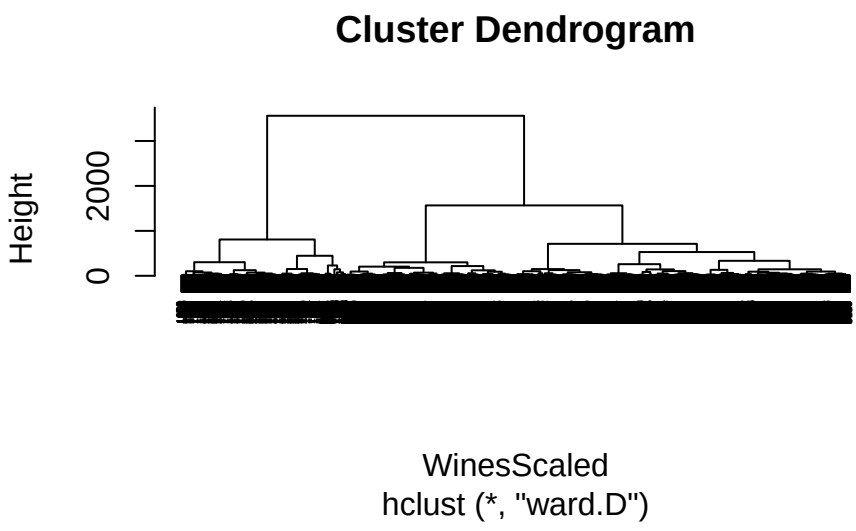
```
##  Max.   :11.50   Max.   :1.000    Max.   :1.000   Max.   :26.05
##    chlorides     free.sulfur.dioxide       pH         sulphates
##  Min.   :0.009   Min.   :  1.0       Min.   :2.72   Min.   :0.220
##  1st Qu.:0.038   1st Qu.: 17.0       1st Qu.:3.11   1st Qu.:0.430
##  Median :0.047   Median : 29.0       Median :3.21   Median :0.500
##  Mean   :0.055   Mean   : 30.8       Mean   :3.22   Mean   :0.528
##  3rd Qu.:0.063   3rd Qu.: 42.0       3rd Qu.:3.32   3rd Qu.:0.600
##  Max.   :0.467   Max.   :146.5       Max.   :4.01   Max.   :1.980
##     alcohol        quality         color
##  Min.   : 8.0   Min.   :3.00   Min.   :0.000
##  1st Qu.: 9.5   1st Qu.:5.00   1st Qu.:1.000
##  Median :10.3   Median :6.00   Median :1.000
##  Mean   :10.5   Mean   :5.82   Mean   :0.768
##  3rd Qu.:11.3   3rd Qu.:6.00   3rd Qu.:1.000
##  Max.   :14.2   Max.   :9.00   Max.   :1.000
```

Our summary reveals that our variables have very different scales and should be standarized. Fixed.acidity, for example, ranges from 3.8 to 11.5 while volatile.acidity from 0.08 to 1.

```
WinesScaled = dist(scale(AllWines3))
```

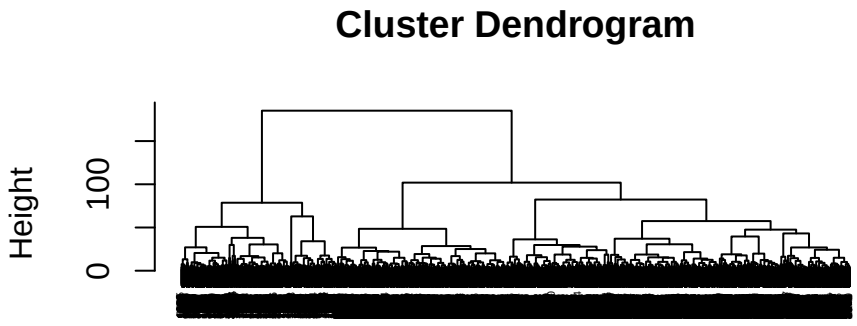
After analyzing different methods we settled for ward.D.

```
wardAll = hclust (WinesScaled, method = "ward.D")
plot(wardAll, cex= 0.3)
```



The ward.D solution is much better because of evenly distributed groups and high height differences. Let's see whether the D2 solution is best.

```
ward2 = hclust (WinesScaled, method = "ward.D2")
plot(ward2, cex=0.3)
```



The ward.D solution is best because it gives clear cutoffs and is widely spread among the data points. We now cut the ward solution. Three to four clusters seems more appropriate because of the high height jump at around height = 100.

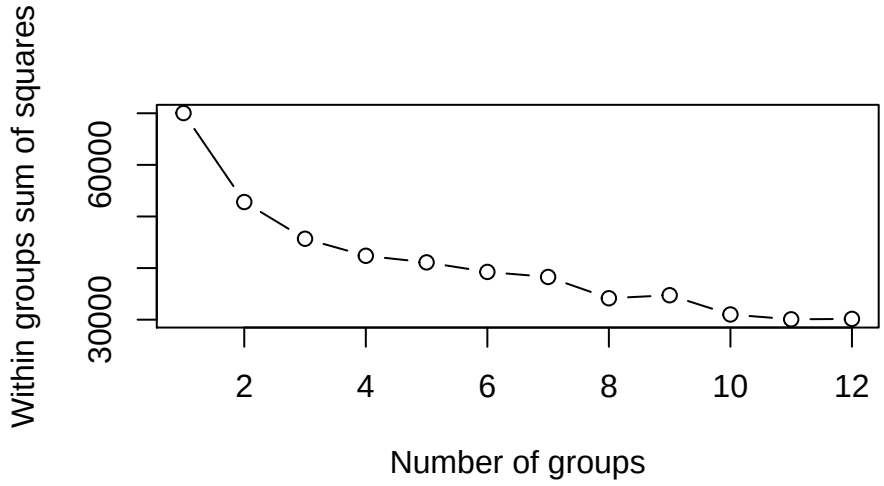
```
wardSolAll = cutree(wardAll, k = 4)
summary(as.factor(wardSolAll))
```

```
##    1    2    3    4
##  925  646 3220 1577
```

There is a fair spread of the observations among the 4 clusters, which is good because we don't have a particularly small group.

Before we decide on the final number of clusters, we perform a K-means solution.

```
set.seed(100)
wss = rep(0, 12)
for(i in 1:12){ wss[i] <- sum(kmeans(scale(AllWines3),centers= i)$withinss)}
plot(1:12, wss, type = "b", xlab = "Number of groups", ylab = "Within groups sum of squares")
```



An elbow happens right at k = 4. We will perform our analysis with 4 clusters.

Now I want to see the overlap of both the k-means and the hierarchical solutions to check how they compare.

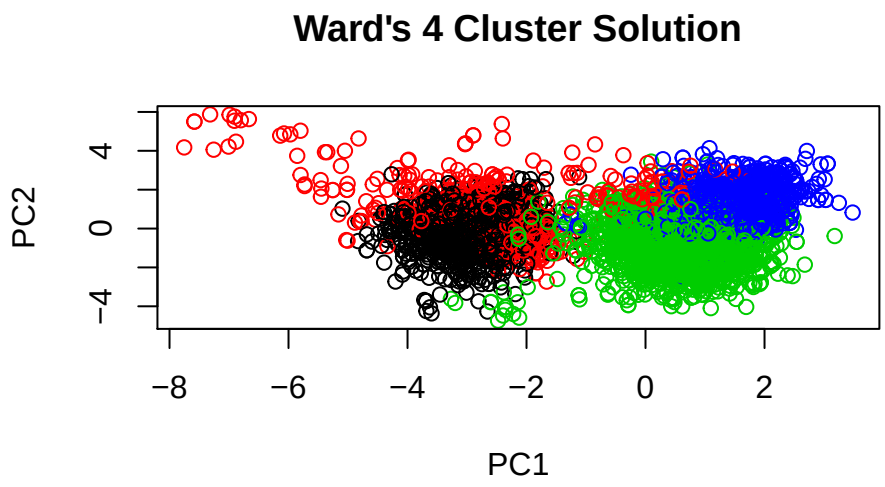
```
set.seed(100)
kSolAll = kmeans(scale(AllWines3), centers =4)
tally(kSolAll$cluster~wardSolAll, format = "count")
```

```
##                wardSolAll
## kSolAll$cluster    1    2    3    4
##               1    1   41  206 1296
##               2    0    9 1694   67
##               3    2   59 1309  214
##               4  922  537   11    0
```

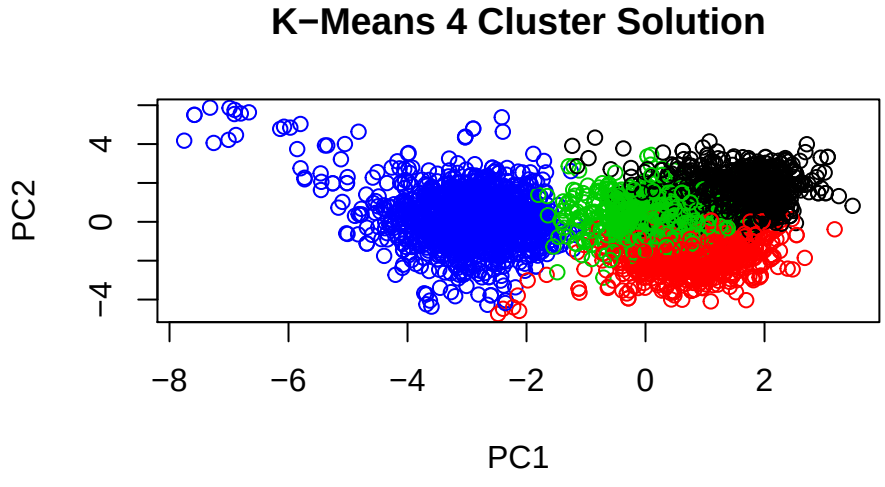
We see some overlap in our solutions. Ksol cluster 1 overlaps the most with ward solution cluster 4. Ksol cluster 2 overlaps almost entirely with ward solution 3. Ksol cluster 3 olveraps almost entirely with ward sol cluster 3. Ksol clutser 4 overlaps with ward sol cluster 1 and 2.

To make the decision between the two solutions, we plot them and see which gives a clearer distinction between groups.

```
wardPCAallWines = prcomp(AllWines3, scale = TRUE)
plot(wardPCAallWines$x[,1:2], xlab = "PC1", ylab = "PC2", main = "Ward's 4 Cluster Solution", col= wardS
```



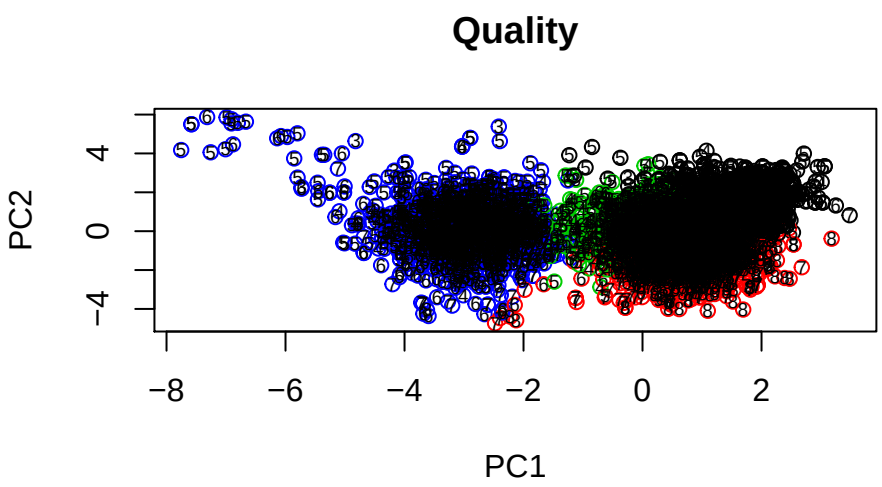

```
plot(wardPCAallWines$x[,1:2], xlab = "PC1", ylab = "PC2", main = "K-Means 4 Cluster Solution", col= kSol
```



Our K-Means 4 Cluster solution has a clearer distinction between the clusters. Therefore, we proceed with K-Means 4-Cluster solution in our analysis.

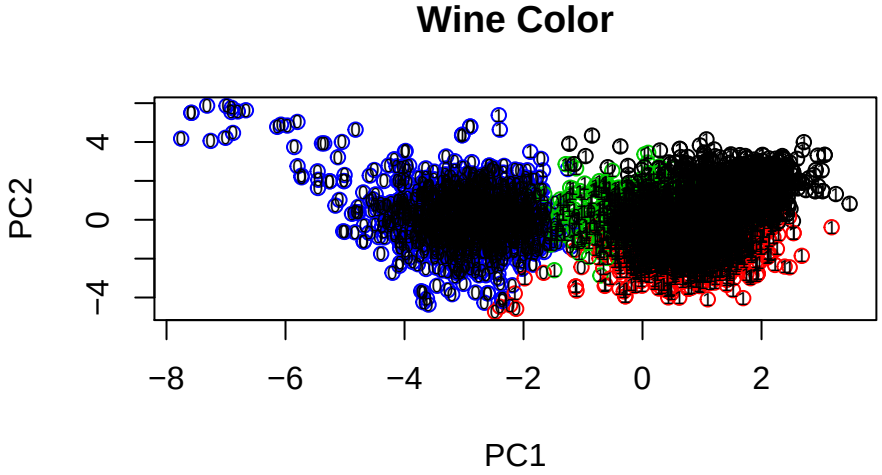
First, verify how well the K-means solution separates different wine quality and both red and white wines.

```
plot(wardPCAallWines$x[,1:2], xlab = "PC1", ylab = "PC2", main = "Quality", col= kSolAll$cluster)
text(wardPCAallWines$x[,1:2], labels = AllWines3$quality, cex = 0.6)
```



The plot is not too clear but it seems like the red cluster contains the highest wine quality while the rest contain average quality wine.

```
plot(wardPCAallWines$x[,1:2], xlab = "PC1", ylab = "PC2", main = "Wine Color", col= kSolAll$cluster)
text(wardPCAallWines$x[,1:2], labels = AllWines3$color, cex = 0.6)
```



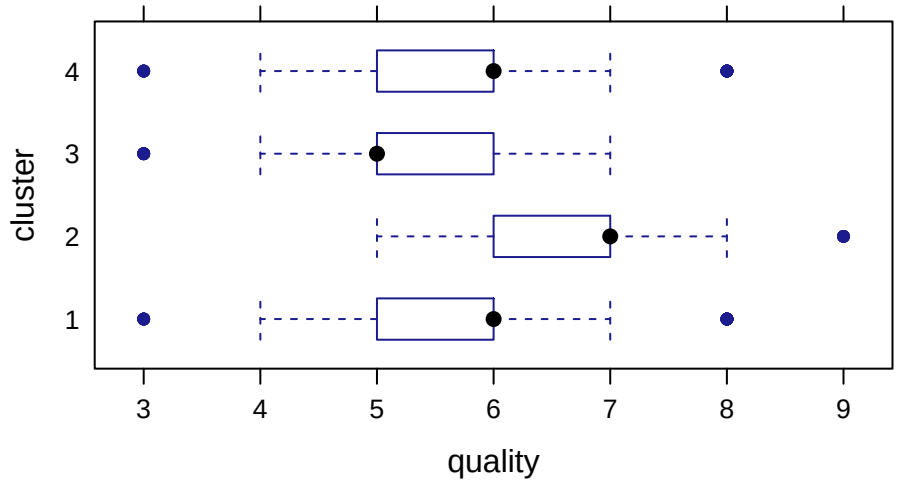
The blue cluster is almost entirely red wine, while clusters red, green, and black contain almost entirely white wine. Overall, this solution does a good job at separating wines by both color and by quality.

We now explain some major features of each group.

We create a new column corresponding to the 4 clusters so that we can compare the variables of interest with each of the clusters.

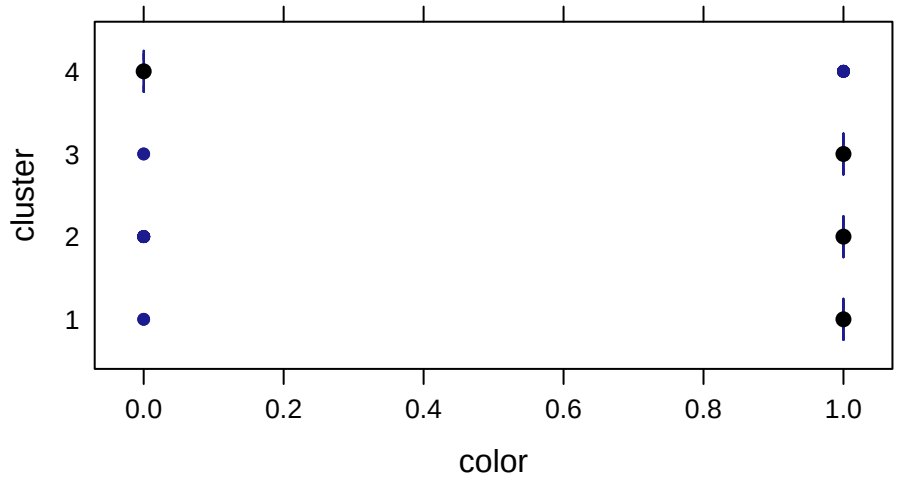
```
AllWines3$cluster = kSolAll$cluster
```

```
bwplot(cluster~quality, data = AllWines3)
```



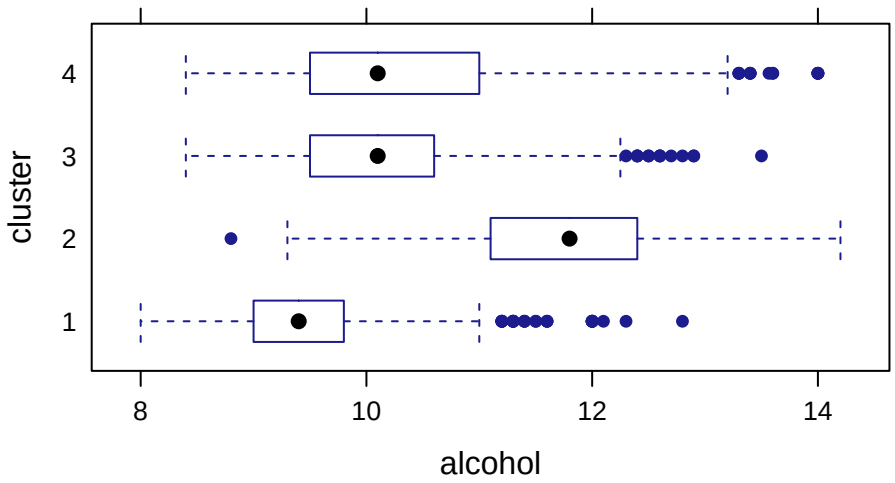
The boxplot shows that all the clusters have roughly the same quality distribution except for cluster 2, which has a higher wine quality on average.

```
bwplot(cluster~color, data = AllWines3)
```



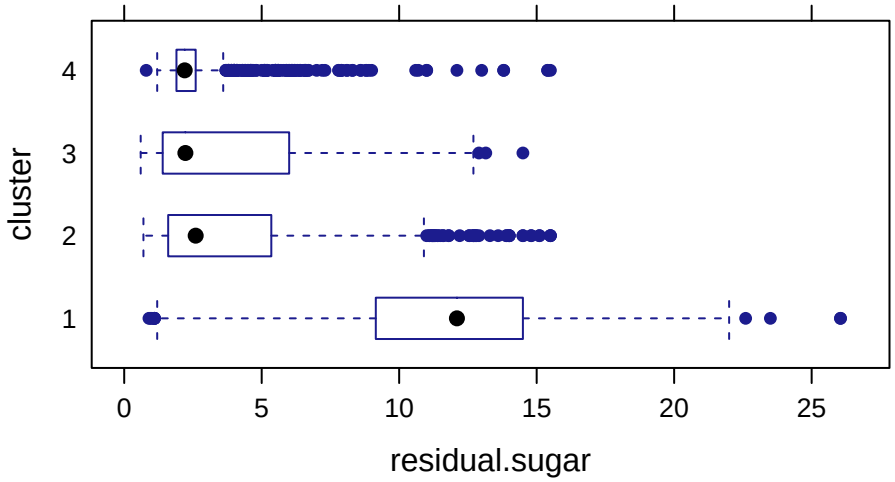
Note that 0 = Red Wine and 1 = white Wine. Most clusters contain a huge amount of white wine except cluster 4. This is expected because the white wine list contains about four times as many observations as red wine. Also note that the cluster with the highest quality corresponds with a white wine cluster.

```
bwplot(cluster~alcohol, data = AllWines3)
```



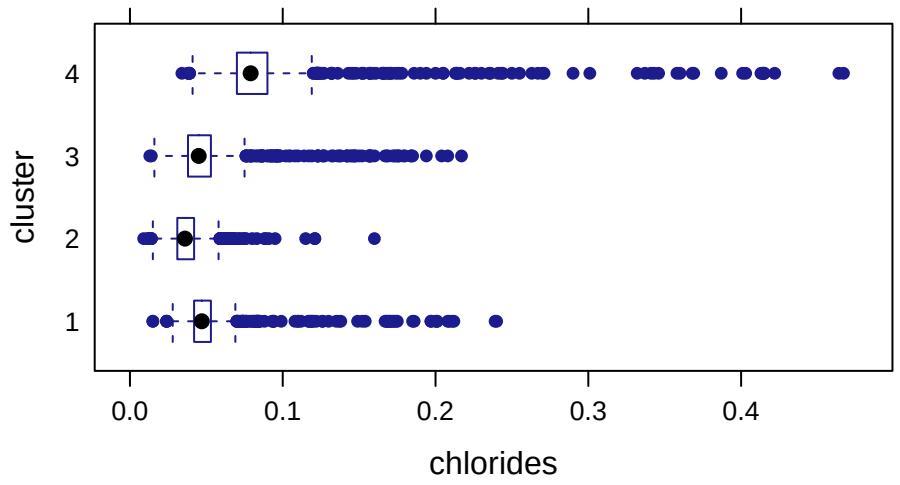
Cluster 2 has the highest alcohol content. out of all the clusters. This cluster also corresponds to the highest wine quality. Cluster 1 has the lowest alcohol content.

```
bwplot(cluster~residual.sugar, data = AllWines3)
```



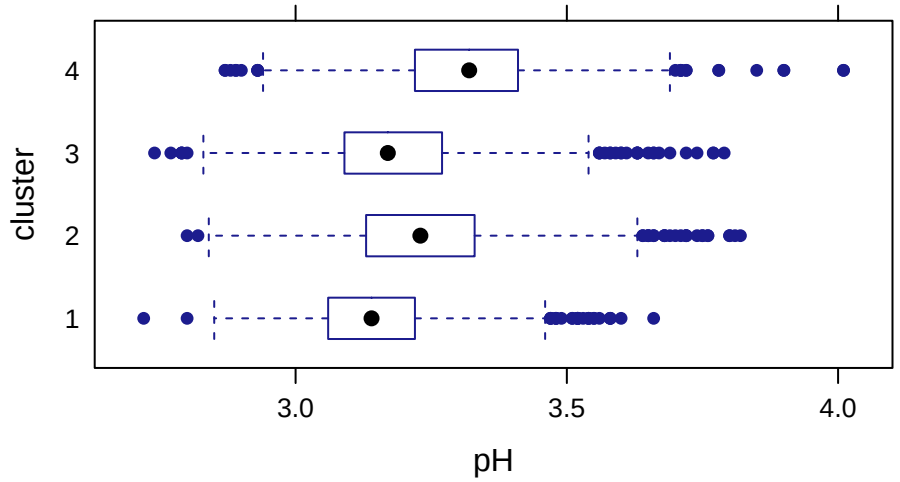
All the clusters have a low residual sugar content except for cluster 1, which is very high. Cluster 1 also has the lowest alcohol content.

```
bwplot(cluster~chlorides, data = AllWines3)
```



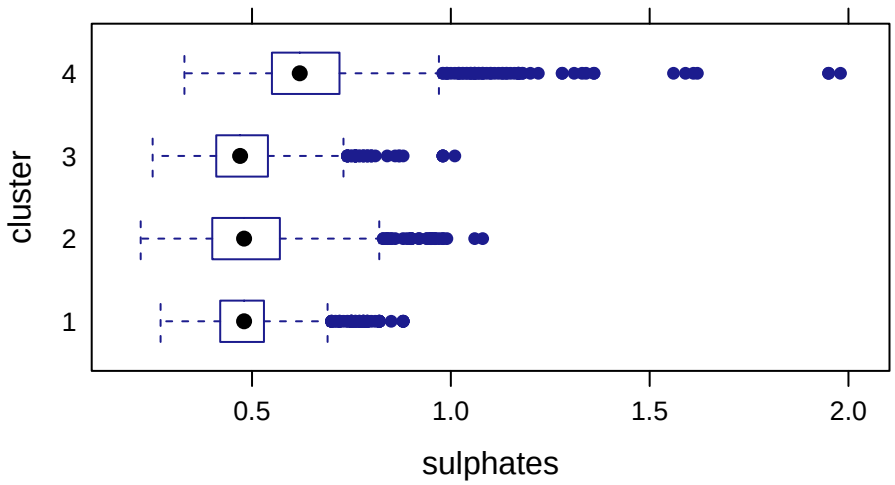
All clusters have a low chloride content, but cluster 4 seems to have a relatively higher one. CLuster 2 seems to have the lowest.

```
bwplot(cluster~pH, data = AllWines3)
```



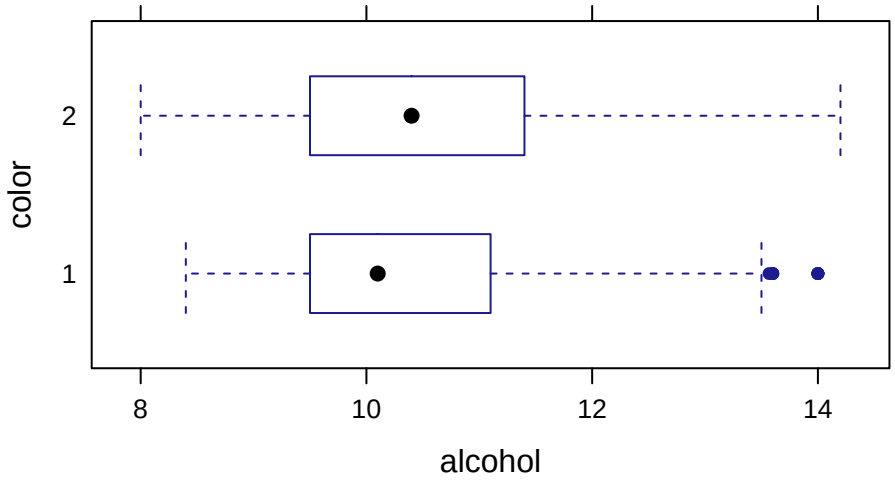
They all have a similar pH level.

```
bwplot(cluster~sulphates, data = AllWines3)
```



They all have similar sulphate distribution except for cluster 4 which is a bit higher.

```
bwplot(color~alcohol, data = AllWines3)
```



In conclusion, our K-means clustering solution was the best at creating 4 clusters on our data. Cluster 2 has the highest wine quality out of all clusters, as well as highest alcohol content. This group also corresponds to white wine. Our findings suggest that there is a substantial relation between alcohol level and quality of wine. Moreover, it seems like white wine scores higher than red wine in terms of both quality and alcohol content.

Random Forest

Random forests is an example of an ensemble method (combine methods to improve prediction). We will generate many predictions for each observation and use majority voting to determine final class. Random forest is not sensitive to outliers. And we will perform Random Forest for red and white wine separately.

White wine

We classify the white wine into good, bad, and normal based on their quality. Those with quality rating less than 6 are 'bad', more than 6 'good' and 'normal' otherwise. We created a new variable 'category' to record these entries.

```
whitewinef <- whitewineA
whitewinef$category <- ifelse(whitewinef$quality < 6, 'bad', 'good')
whitewinef$category[whitewinef$quality == 6] <- 'normal'
whitewinef$category <- as.factor(whitewinef$category)
```

This is the break down of category for white wine

```
whitewinef <- subset(whitewinef, select=-quality)
table(whitewinef$category)
```

```
##
##    bad   good normal
##   1640   1060   2198
```

Before we build our model, let's separate our data into testing and training sets.

```
samp <- sample(nrow(whitewinef), 0.7 * nrow(whitewinef))
train1 <- whitewinef[samp, ]
test1 <- whitewinef[-samp, ]
train1$category <- factor(train1$category)
```

This will place 70% of the observations in the original dataset into train and the remaining 30% of the observations into test.

Building the model

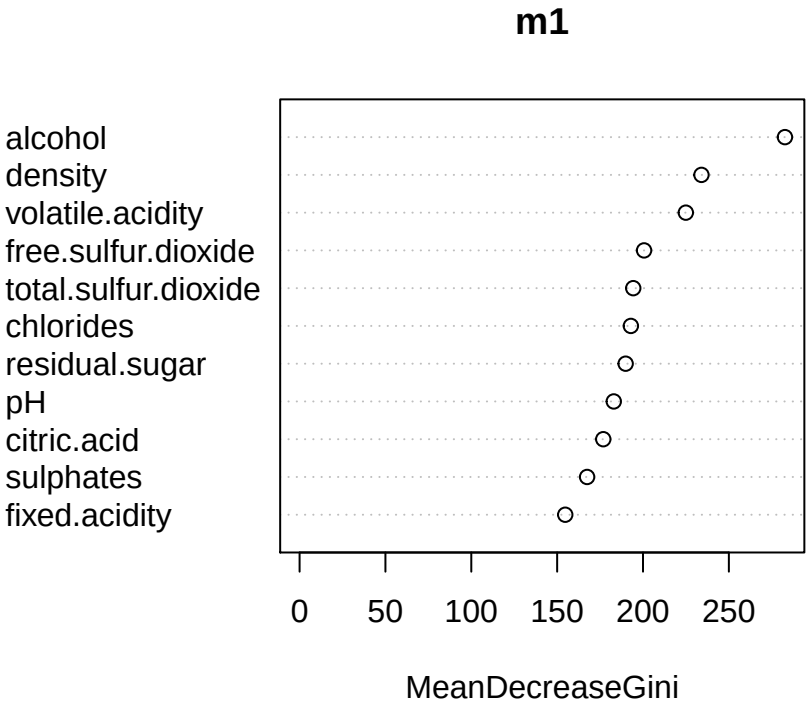
We use all our continuous variable to build our forest.

```
m1 <- randomForest(as.factor(category) ~ ., data = train1)#; m1
```

We can use ntree and mtry to specify the total number of trees to build (default = 500), and the number of predictors to randomly sample at each split respectively. Let's take a look at the model.

We can see that 500 trees were built, and the model randomly sampled 3 predictors at each split. It also shows a matrix containing prediction vs actual, as well as classification error for each class.

```
varImpPlot(m1)
```



Alcohol and density seem to be the most important predictors based on the above graph.

Let's test the model on the test data set.

```
pred1 <- predict(m1, newdata = test1)
table(pred1, test1$category)
```

```
##
## pred1    bad good normal
##   bad    361    5    105
##   good     8  212     65
##   normal 121   81    512
```

We can test the accuracy as follows:

```
(352+174+512) / nrow(test1)
```

```
## [1] 0.706
```

We have achieved around 71% accuracy. This means that our model has correctly predicted the category in around 71% of cases on a new data.

Red wine

We know perform the same analysis for red wine.

We classify the wines into good, bad, and normal based on their quality by the same standard as for white wine.

```
redwinef <- redwineA
```

```
redwinef$category <- ifelse(redwinef$quality < 6, 'bad', 'good')
redwinef$category[redwinef$quality == 6] <- 'normal'
redwinef$category <- as.factor(redwinef$category)
```

We look at the distribution:

```
table(redwinef$category)
```

```
##
##    bad   good normal
##    744    217    638
```

Again, before we build our model, let's separate our data into testing and training sets.

```
samp <- sample(nrow(redwinef), 0.7 * nrow(redwinef))
train2 <- redwinef[samp, ]
test2 <- redwinef[-samp, ]
```

This will place 70% of the observations in the original dataset into train and the remaining 30% of the observations into test.

Building the model

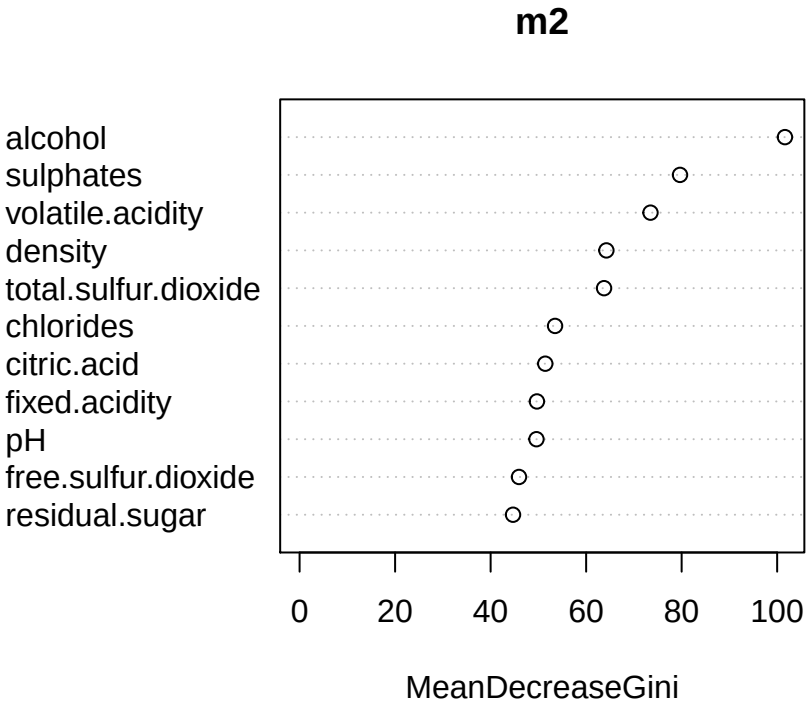
We use all our continuous variable to build our forest.

```
m2 <- randomForest(category ~ . - quality, data = train2)#; m2
```

We can use ntree and mtry to specify the total number of trees to build (default = 500), and the number of predictors to randomly sample at each split respectively. Let's take a look at the model.

We can see that 500 trees were built, and the model randomly sampled 3 predictors at each split. It also shows a matrix containing prediction vs actual, as well as classification error for each class.

```
varImpPlot(m2)
```



Alcohol and sulphates seem to be the most important predictors according to the above graph.

Let's test the model on the test data set.

```
pred2 <- predict(m2, newdata = test2)
table(pred2, test2$category)
```

```
##
## pred2    bad good normal
##   bad    182    5     42
##   good     1   27     11
##   normal  47   33    132
```

We can test the accuracy as follows:

```
(178+40+131) / nrow(test2)
```

```
## [1] 0.727
```

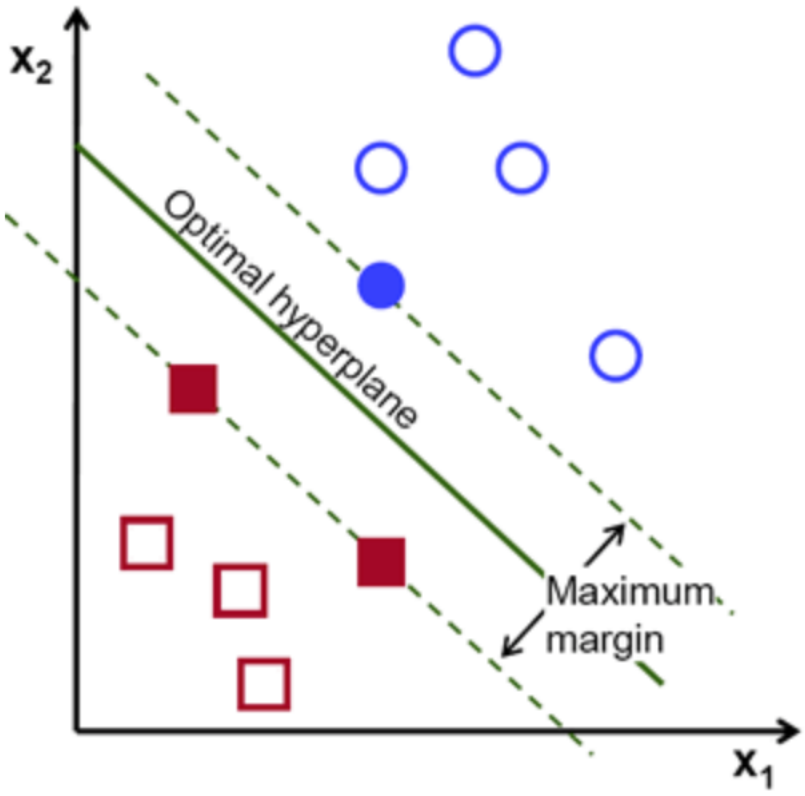
We have achieved 72.7% accuracy. This means that our model has correctly predicted the category in around 72% of cases on a new data. Alcohol level seem to be the most influential variable for both white and red wine.

Random Forest		
wine type	Accuracy level	test set size(30% of our data)
red wine	73%	480
white wine	71%	1470

So Random Forest gives us an accuracy level of around 70% which is not bad. This is expected for a greedy algorithm. We will now turn to a more powerful machine learning algorithm - the Support Vector Machine.

Expository Topic: Support Vector Machine (SVM)

SVM project data into higher dimension to find the best classification rule. The goal in an SVM is to find the separating hyperplane between classes with maximal distance between classes and the plane. They are sensitive to outliers so we applied our outlier analysis from our data pre-processing. Support vectors are the data points that are closest to the decision surface. SVMs maximize the margin around the separating hyperplane. Simply put, given a labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. The following image illustrates the algorithm.



{Separate hyperplane with classes}

It may be viewed as a function where our input are the input attributes of the problems which are mapped to a new set of quantities through the algorithm called features. We will also let φ denote the feature mapping, which maps from the attributes to the features. Given a feature mapping φ, we define the corresponding **Kernel** to be

$$K(x, z) = \varphi(x)^T \varphi(z).$$

Our algorithm is now learning using the features of φ. We usually use the radial basis. **Gamma** is the parameter needed for all kernels except linear. The **cost** of constraints violation (default: 1)—it is the 'C'-constant of the regularization term in the Lagrange formulation that goes in the algorithm.

We are using SVM because we want to obtain better result when compared to random forest. Due to their higher flexibility and nonlinear learning capabilities, SVM has gain importance in the field of data mining.

Here is our SVM analysis. We are using all of our continuous variables to predict which category a wine sample belongs to.

White Wine

We classify the wines into good, bad, and normal based on their quality as before.

```
refinedWhite$category <- ifelse(refinedWhite$quality < 6, 'bad', 'good')
refinedWhite$category[refinedWhite$quality == 6] <- 'normal'
refinedWhite$category <- as.factor(refinedWhite$category)
```

We run the svm algorithm after removing outliers.

```
whiteQ <- subset(refinedWhite, select=-quality)
x <- subset(whiteQ, select=-c(category))
y <- refinedWhite$category
svm_model <- svm(category ~ ., data=whiteQ)
summary(svm_model)
```

```
##
## Call:
## svm(formula = category ~ ., data = whiteQ)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.0909
##
## Number of Support Vectors:  4107
##
##  ( 1909 1254 944 )
##
##
## Number of Classes:  3
##
## Levels:
##  bad good normal
```

```
pred1 <- predict(svm_model,x)
table(pred1,y)
```

```
##         y
## pred1     bad good normal
##   bad    1111   30    363
##   good     20  482    167
##   normal  498  547   1659
```

From the above confusion matrix we see that we are predicting the category in 66.7% which is not better than the random forest result.

```
(1111+482+1659)/4877*100
```

```
## [1] 66.7
```

So we will now tune our model to try find the optimal hyperplane. In other words we hope to find the best cost and gamma. We choose cost and gamma at random here.

```
svm_tune <- tune(svm, train.x=x, train.y=y,
              kernel="radial", ranges=list(cost=10^(-1:2), gamma=c(.5,1,2)))

print(svm_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10     1
##
## - best performance: 0.29
```

After we have found the best cost and gamma, we can create svm model again and try to run again. It would seem that cost=100 and gamma = 1 would be best.

```
### change parameter
svm_model_after_tune <- svm(category ~ ., data=whiteQ, kernel="radial", cost=100, gamma=1)
summary(svm_model_after_tune)
```

```
##
## Call:
## svm(formula = category ~ ., data = whiteQ, kernel = "radial",
##     cost = 100, gamma = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##       gamma:  1
##
## Number of Support Vectors:  3870
##
##  ( 1733 1308 829 )
##
##
## Number of Classes:  3
##
## Levels:
##  bad good normal
```

We run our confusion matrix again

```
pred <- predict(svm_model_after_tune,x)
table(pred,y)
```

```
##         y
## pred      bad good normal
##   bad    1629    0      0
##   good      0 1059      0
##   normal    0    0   2189
```

We see that we have achieved an accuracy level of 100 %! We do get great result but the learning time takes a lot time the larger the sample size.

Red Wine

We carry out a similar analysis for red wine as we did above.

We classify the red wine into good, bad, and normal based on their quality as we did before.

```
refinedRed$category <- ifelse(refinedRed$quality < 6, 'bad', 'good')
refinedRed$category[refinedRed$quality == 6] <- 'normal'
refinedRed$category <- as.factor(refinedRed$category)
```

We run our SVM model on all continuous variables with category as our response variable.

```
redQ <- subset(refinedRed, select=-quality)
x <- subset(redQ, select=-c(category))
y <- refinedRed$category
svm_model <- svm(category~ .,type='C-classification', data=redQ)
summary(svm_model)
```

```
##
## Call:
## svm(formula = category ~ ., data = redQ, type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.0909
##
## Number of Support Vectors:  1222
##
##  ( 464 563 195 )
##
##
## Number of Classes:  3
##
## Levels:
##  bad good normal
```

```
pred1 <- predict(svm_model,x)
table(pred1,y)
```

```
##         y
## pred1    bad good normal
##   bad    600   14    184
##   good     4   89     21
##   normal 122  109    420
```

Accuracy : 71%. This is around what we would expect with a random forest.

```
(600+89+420)/1563*100
```

```
## [1] 71
```

We then tune our model to find ther best cost and gamma. We choose cost and gamma at random here first.

```
svm_tune <- tune(svm, train.x=x, train.y=y,
              kernel="radial", ranges=list(cost=10^(-1:2), gamma=c(.5,1,2)))

print(svm_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 0.302
```

It would seem cost=1 and gamma=0.5 would give us better result. So we run our improved model based on these new parameters.

```
### change parameter
svm_model_after_tune2 <- svm(category ~ ., data=redQ, kernel="radial", cost=1, gamma=0.5)
summary(svm_model_after_tune2)
```

```
##
## Call:
## svm(formula = category ~ ., data = redQ, kernel = "radial", cost = 1,
##     gamma = 0.5)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  1349
##
```

```
##  ( 574 575 200 )
##
##
## Number of Classes:  3
##
## Levels:
##  bad good normal
```

We create our confusion matrix

```
pred <- predict(svm_model_after_tune2,x)
table(pred,y)
```

```
##         y
## pred     bad good normal
##   bad    671    8     70
##   good     3  169      6
##   normal  52   35    549
```

```
(671+169+549)/1563*100
```

```
## [1] 88.9
```

We have achieved an 88.9% accuracy level which is much better than the random forest prediction. This is less than the accuracy level we got for white wine because the sample size is much smaller (1563 vs 4877) but it takes much less computational time.

As we can see SVM gives us a much better accuracy level in both cases. SVM has been considered one of the most influential data mining algorithms at the moment because of its strong predictive result.

SVM(Accuracy Level)			
wine type	without tuning	with tuning	sample size
red wine	71%	88.9%	1563
white wine	66.7%	100%	4877

Conclusion

The model approach is based on objective tests and thus it can be integrated into a decision support system when it comes to assessing wine quality and setting prices. This makes the process quicker and more efficient. For instance, the expert could repeat the tasting only if the grade assigned is far from the one predicted by our model. We found that white wine with relatively higher alcohol content had higher ratings but red wine with relatively higher alcohol content had low ratings which is quite surprising.

Sources

- SVM Example http://rischanlab.github.io/SVM.html
- SVM Example http://cs229.stanford.edu/notes/cs229-notes3.pdf
- *http* : *//docs.opencv.org*/2.4/*doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html*