```r
# Set the CRAN mirror to RStudio's cloud server
options(repos = c(CRAN = "https://cloud.r-project.org"))

# Install the package
install.packages("corrplot")
```

```
The downloaded binary packages are in
	/var/folders/kj/zb0ctvqs5zjbr_x5cj8crxjh0000gn/T//RtmpmoMCiW/downloaded_packages
```

```r
library(leaps)
set.seed(1)
red_wine <- read.csv("winequality-red.csv", sep = ";")
white_wine <- read.csv("winequality-white.csv", sep = ";")
summary(red_wine)
```

```
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar  
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900  
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900  
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200  
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539  
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600  
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500  
   chlorides       free.sulfur.dioxide total.sulfur.dioxide    density      
 Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901  
 1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956  
 Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968  
 Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967  
 3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978  
 Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037  
       pH          sulphates         alcohol         quality     
 Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000  
 1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000  
 Median :3.310   Median :0.6200   Median :10.20   Median :6.000  
 Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636  
 3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000  
 Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000  
```

```r
summary(white_wine)
```

```
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar  
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600  
 1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700  
 Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200  
 Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391  
 3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900  
 Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800  
```

```
   chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
 Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
 1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
 Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
 Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
 3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
 Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
       pH          sulphates         alcohol         quality
 Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000
 1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
 Median :3.180   Median :0.4700   Median :10.40   Median :6.000
 Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
 3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
 Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

Exploratory Data Analysis I started the EDA by first examine if there is any skew in the dataset as well as seeing the general distribution of the 2. It is interesting to further look into the dataset since there are so many variables and some of them will significantly impact the future analysis if we do not take them into factors.

```
set.seed(1)
library(ggplot2)
library(dplyr)
```

```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
install.packages("corrplot")
```

```

The downloaded binary packages are in
    /var/folders/kj/zb0ctvqs5zjbr_x5cj8crxjh0000gn/T//RtmpmoMCiW/downloaded_packages
```

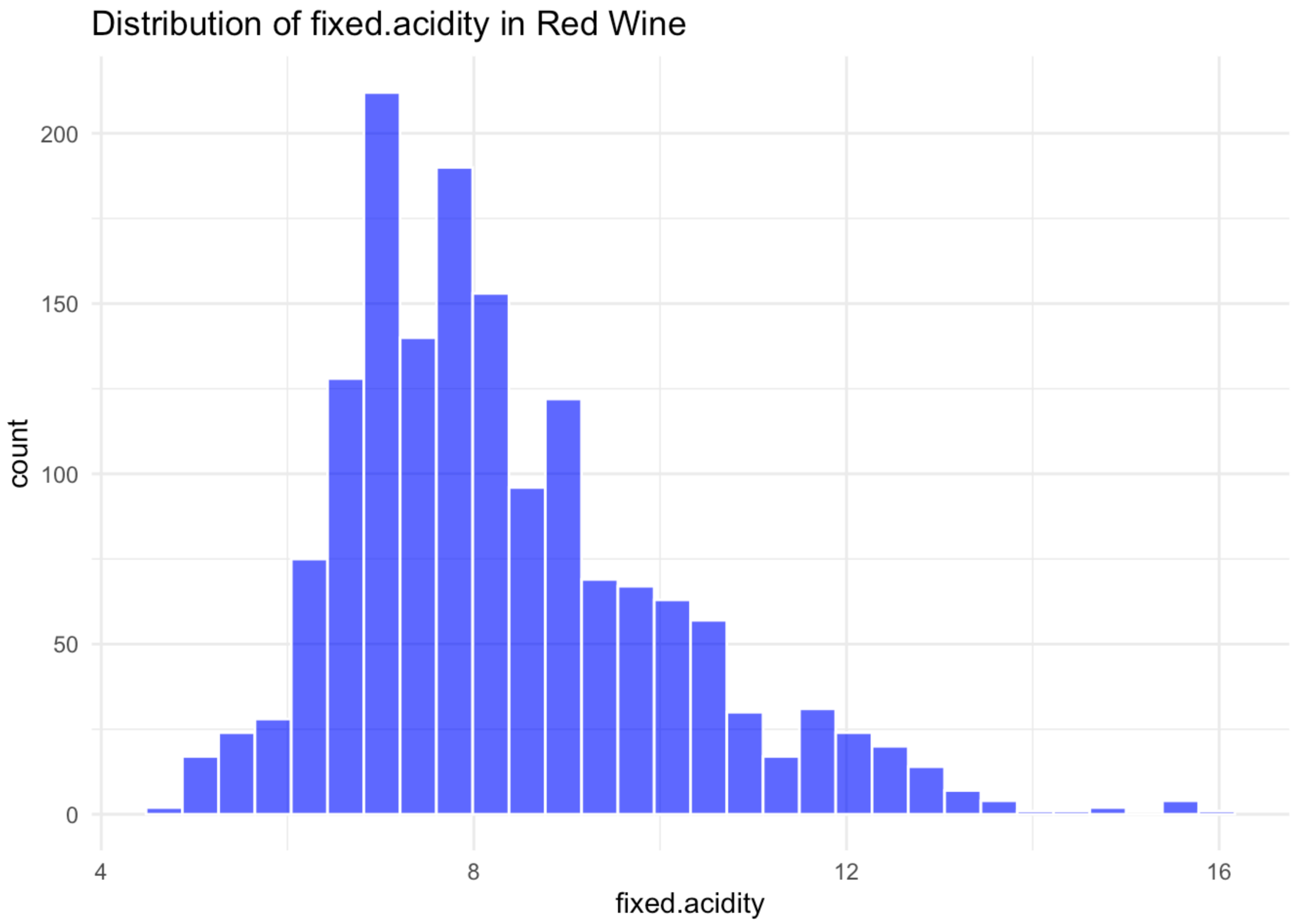
```
library(corrplot)
```

```
corrplot 0.92 loaded
```

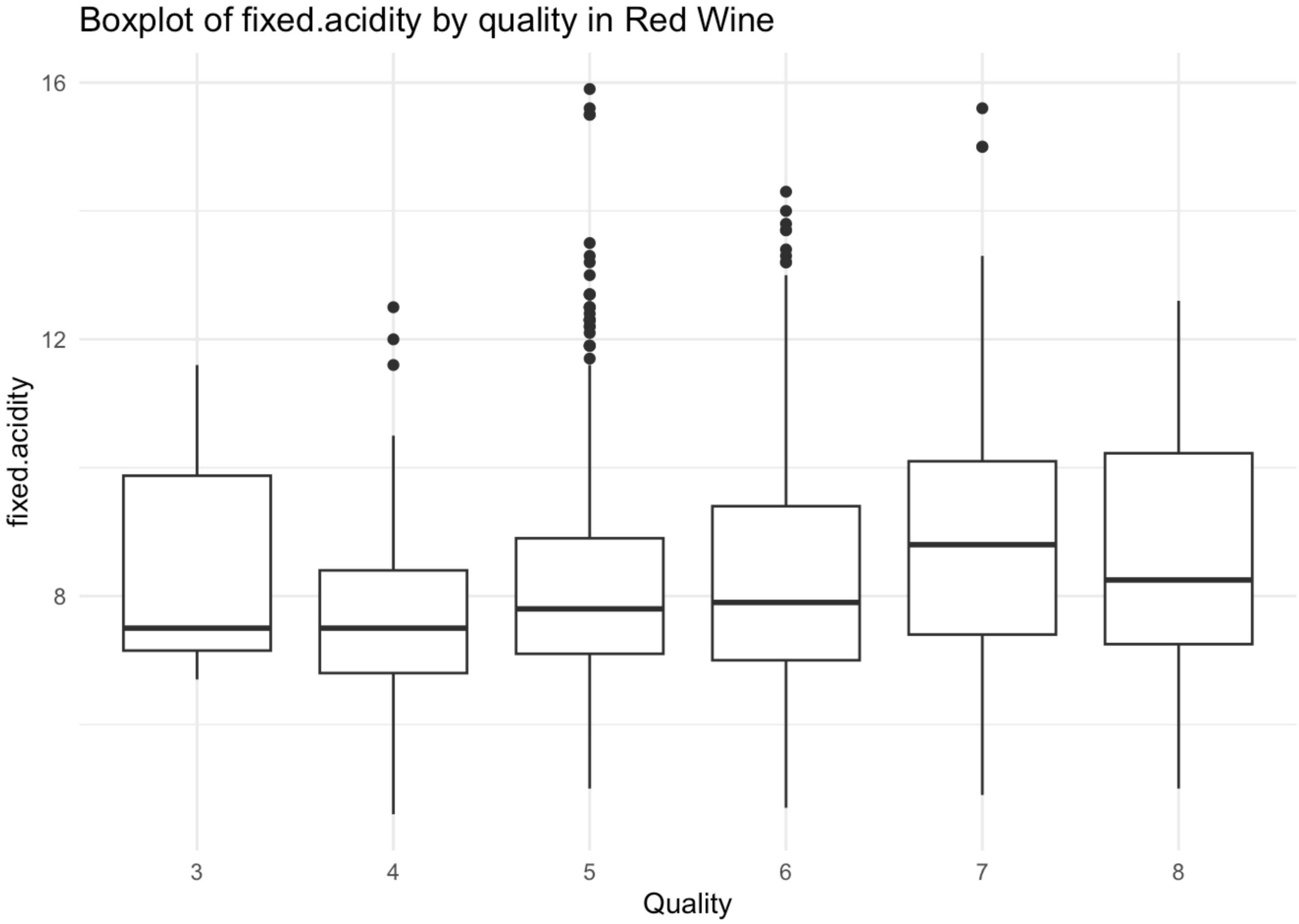
```
visualize_distribution <- function(df, dataset_name) {
  input_vars <- names(df)[1:11]
  hist_plots <- lapply(input_vars, function(var) {
    ggplot(df, aes_string(x = var)) +
      geom_histogram(bins = 30, fill = "blue", color = "white", alpha = 0.7) +
      labs(title = paste("Distribution of", var, "in", dataset_name), x = var) +
      theme_minimal()
  })
  box_plots <- lapply(input_vars, function(var) {
    ggplot(df, aes_string(x = "factor(quality)", y = var)) +
      geom_boxplot() +
      labs(title = paste("Boxplot of", var, "by quality in", dataset_name), x = "Quality"
      theme_minimal()
  })
  return(list(histograms = hist_plots, boxplots = box_plots))
}
red_wine_visuals <- visualize_distribution(red_wine, "Red Wine")
```

```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
ℹ Please use tidy evaluation idioms with `aes()`.
ℹ See also `vignette("ggplot2-in-packages")` for more information.
```

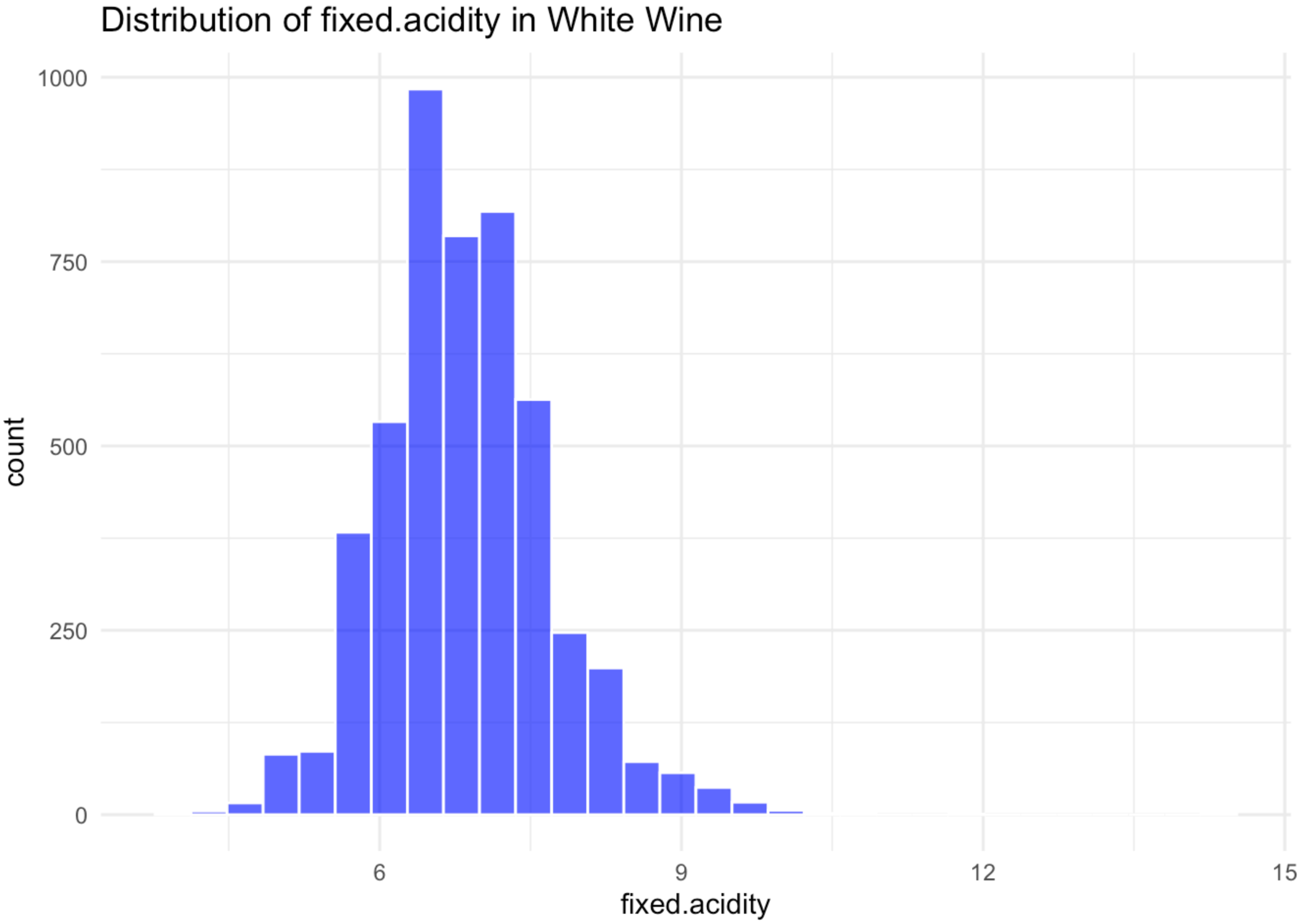
```
white_wine_visuals <- visualize_distribution(white_wine, "White Wine")
print(red_wine_visuals$histograms[[1]])
```



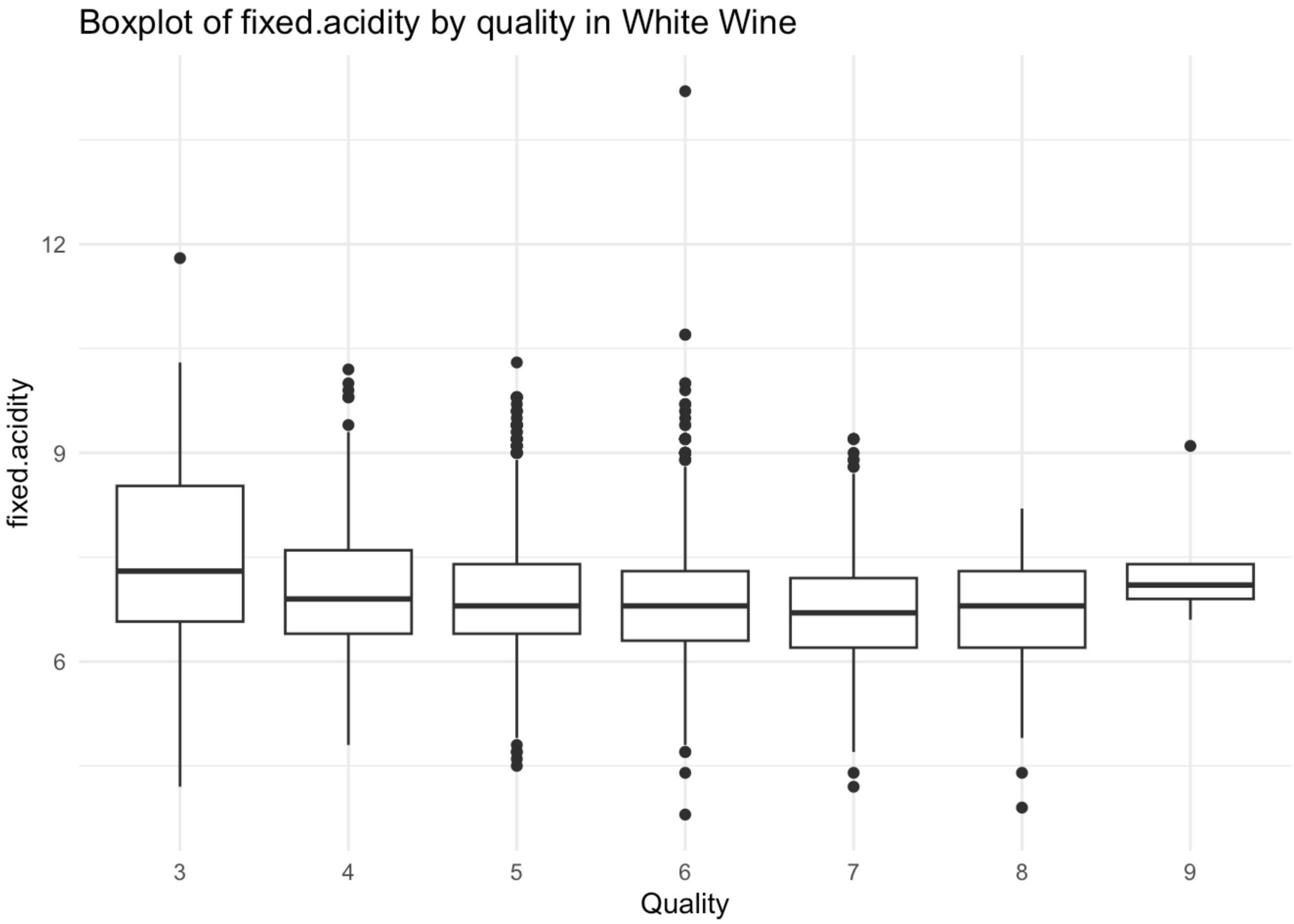
```
print(red_wine_visuals$boxplots[[1]])
```



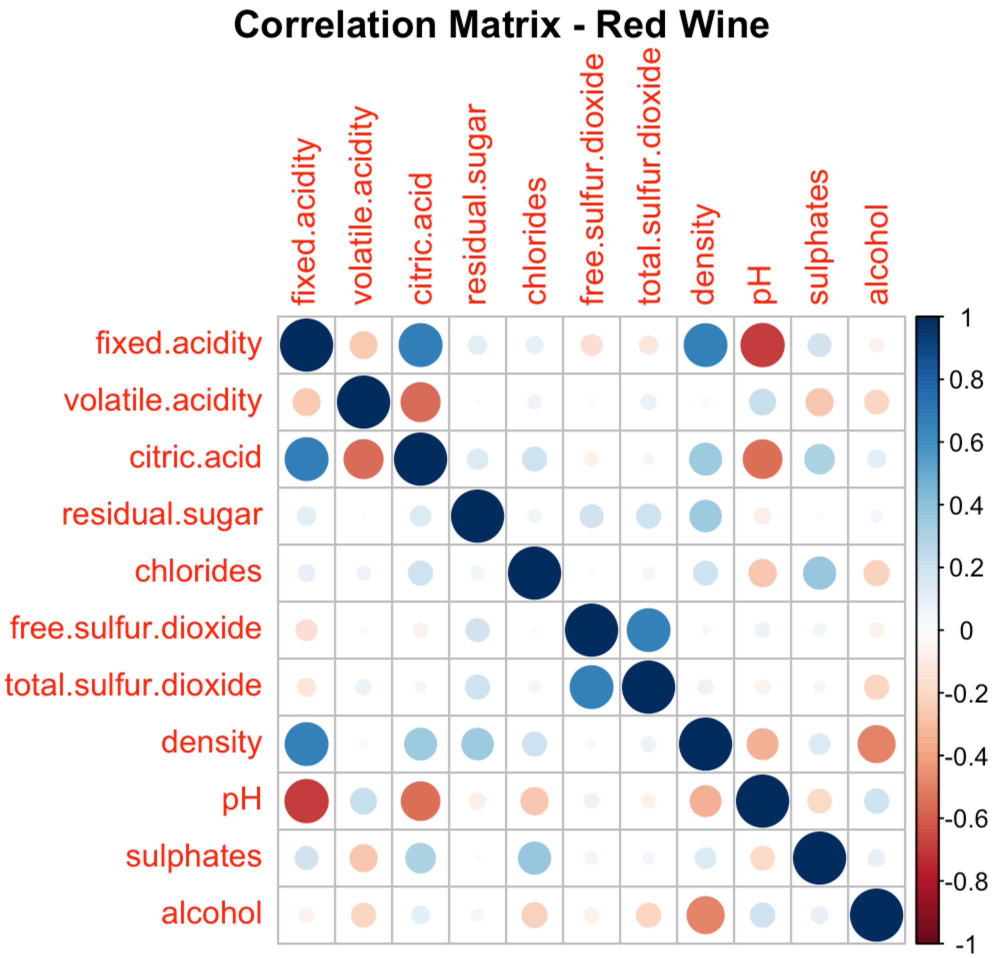
```
print(white_wine_visuals$histograms[[1]])
```



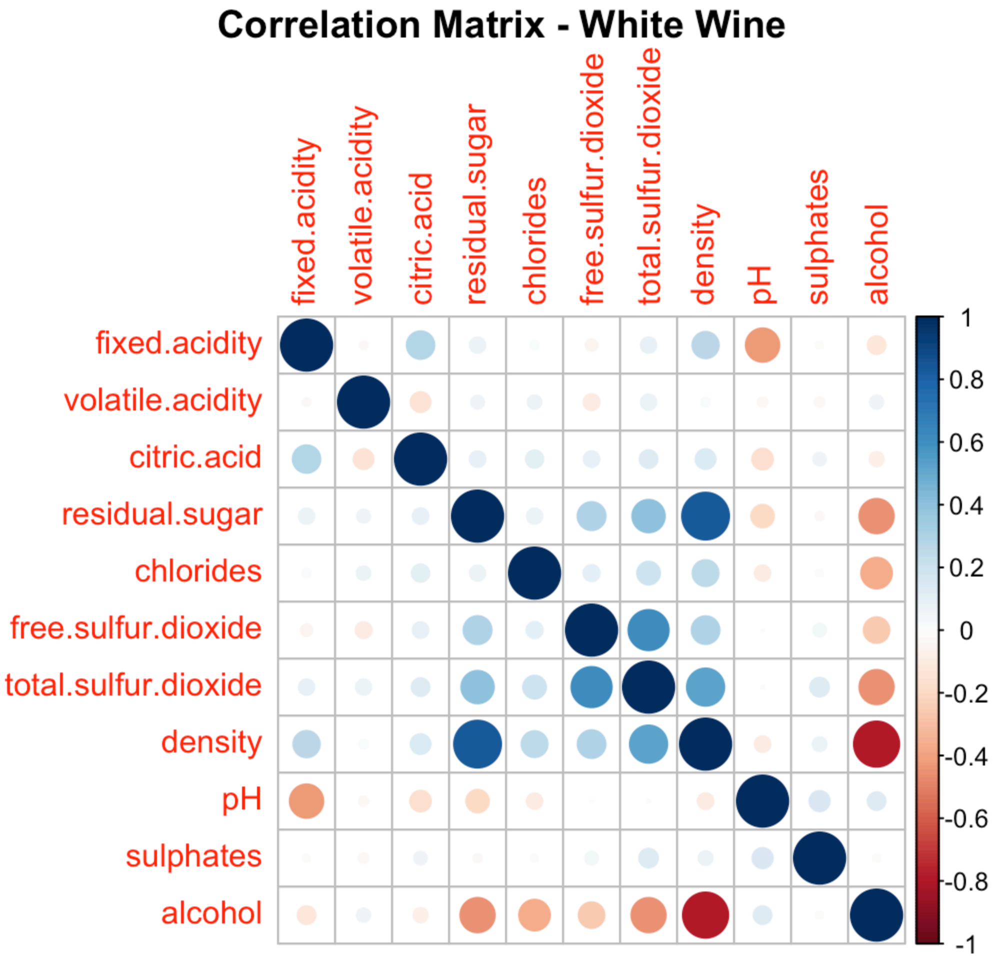
```r
print(white_wine_visuals$boxplots[[1]])
```



```
corr_red <- cor(red_wine[, 1:11])
corr_white <- cor(white_wine[, 1:11])
corrplot(corr_red, method = "circle", title = "Correlation Matrix - Red Wine", mar = c(0,
```



```
corrplot(corr_white, method = "circle", title = "Correlation Matrix - White Wine", mar =
```



```
table(red_wine$quality)
```

```
  3   4   5   6   7   8 
 10  53 681 638 199  18 
```

```
table(white_wine$quality)
```

```
   3    4    5    6    7    8    9 
  20  163 1457 2198  880  175    5 
```

This part is mainly about examine the correlation among the variables and how they will impact the outcome of our predictions. We also do some initial analysis by plotting the graph by counting the frequency for the variables, just to have some grasps about how the data perform in general or if there is any outliers for both dataset.

```
library(ggplot2)
library(dplyr)
library(caret)
```

```
Loading required package: lattice
```

```
install.packages("ggcorrplot")
```

```

The downloaded binary packages are in
	/var/folders/kj/zb0ctvqs5zjbr_x5cj8crxjh0000gn/T//RtmpmoMCiW/downloaded_packages
```

```
library(ggcorrplot)
library(GGally)
```

```
Registered S3 method overwritten by 'GGally':
  method from   
  +.gg   ggplot2
```

```
library(cluster)
install.packages("factoextra")
```

```

The downloaded binary packages are in
	/var/folders/kj/zb0ctvqs5zjbr_x5cj8crxjh0000gn/T//RtmpmoMCiW/downloaded_packages
```

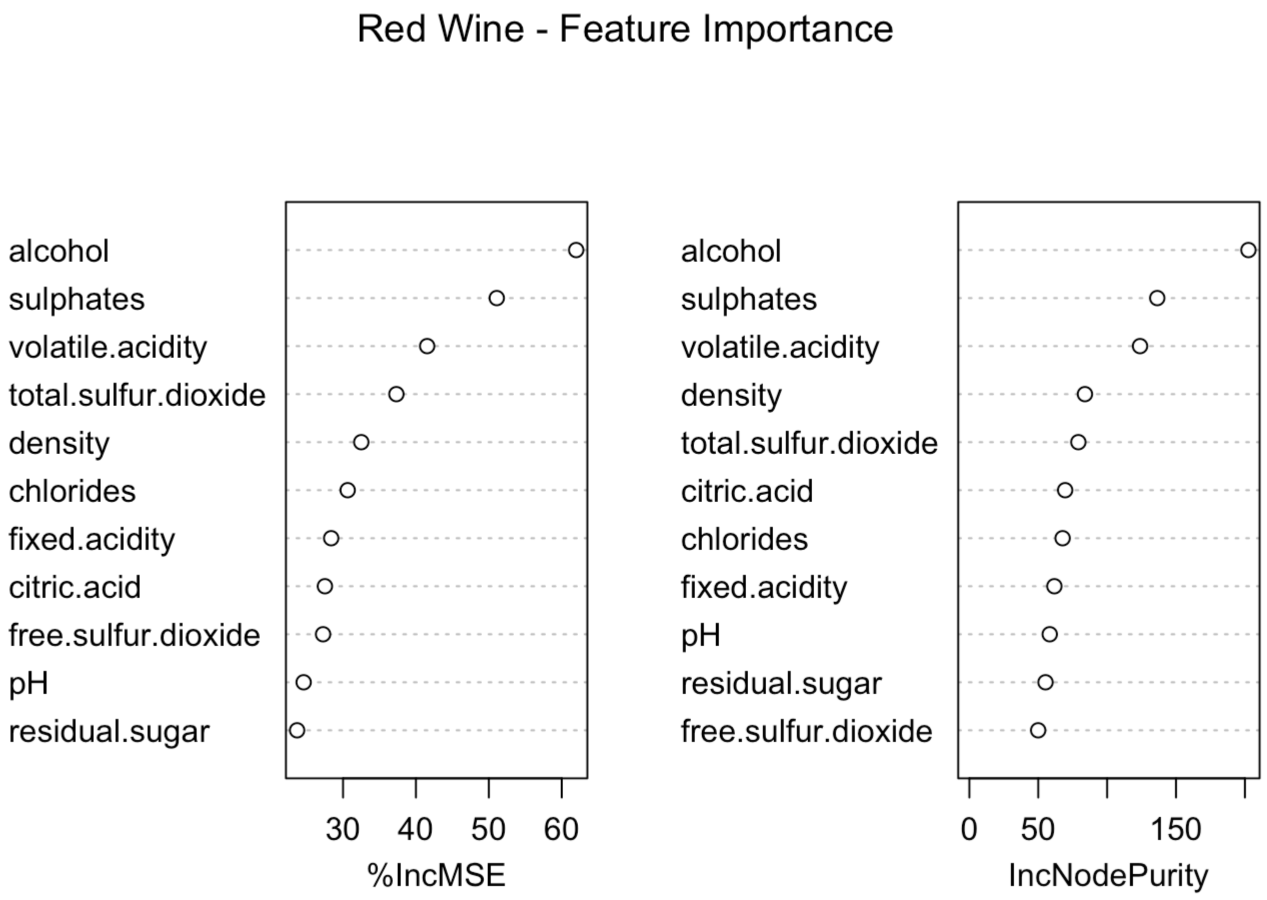
```
library(factoextra)
```

```
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

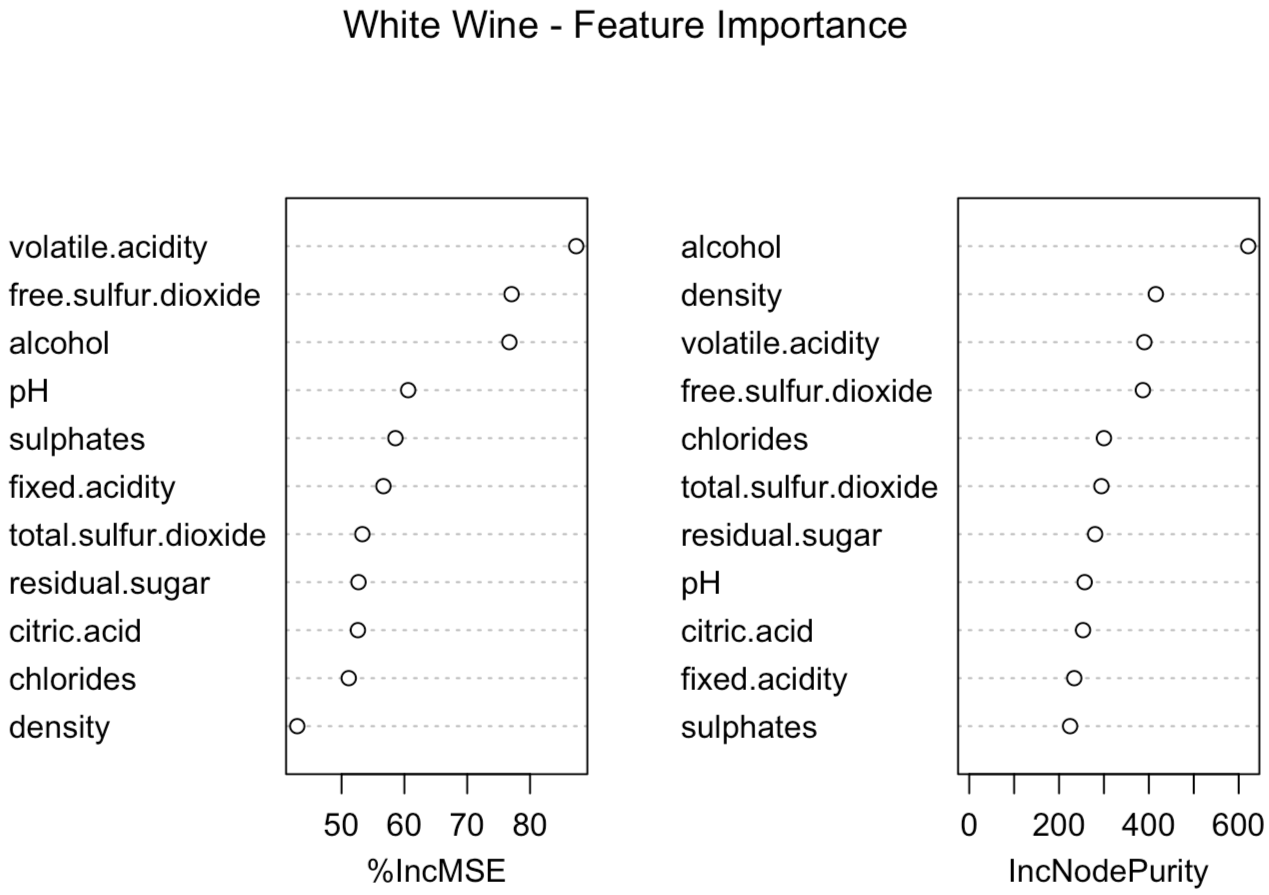
```
library(cowplot)
library(randomForest)
```

```
randomForest 4.7-1.1

Type rfNews() to see new features/changes/bug fixes.


Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':

    combine

The following object is masked from 'package:ggplot2':

    margin
```

```
detect_outliers <- function(df, var) {
  Q1 <- quantile(df[[var]], 0.25)
  Q3 <- quantile(df[[var]], 0.75)
  IQR <- Q3 - Q1
  outliers <- df |>
    filter(df[[var]] < (Q1 - 1.5 * IQR) | df[[var]] > (Q3 + 1.5 * IQR))
  return(outliers)
}
red_outliers <- lapply(names(red_wine)[1:11], detect_outliers, df = red_wine)
white_outliers <- lapply(names(white_wine)[1:11], detect_outliers, df = white_wine)
set.seed(123)
red_rf <- randomForest(quality ~ ., data = red_wine, importance = TRUE)
white_rf <- randomForest(quality ~ ., data = white_wine, importance = TRUE)
red_imp <- varImpPlot(red_rf, main = "Red Wine - Feature Importance")
```



```
white_imp <- varImpPlot(white_rf, main = "White Wine - Feature Importance")
```



```
pca_visualization <- function(df, dataset_name) {
  pca <- prcomp(df[, 1:11], scale. = TRUE)
  pca_df <- data.frame(pca$x)
  pca_df$quality <- df$quality

  ggplot(pca_df, aes(PC1, PC2, color = factor(quality))) +
    geom_point(alpha = 0.7) +
    labs(title = paste("PCA Clustering for", dataset_name), color = "Quality") +
    theme_minimal()
}
red_pca_plot <- pca_visualization(red_wine, "Red Wine")
white_pca_plot <- pca_visualization(white_wine, "White Wine")
```

In the graphs above, I mainly want to explore how different variables can affect the performance if we do model analysis as it is a very significant factor to consider. And the graph clearly demonstrates the the importance of each variables.

```
set.seed(1)
# Perform best subset selection for white wines
regfit.full_white <- regsubsets(quality ~ ., data = white_wine, nvmax = 12)
reg.summary_white <- summary(regfit.full_white)
(reg.summary_white)
```

```
Subset selection object
Call: regsubsets.formula(quality ~ ., data = white_wine, nvmax = 12)
11 Variables  (and intercept)
                     Forced in Forced out
fixed.acidity            FALSE      FALSE
volatile.acidity         FALSE      FALSE
citric.acid              FALSE      FALSE
residual.sugar           FALSE      FALSE
chlorides                FALSE      FALSE
free.sulfur.dioxide      FALSE      FALSE
total.sulfur.dioxide     FALSE      FALSE
density                  FALSE      FALSE
pH                       FALSE      FALSE
sulphates                FALSE      FALSE
alcohol                  FALSE      FALSE
1 subsets of each size up to 11
Selection Algorithm: exhaustive
          fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1  ( 1 )  " "           " "              " "         " "            " "      
2  ( 1 )  " "           "*"              " "         " "            " "      
3  ( 1 )  " "           "*"              " "         "*"            " "      
4  ( 1 )  " "           "*"              " "         "*"            " "      
5  ( 1 )  " "           "*"              " "         "*"            " "      
6  ( 1 )  " "           "*"              " "         "*"            " "      
7  ( 1 )  " "           "*"              " "         "*"            " "      
8  ( 1 )  "*"           "*"              " "         "*"            " "      
9  ( 1 )  "*"           "*"              " "         "*"            " "      
10  ( 1 ) "*"           "*"              " "         "*"            "*"      
11  ( 1 ) "*"           "*"              "*"         "*"            "*"      
          free.sulfur.dioxide total.sulfur.dioxide density pH  sulphates
1  ( 1 )  " "                 " "                  " "     " " " "      
2  ( 1 )  " "                 " "                  " "     " " " "      
3  ( 1 )  " "                 " "                  " "     " " " "      
4  ( 1 )  "*"                 " "                  " "     " " " "      
5  ( 1 )  " "                 " "                  "*"     "*" " "      
6  ( 1 )  " "                 " "                  "*"     "*" "*"      
7  ( 1 )  "*"                 " "                  "*"     "*" "*"      
8  ( 1 )  "*"                 " "                  "*"     "*" "*"      
9  ( 1 )  "*"                 "*"                  "*"     "*" "*"      
10  ( 1 ) "*"                 "*"                  "*"     "*" "*"      
11  ( 1 ) "*"                 "*"                  "*"     "*" "*"      
          alcohol
1  ( 1 )  "*"    
2  ( 1 )  "*"    
3  ( 1 )  "*"    
4  ( 1 )  "*"    
5  ( 1 )  "*"    
6  ( 1 )  "*"    
7  ( 1 )  "*"    
8  ( 1 )  "*"    
```

```
9  ( 1 )  "*"
10  ( 1 ) "*"
11  ( 1 ) "*"
```

```
(reg.summary_white$adjr2)
```

```
 [1] 0.1895598 0.2399208 0.2580716 0.2633925 0.2703282 0.2757705 0.2790891
 [8] 0.2805767 0.2805130 0.2803931 0.2802536
```

```
# Find best subset based on BIC
best_subset_white <- which.min(reg.summary_white$bic)
# Perform best subset selection for red wines
regfit.full_red <- regsubsets(quality ~ ., data = red_wine, nvmax = 12)
reg.summary_red <- summary(regfit.full_red)
(reg.summary_red)
```

```
Subset selection object
Call: regsubsets.formula(quality ~ ., data = red_wine, nvmax = 12)
11 Variables  (and intercept)
                     Forced in Forced out
fixed.acidity            FALSE      FALSE
volatile.acidity         FALSE      FALSE
citric.acid              FALSE      FALSE
residual.sugar           FALSE      FALSE
chlorides                FALSE      FALSE
free.sulfur.dioxide      FALSE      FALSE
total.sulfur.dioxide     FALSE      FALSE
density                  FALSE      FALSE
pH                       FALSE      FALSE
sulphates                FALSE      FALSE
alcohol                  FALSE      FALSE
1 subsets of each size up to 11
Selection Algorithm: exhaustive
          fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1  ( 1 )  " "           " "              " "         " "            " "      
2  ( 1 )  " "           "*"              " "         " "            " "      
3  ( 1 )  " "           "*"              " "         " "            " "      
4  ( 1 )  " "           "*"              " "         " "            " "      
5  ( 1 )  " "           "*"              " "         " "            "*"      
6  ( 1 )  " "           "*"              " "         " "            "*"      
7  ( 1 )  " "           "*"              " "         " "            "*"      
8  ( 1 )  " "           "*"              "*"         " "            "*"      
9  ( 1 )  " "           "*"              "*"         "*"            "*"      
10  ( 1 ) "*"           "*"              "*"         "*"            "*"      
11  ( 1 ) "*"           "*"              "*"         "*"            "*"      
          free.sulfur.dioxide total.sulfur.dioxide density pH  sulphates
1  ( 1 )  " "                 " "                  " "     " " " "      
2  ( 1 )  " "                 " "                  " "     " " " "      
3  ( 1 )  " "                 " "                  " "     " " "*"      
```

```
4  ( 1 )  " "                 "*"                  " "     " " "*"
5  ( 1 )  " "                 "*"                  " "     " " "*"
6  ( 1 )  " "                 "*"                  " "     "*" "*"
7  ( 1 )  "*"                 "*"                  " "     "*" "*"
8  ( 1 )  "*"                 "*"                  " "     "*" "*"
9  ( 1 )  "*"                 "*"                  " "     "*" "*"
10  ( 1 ) "*"                 "*"                  " "     "*" "*"
11  ( 1 ) "*"                 "*"                  "*"     "*" "*"
          alcohol
1  ( 1 )  "*"
2  ( 1 )  "*"
3  ( 1 )  "*"
4  ( 1 )  "*"
5  ( 1 )  "*"
6  ( 1 )  "*"
7  ( 1 )  "*"
8  ( 1 )  "*"
9  ( 1 )  "*"
10  ( 1 ) "*"
11  ( 1 ) "*"
```

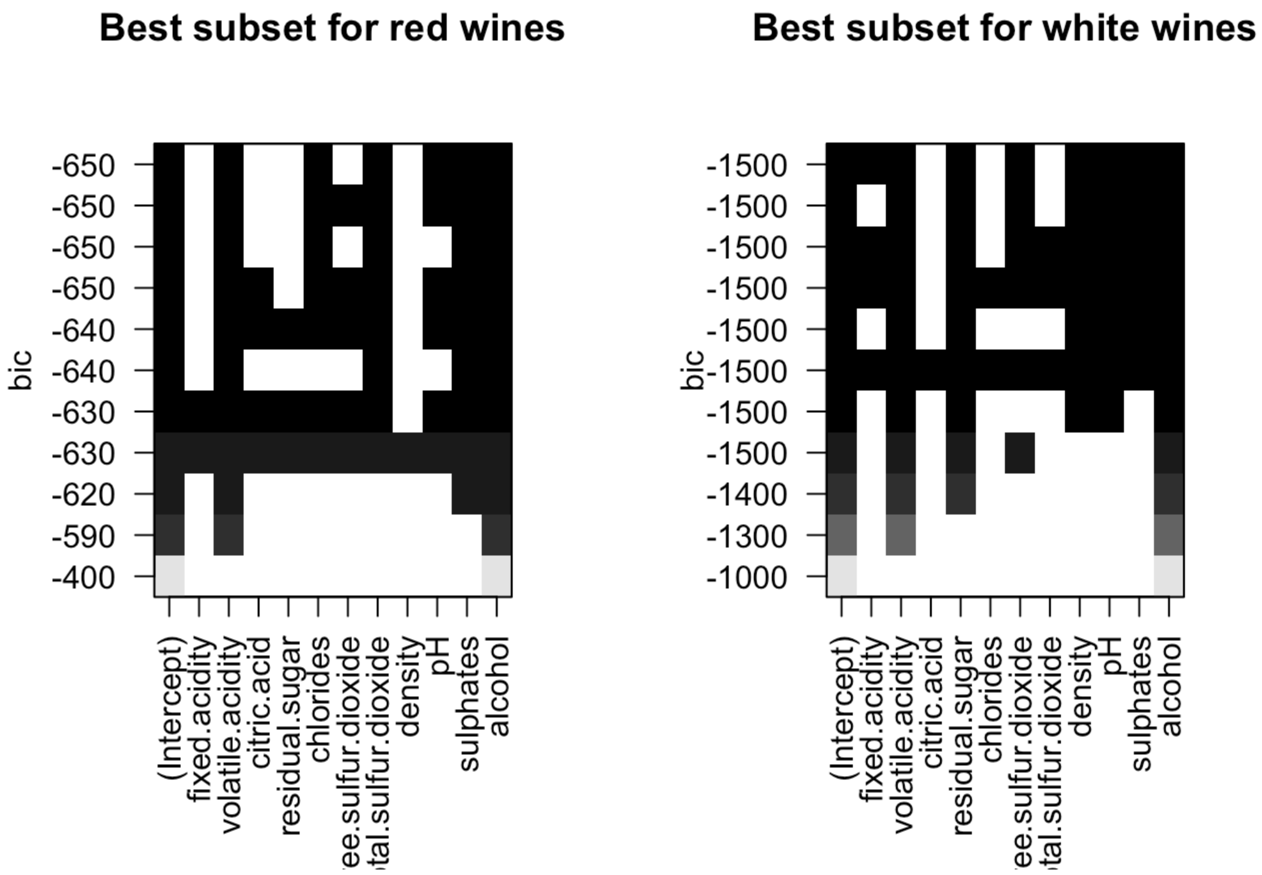
```
(reg.summary_red$adjr2)
```

```
 [1] 0.2262502 0.3161465 0.3346482 0.3421357 0.3494588 0.3547509 0.3566527
 [8] 0.3567060 0.3565489 0.3562479 0.3561195
```

```
# Find best subset based on BIC
best_subset_red <- which.min(reg.summary_red$bic)
```

Here we are just running the best subset selection so that we can run the future model using this as a reference. It is also important to note that we are looking for the one that has the smallest BIC is the set of predictors that we are going to use.

```
set.seed(1)
par(mfrow = c(1,2))
plot(regfit.full_red, scale = "bic", main = 'Best subset for red wines')
plot(regfit.full_white, scale = "bic", main = 'Best subset for white wines')
```



In this case, we are just grpahing the best subset based on the result above.

```r
library(caret)
library(randomForest)
set.seed(1)
adjusted_r2 <- function(r2, n, p) {
  return(1 - ((1 - r2) * (n - 1) / (n - p - 1)))
}
aic_calculator <- function(n, mse, num_params) {
  return(n * log(mse) + 2 * num_params)
}
ctrl <- trainControl(method = "cv", number = 10)
model_white_rf <- train(
  quality ~ volatile.acidity + citric.acid + chlorides + free.sulfur.dioxide +
    total.sulfur.dioxide + pH + sulphates + alcohol,
  data = white_wine,
  method = "rf",
  trControl = ctrl
)
white_rf_results <- model_white_rf$results
best_white_rf <- model_white_rf$bestTune
final_model_white_rf <- model_white_rf$finalModel
white_r2 <- max(white_rf_results$Rsquared)
n_white <- nrow(white_wine)
p_white <- ncol(white_wine) - 1
```

```
white_adj_r2 <- adjusted_r2(white_r2, n_white, p_white)
white_rf_predictions <- predict(final_model_white_rf, newdata = white_wine)
white_mse <- mean((white_wine$quality - white_rf_predictions)^2)
white_aic <- aic_calculator(n_white, white_mse, p_white)
model_red_rf <- train(
  quality ~ volatile.acidity + citric.acid + chlorides + free.sulfur.dioxide +
    total.sulfur.dioxide + pH + sulphates + alcohol,
  data = red_wine,
  method = "rf",
  trControl = ctrl
)
red_rf_results <- model_red_rf$results
best_red_rf <- model_red_rf$bestTune
final_model_red_rf <- model_red_rf$finalModel
red_r2 <- max(red_rf_results$Rsquared)
n_red <- nrow(red_wine)
p_red <- ncol(red_wine) - 1
red_adj_r2 <- adjusted_r2(red_r2, n_red, p_red)
red_rf_predictions <- predict(final_model_red_rf, newdata = red_wine)
red_mse <- mean((red_wine$quality - red_rf_predictions)^2)
red_aic <- aic_calculator(n_red, red_mse, p_red)
cat("Best Random Forest model for white wine quality:\n")
```

```
Best Random Forest model for white wine quality:
```

```
print(best_white_rf)
```

```
  mtry
2    5
```

```
cat("\nWhite wine performance metrics:\n")
```

```

White wine performance metrics:
```

```
cat("Adjusted R^2:", white_adj_r2, "\n")
```

```
Adjusted R^2: 0.5408396
```

```
cat("MSE:", white_mse, "\n")
```

```
MSE: 0.0671829
```

```
cat("AIC:", white_aic, "\n")
```

```
AIC: -13204.25
```

```r
cat("\nBest Random Forest model for red wine quality:\n")
```

```

Best Random Forest model for red wine quality:
```

```r
print(best_red_rf)
```

```
  mtry
1    2
```

```r
cat("\nRed wine performance metrics:\n")
```

```

Red wine performance metrics:
```

```r
cat("Adjusted R^2:", red_adj_r2, "\n")
```

```
Adjusted R^2: 0.5156118
```

```r
cat("MSE:", red_mse, "\n")
```

```
MSE: 0.0698625
```

```r
cat("AIC:", red_aic, "\n")
```

```
AIC: -4233.301
```

Lastly, I performed the random forest model to train the model so that we can get the best adjusted r square, mse, aic. We can tell that from the result, it is probably not the best model to run for this specific dataset since a lot of the performance metrics are extremely poor.