Project 3 - DS8002

Rafik Matta

Exploratorary Data Analysis

```r
library(readr)
library("MASS")
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(plotly)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'plotly'

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

## The following object is masked from 'package:MASS':
##
##     select

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```r
winequality_red <- read_delim("C:/Users/rafik/Google Drive/Education/ds8002/projects/project3/winequali
    ";", escape_double = FALSE, trim_ws = TRUE)
```

```
## Parsed with column specification:
## cols(
##   `fixed acidity` = col_double(),
##   `volatile acidity` = col_double(),
##   `citric acid` = col_double(),
##   `residual sugar` = col_double(),
##   chlorides = col_double(),
##   `free sulfur dioxide` = col_double(),
##   `total sulfur dioxide` = col_integer(),
##   density = col_double(),
##   pH = col_double(),
##   sulphates = col_double(),
##   alcohol = col_double(),
##   quality = col_integer()
## )

## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)
## Warning: 2 parsing failures.
## row # A tibble: 2 x 5 col     row                  col               expected actual expected   <int
```

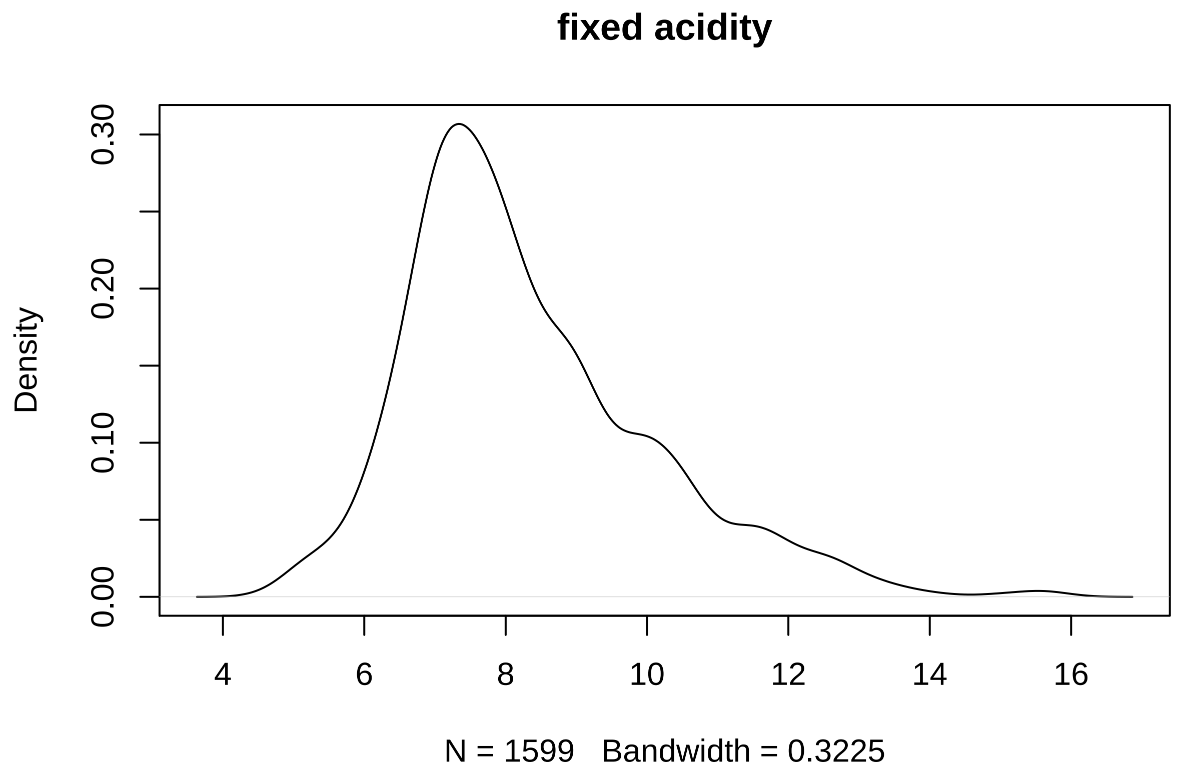
```r
winequality_white <- read_delim("C:/Users/rafik/Google Drive/Education/ds8002/projects/project3/winequal
    ";", escape_double = FALSE, trim_ws = TRUE)
```

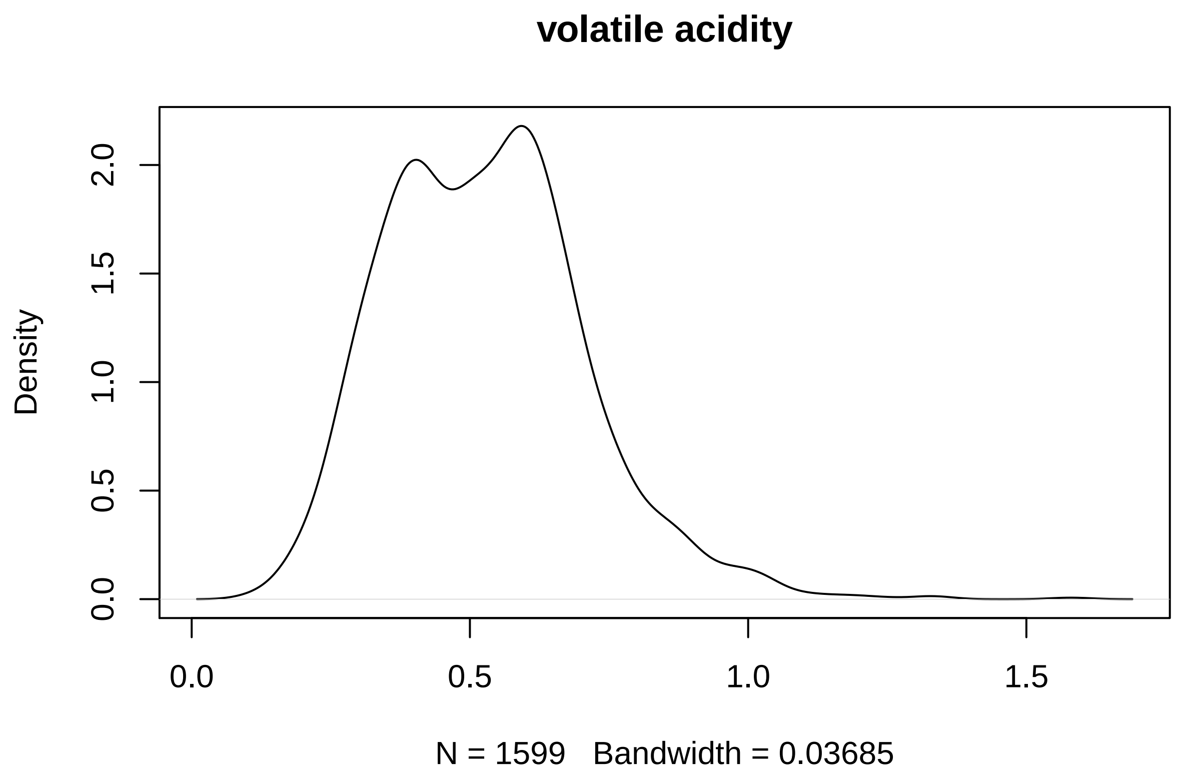
```
## Parsed with column specification:
## cols(
##   `fixed acidity` = col_double(),
##   `volatile acidity` = col_double(),
##   `citric acid` = col_double(),
##   `residual sugar` = col_double(),
##   chlorides = col_double(),
##   `free sulfur dioxide` = col_double(),
##   `total sulfur dioxide` = col_double(),
##   density = col_double(),
##   pH = col_double(),
##   sulphates = col_double(),
##   alcohol = col_double(),
##   quality = col_integer()
## )
```

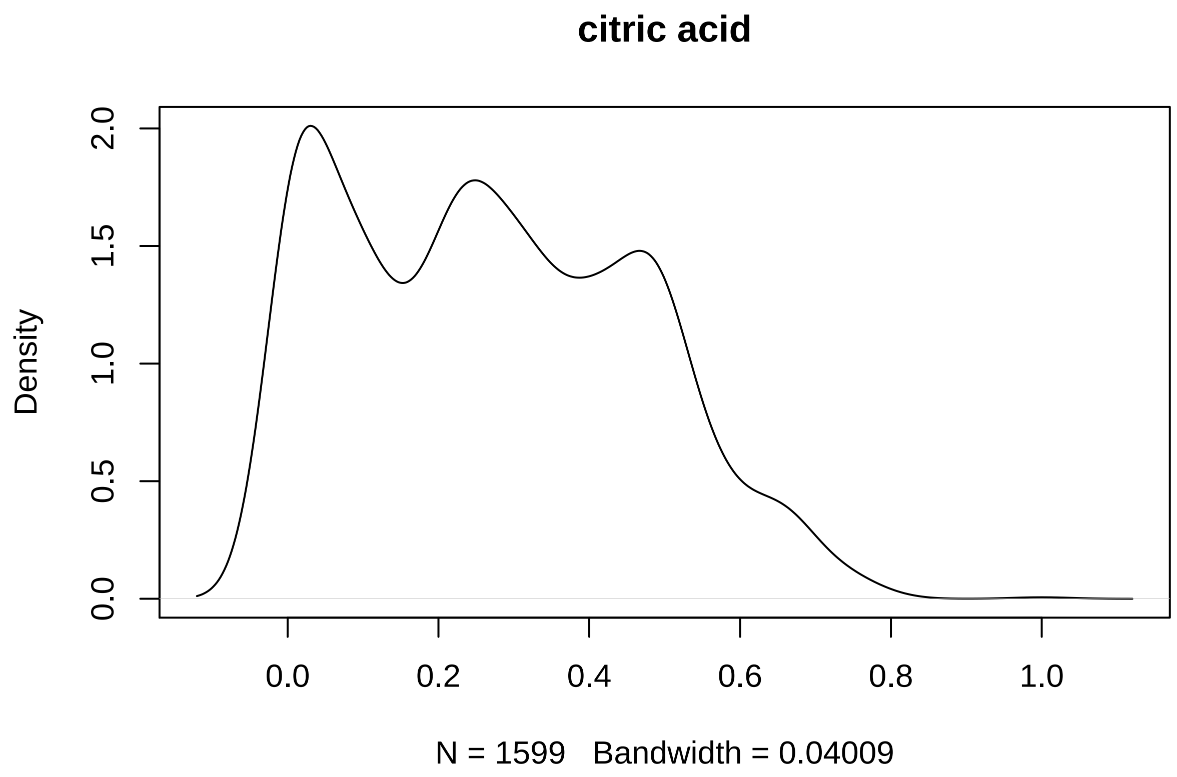
EDA Question 1- Variance of each column

Plots for Red DataSet

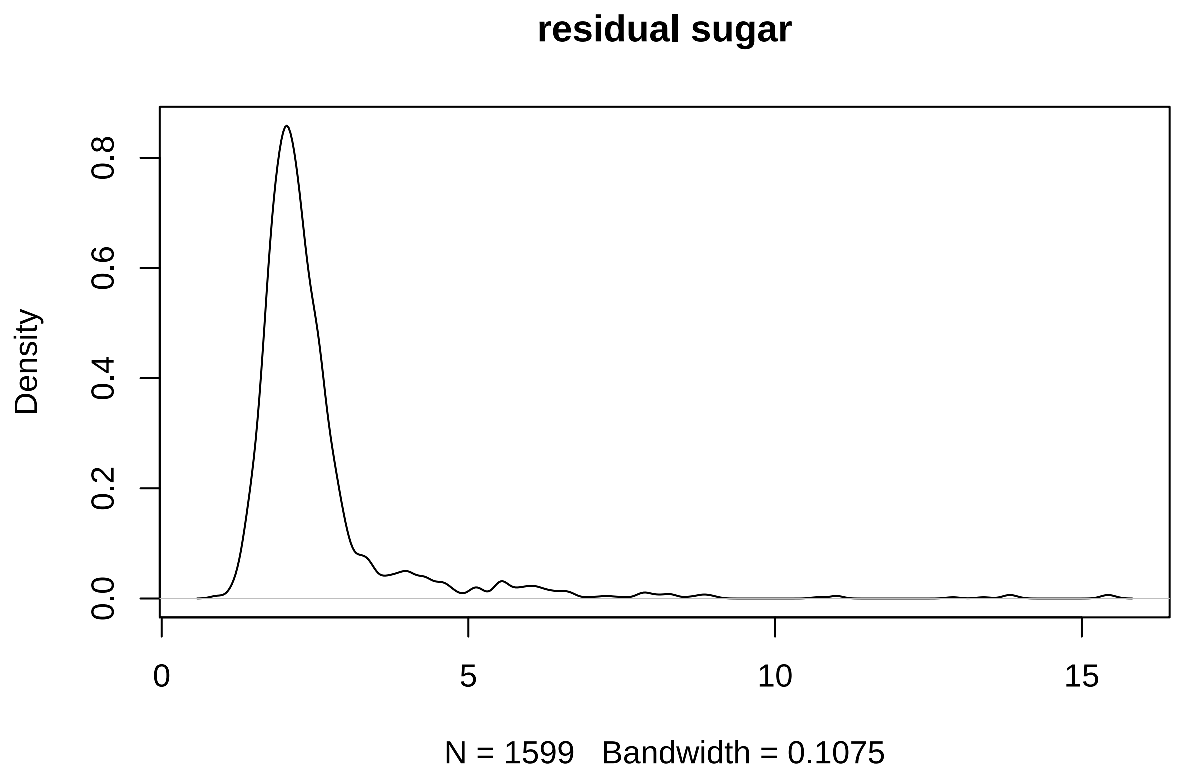
```
for( i in colnames(winequality_red))
{
  d <- density(winequality_red[[i]],na.rm = TRUE)
  plot(d,main=i)

}
```

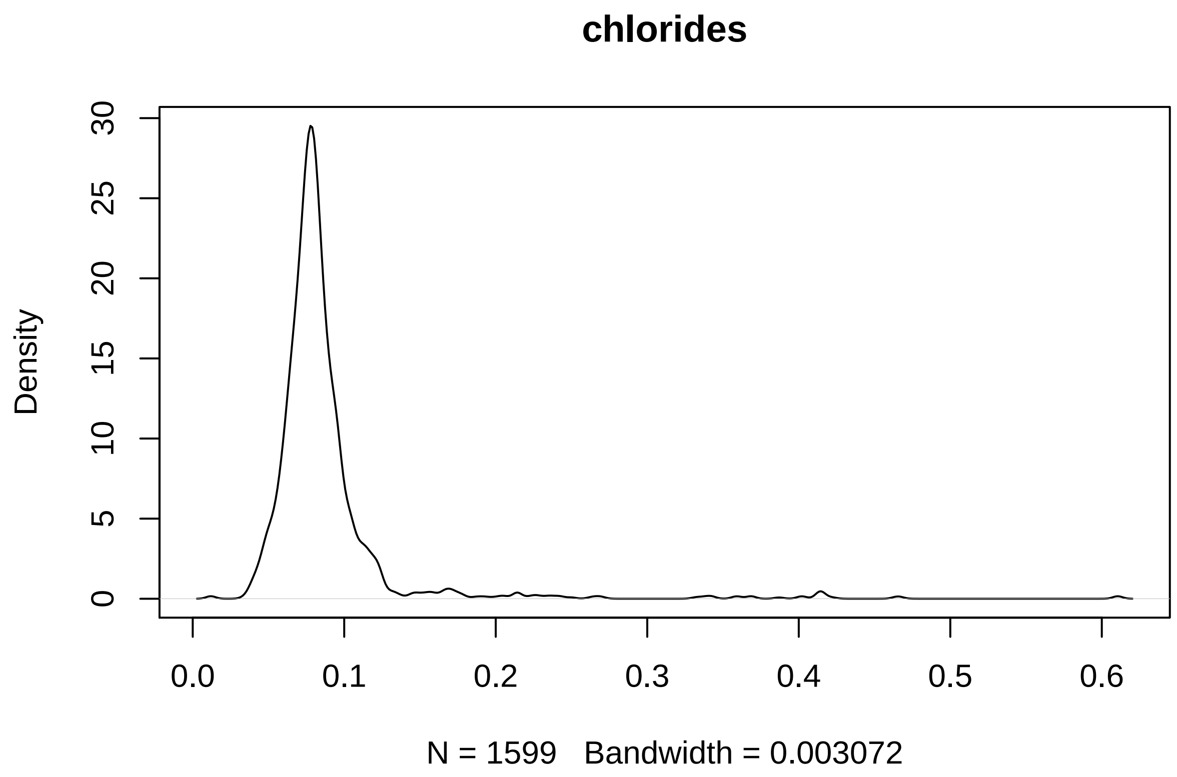


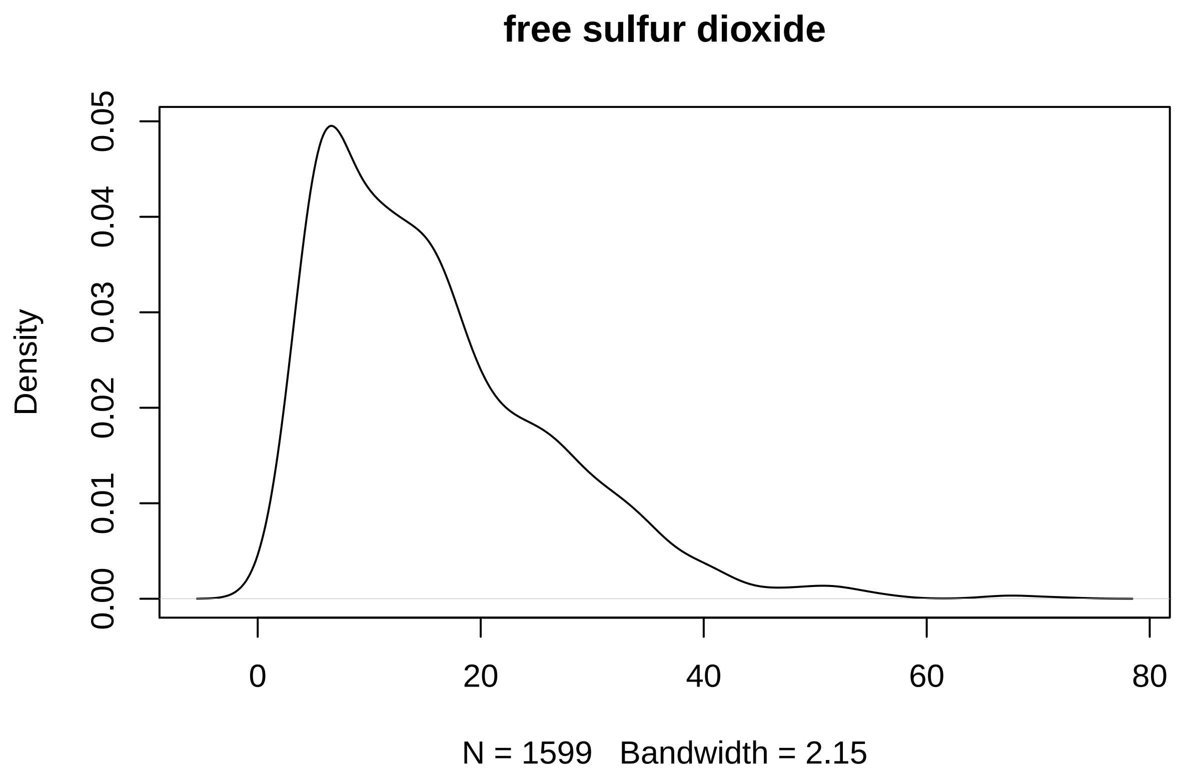

volatile acidity
Density
0.0
0.5
1.0
1.5
2.0
0.0
0.5
1.0
1.5
N = 1599   Bandwidth = 0.03685

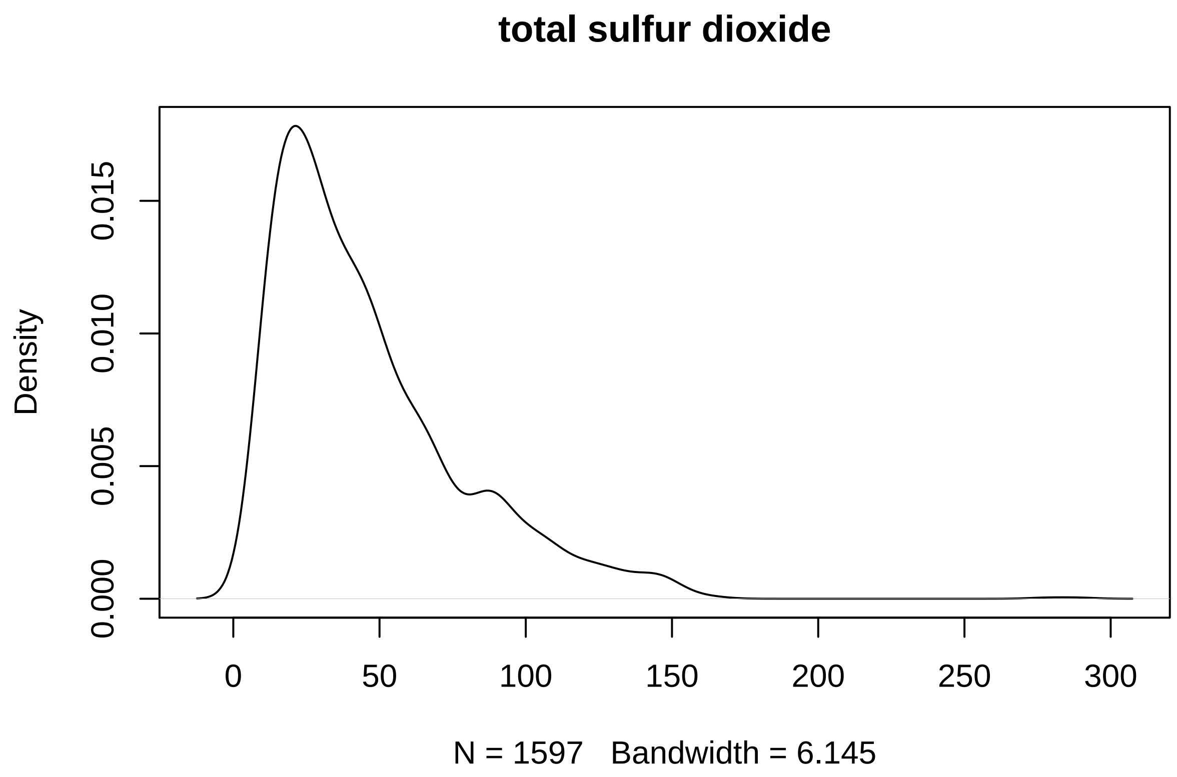


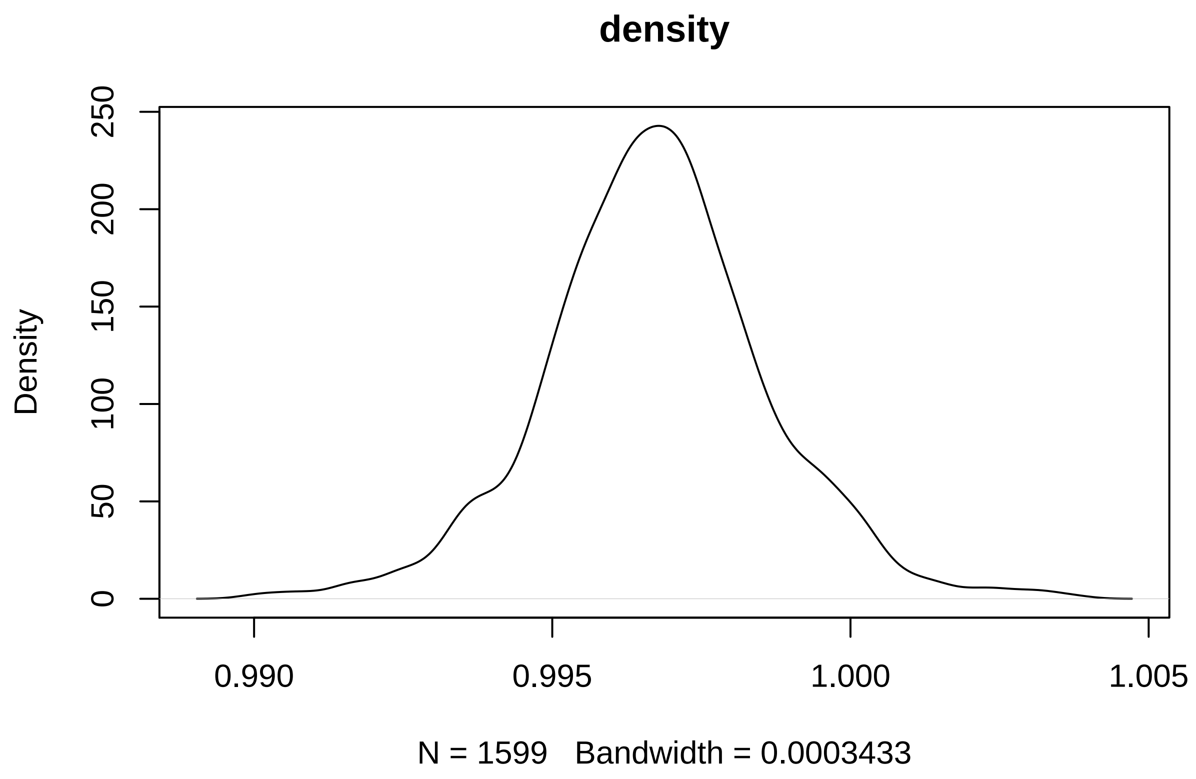
citric acid
Density
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0
N = 1599   Bandwidth = 0.04009

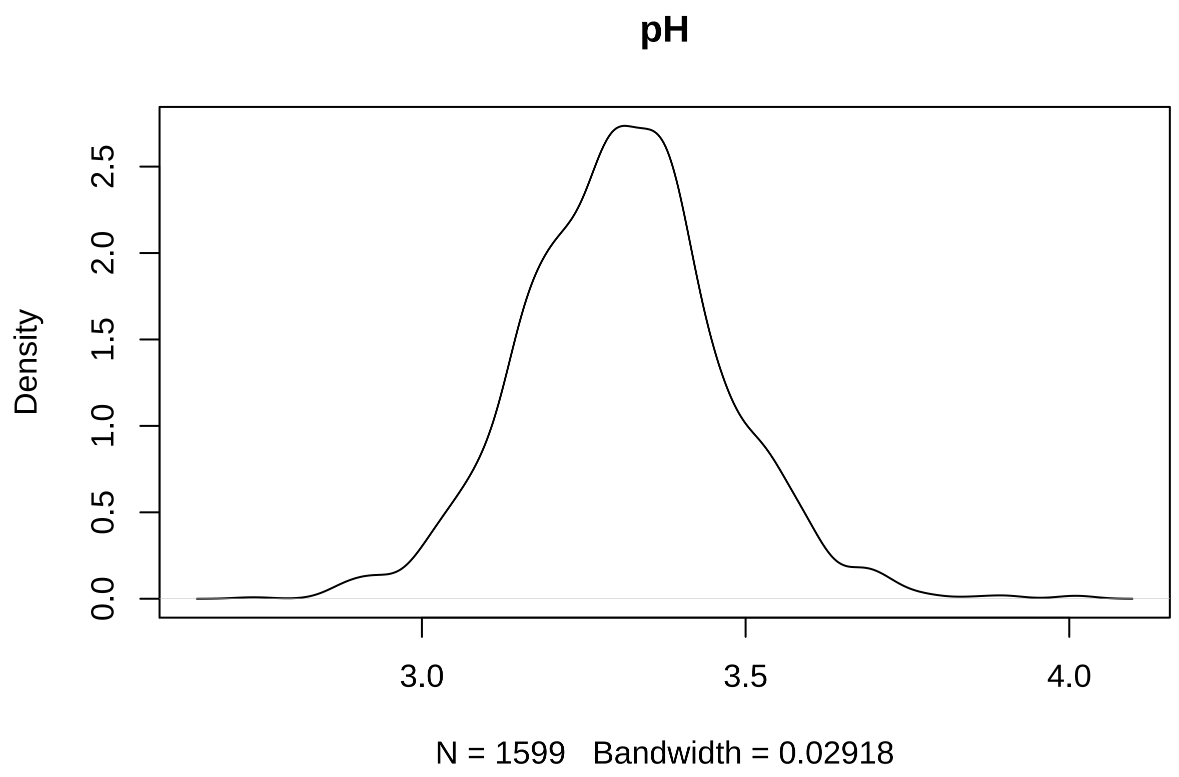

residual sugar
Density
0.0
0.2
0.4
0.6
0.8
0
5
10
15
N = 1599 Bandwidth = 0.1075

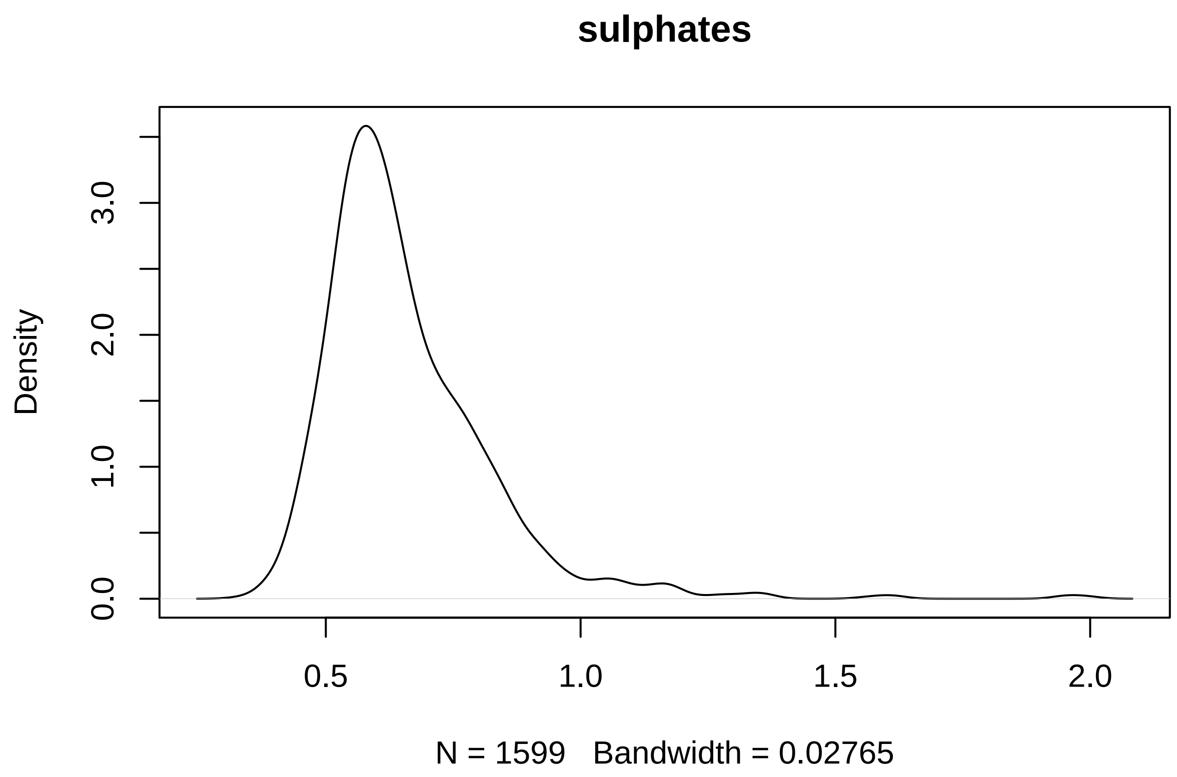

chlorides
Density
30
25
20
15
10
5
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
N = 1599   Bandwidth = 0.003072

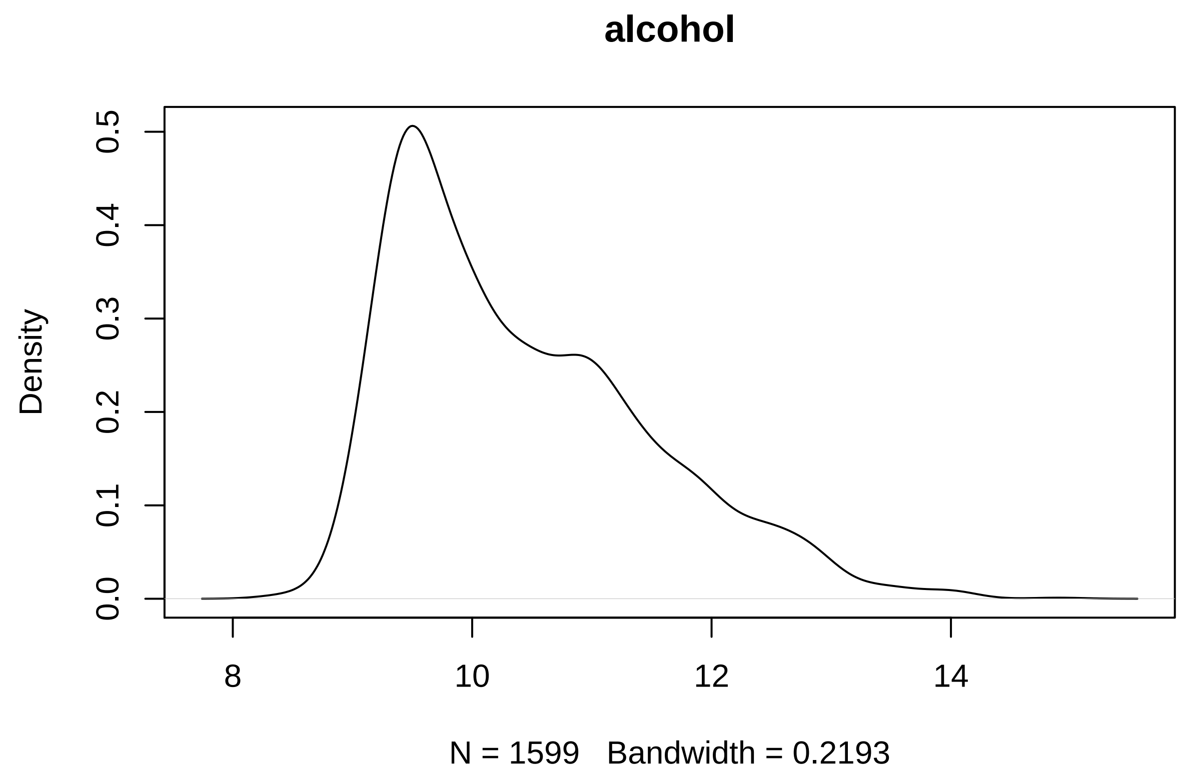

free sulfur dioxide
Density
0.00
0.01
0.02
0.03
0.04
0.05
0
20
40
60
80
N = 1599 Bandwidth = 2.15

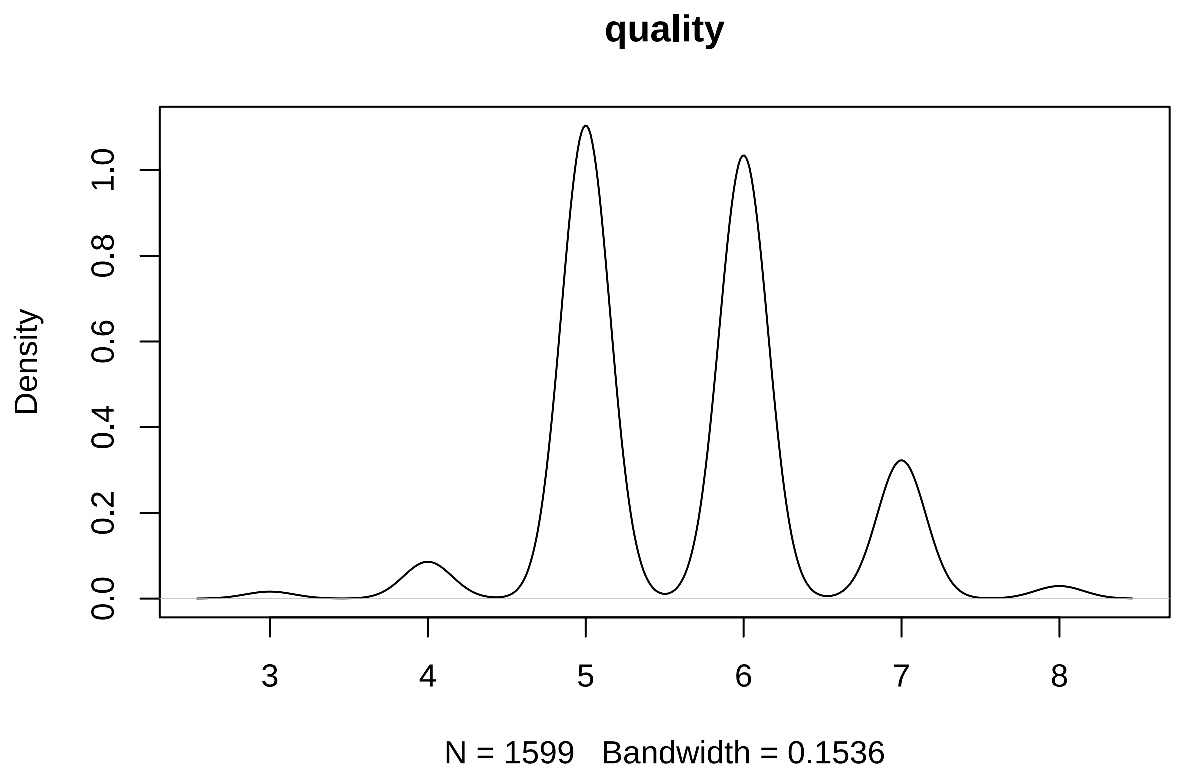

total sulfur dioxide
Density
0.000
0.005
0.010
0.015
0
50
100
150
200
250
300
N = 1597   Bandwidth = 6.145


density
Density
0
50
100
150
200
250
0.990
0.995
1.000
1.005
N = 1599   Bandwidth = 0.0003433


pH
Density
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
N = 1599 Bandwidth = 0.02918

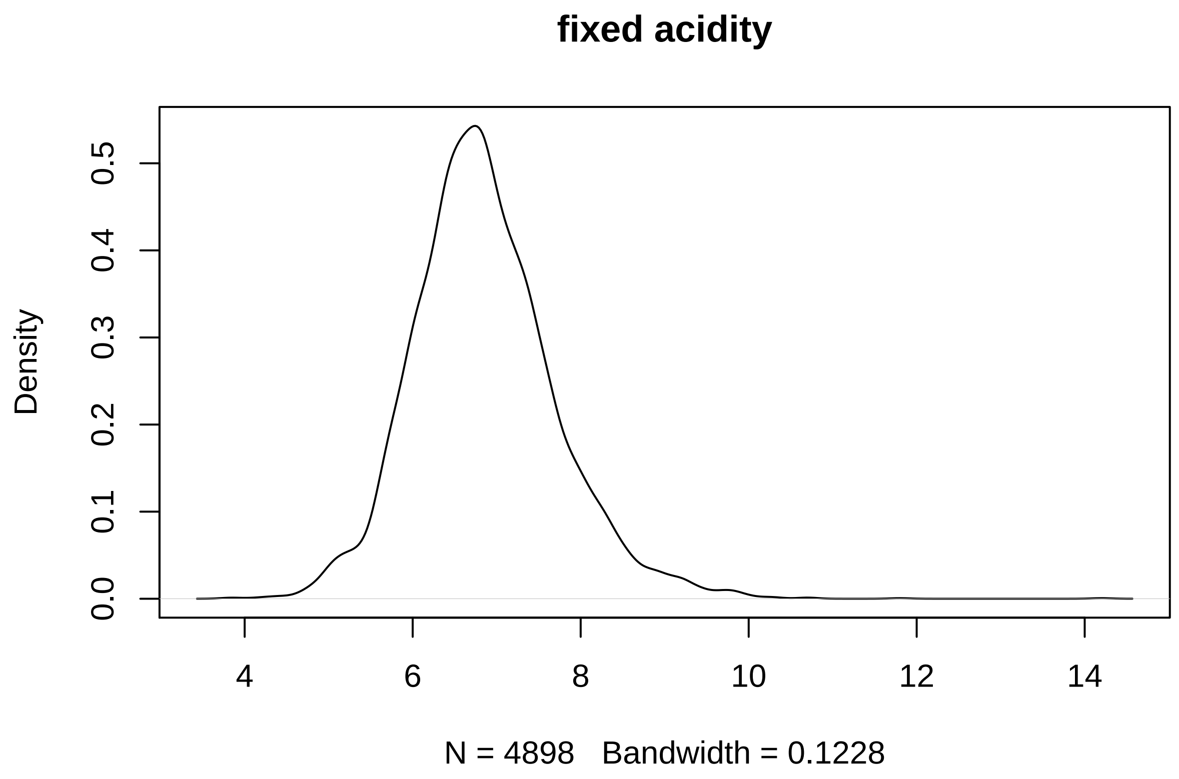

sulphates
Density
0.0
1.0
2.0
3.0
0.5
1.0
1.5
2.0
N = 1599   Bandwidth = 0.02765

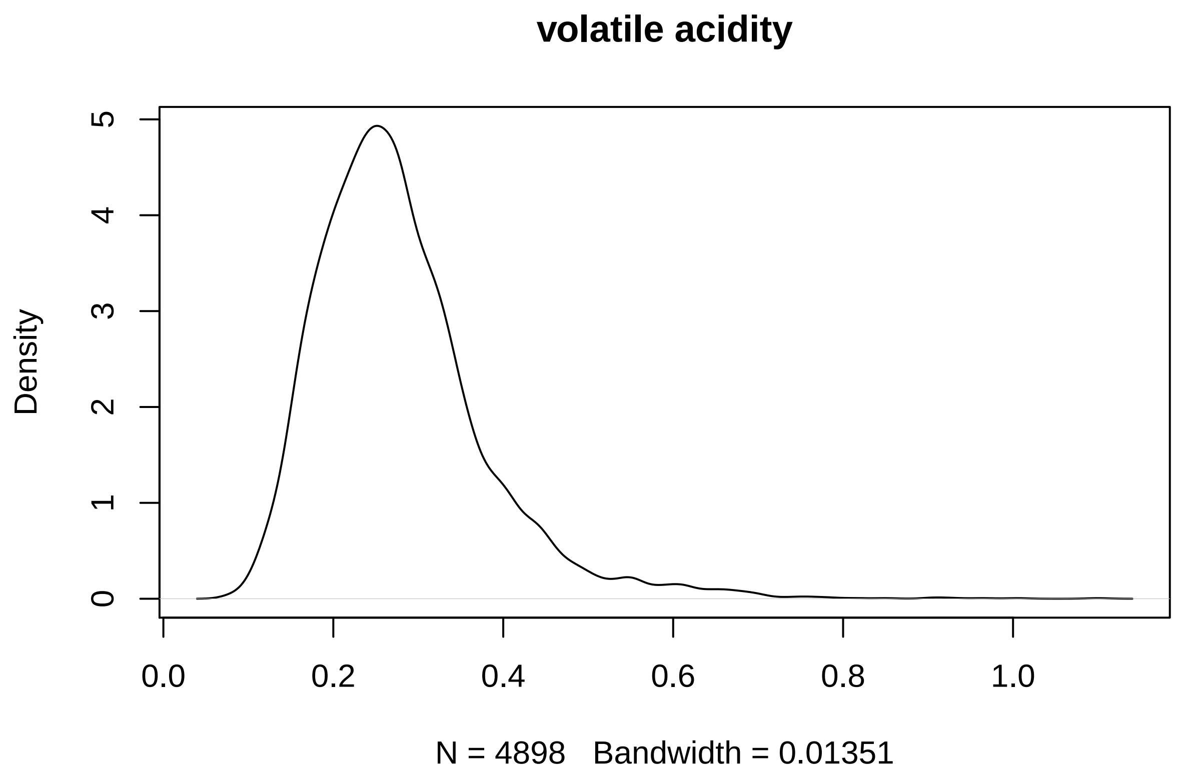

alcohol
Density
0.0
0.1
0.2
0.3
0.4
0.5
8
10
12
14
N = 1599 Bandwidth = 0.2193



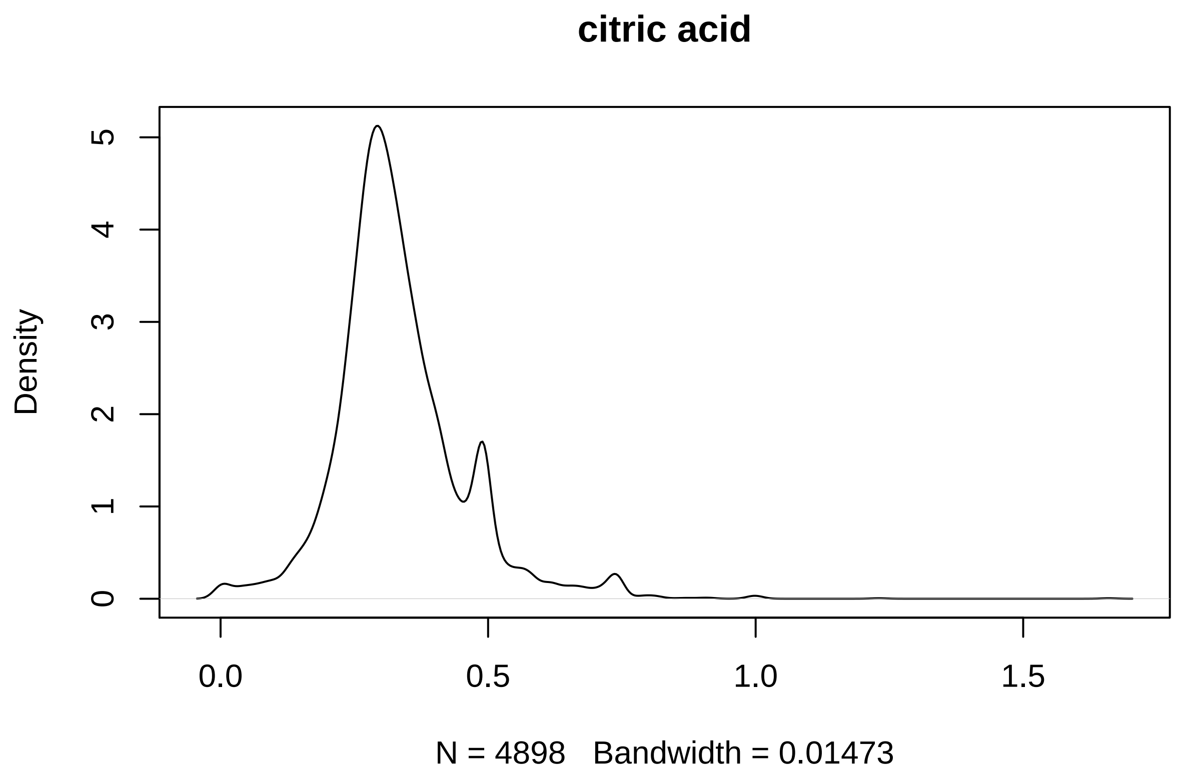
Plots for White DataSet

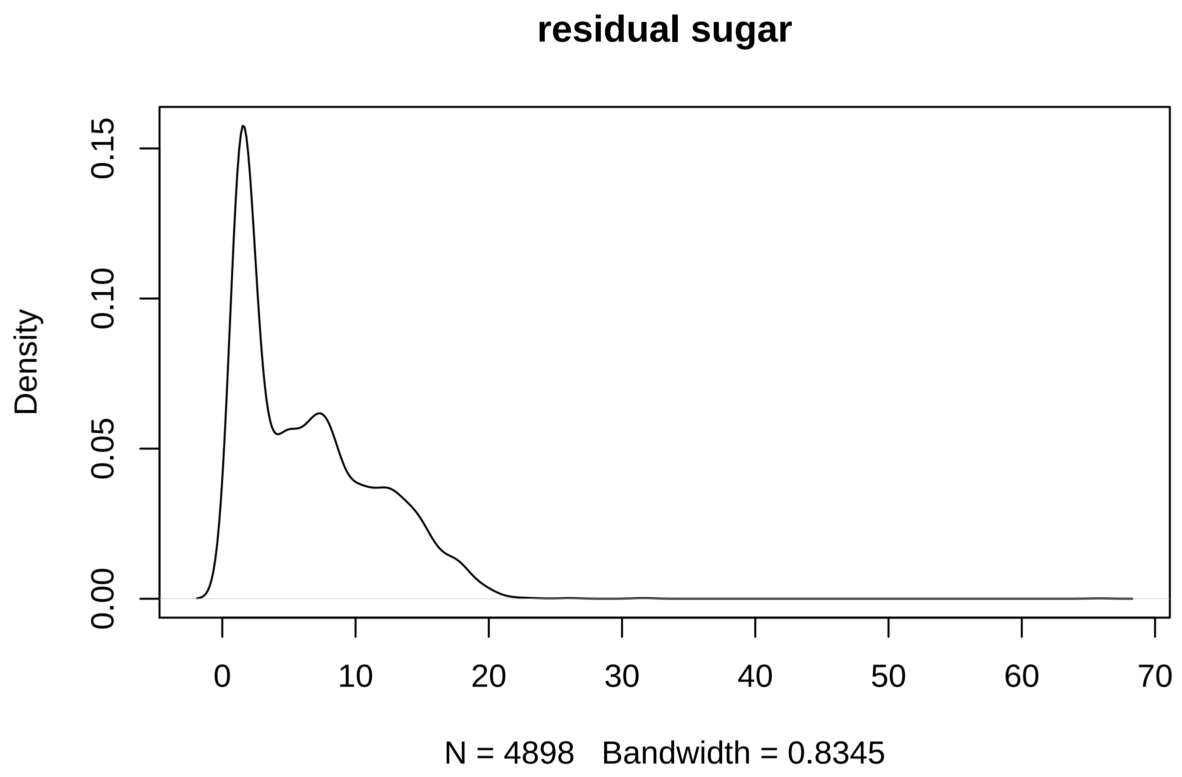
```
for( i in colnames(winequality_white))
{
  d <- density(winequality_white[[i]])
  plot(d,main=i)
}
```

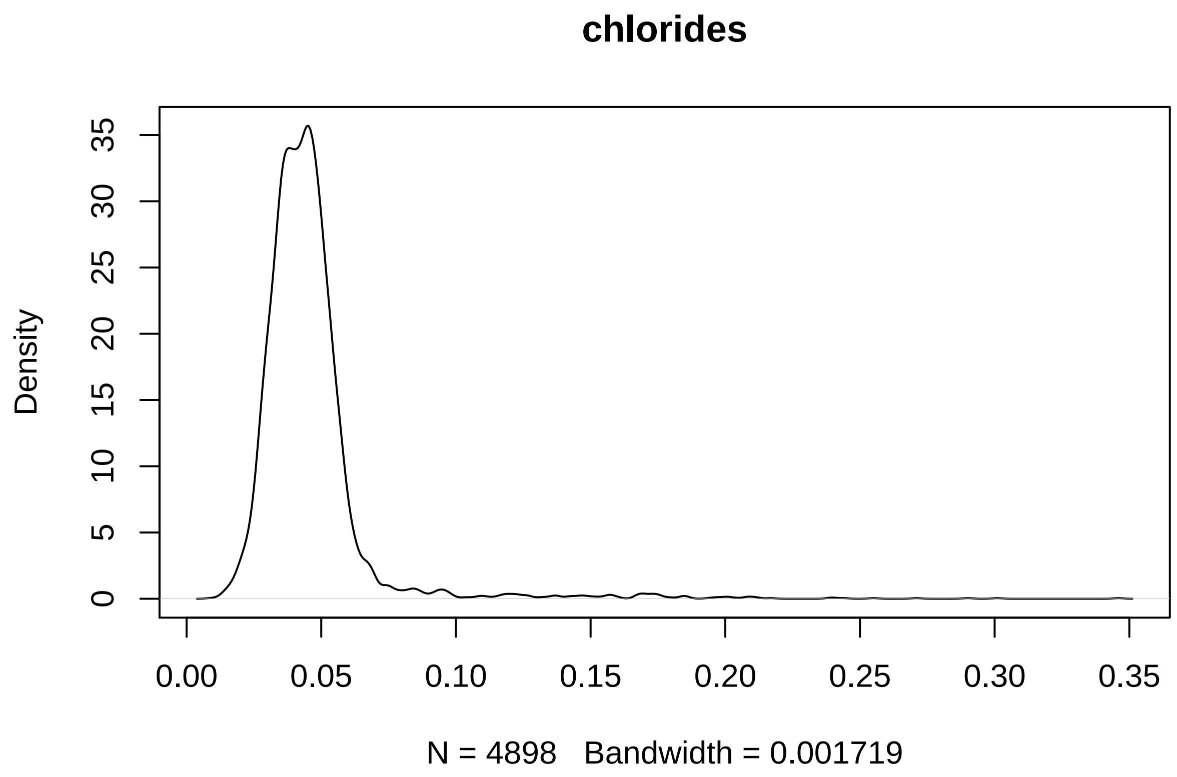

fixed acidity
Density
0.0
0.1
0.2
0.3
0.4
0.5
4
6
8
10
12
14
N = 4898 Bandwidth = 0.1228

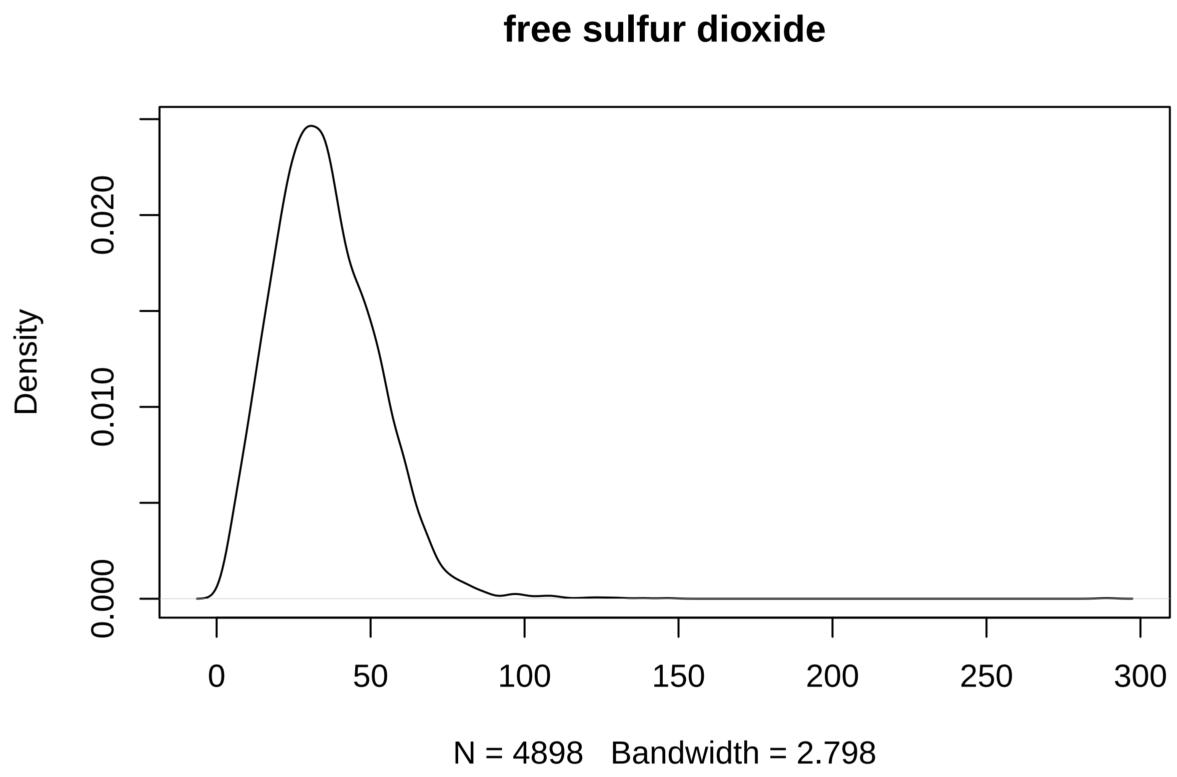


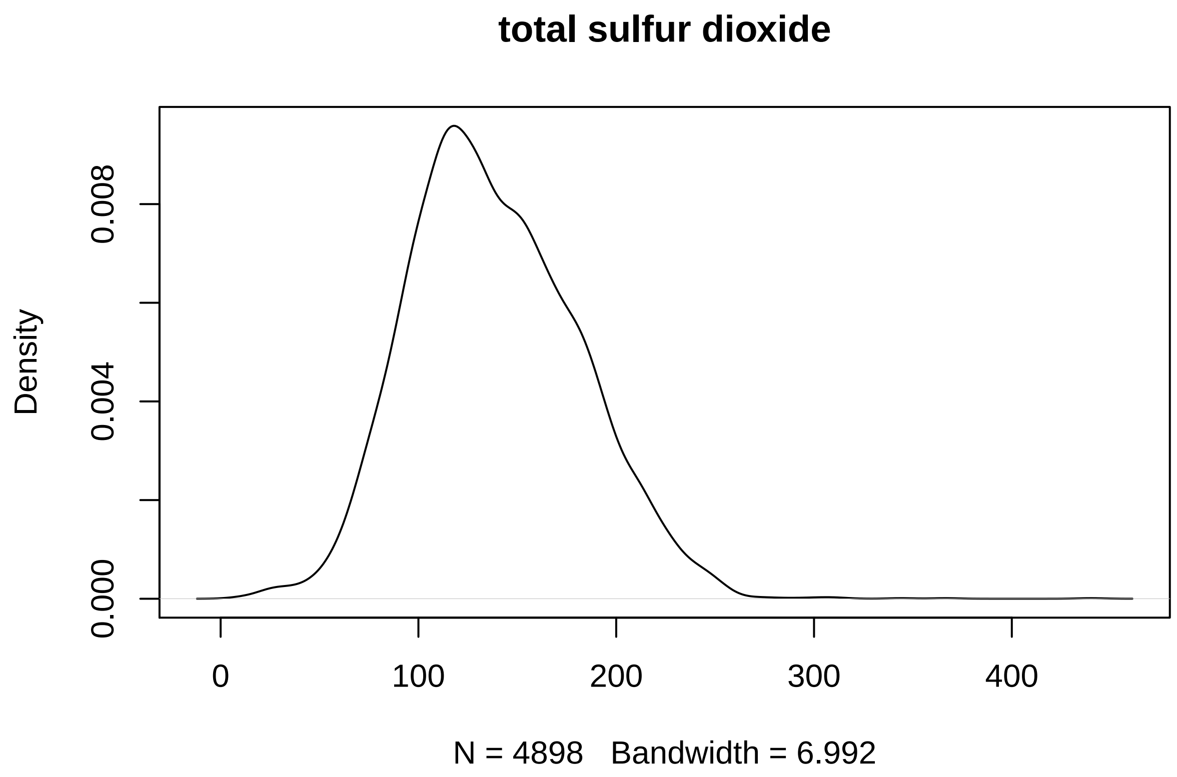
volatile acidity
Density
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
N = 4898   Bandwidth = 0.01351

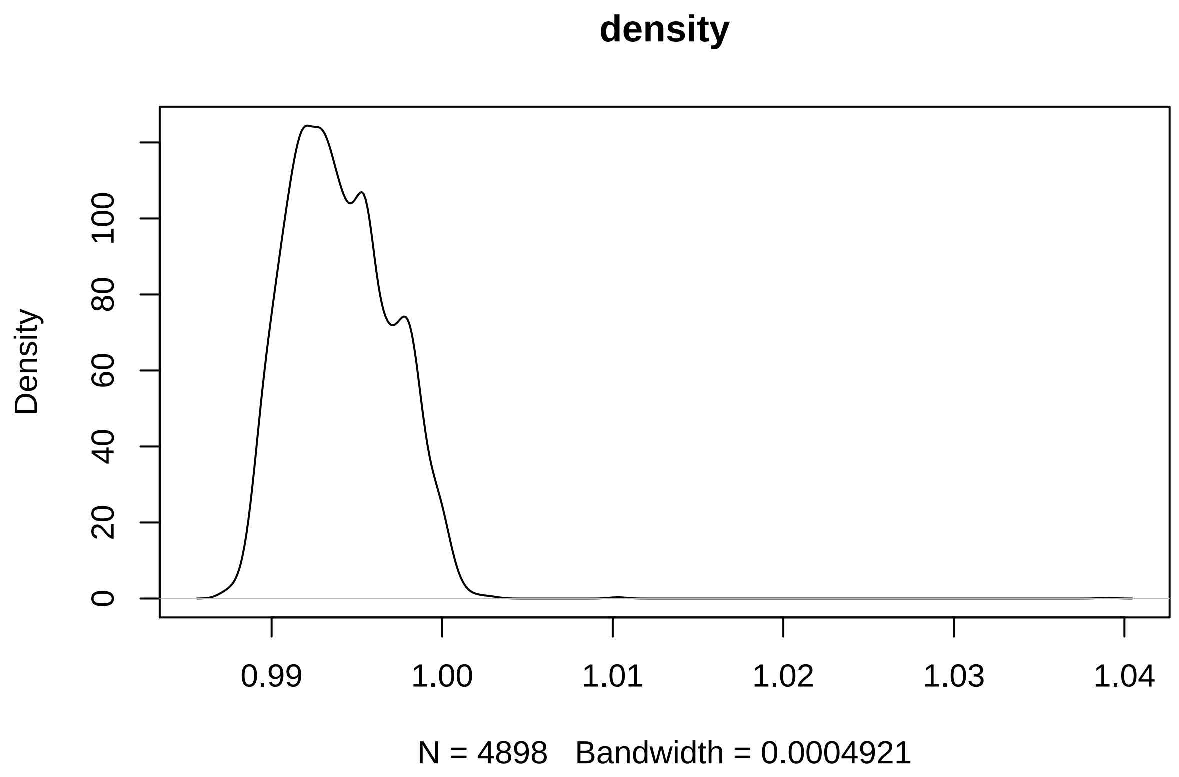

citric acid
Density
0
1
2
3
4
5
0.0
0.5
1.0
1.5
N = 4898   Bandwidth = 0.01473

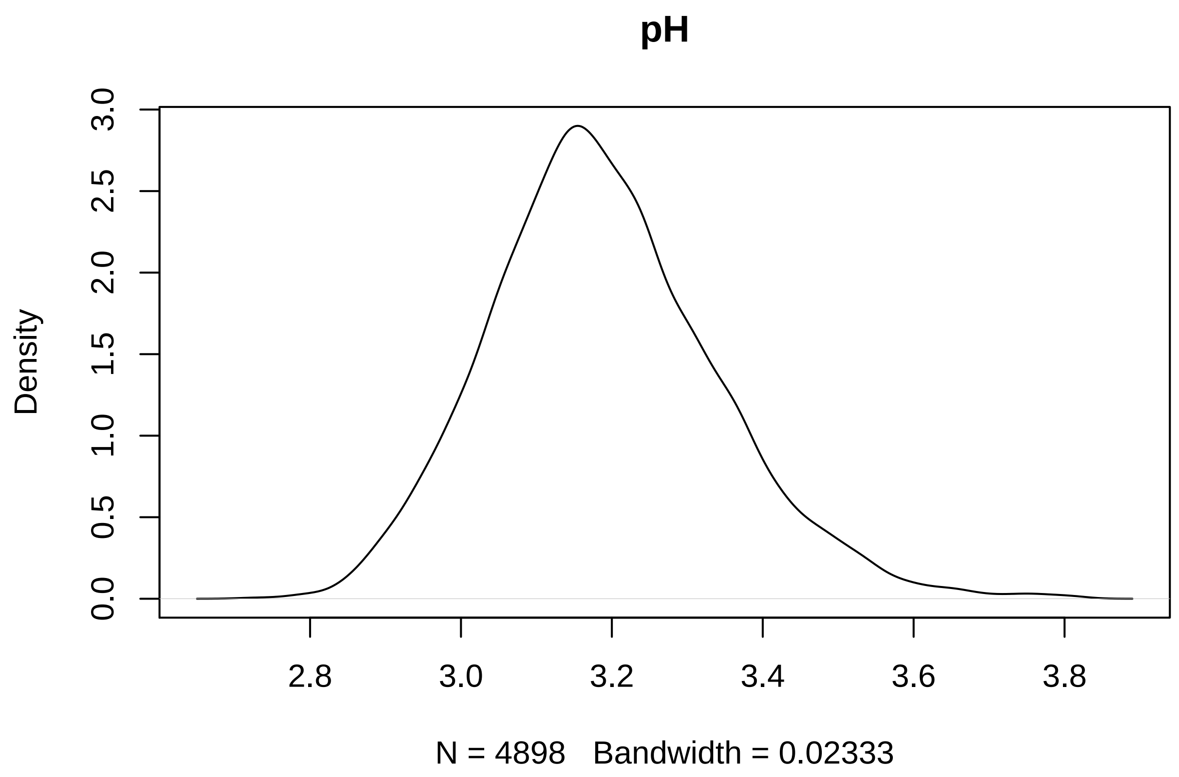

residual sugar
Density
0.00
0.05
0.10
0.15
0
10
20
30
40
50
60
70
N = 4898 Bandwidth = 0.8345

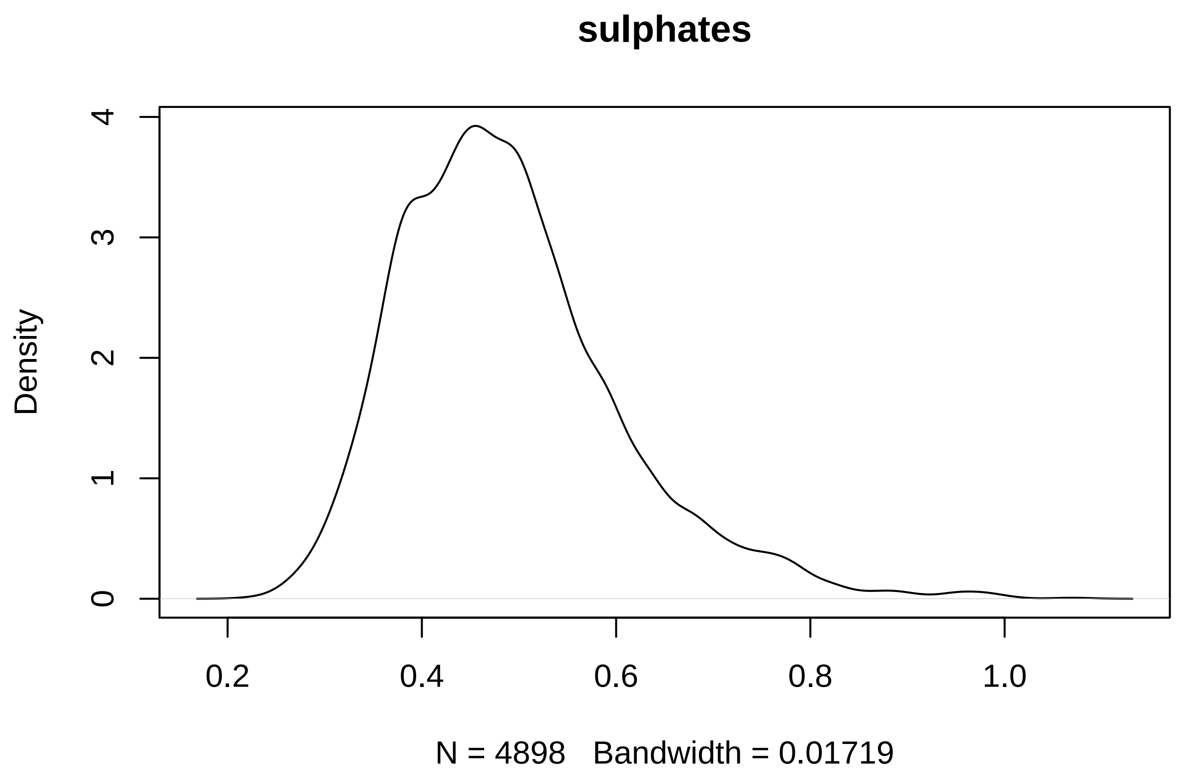

chlorides
Density
0
5
10
15
20
25
30
35
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
N = 4898   Bandwidth = 0.001719

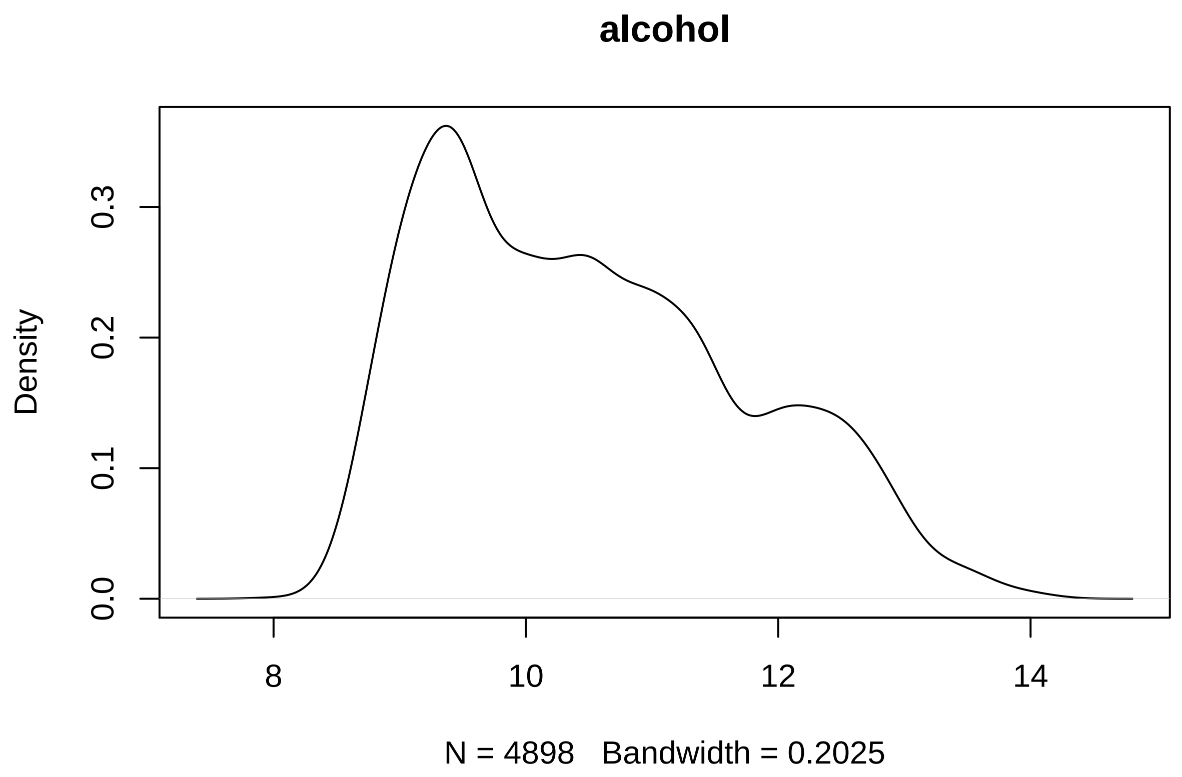

free sulfur dioxide
Density
0.000
0.010
0.020
0
50
100
150
200
250
300
N = 4898 Bandwidth = 2.798

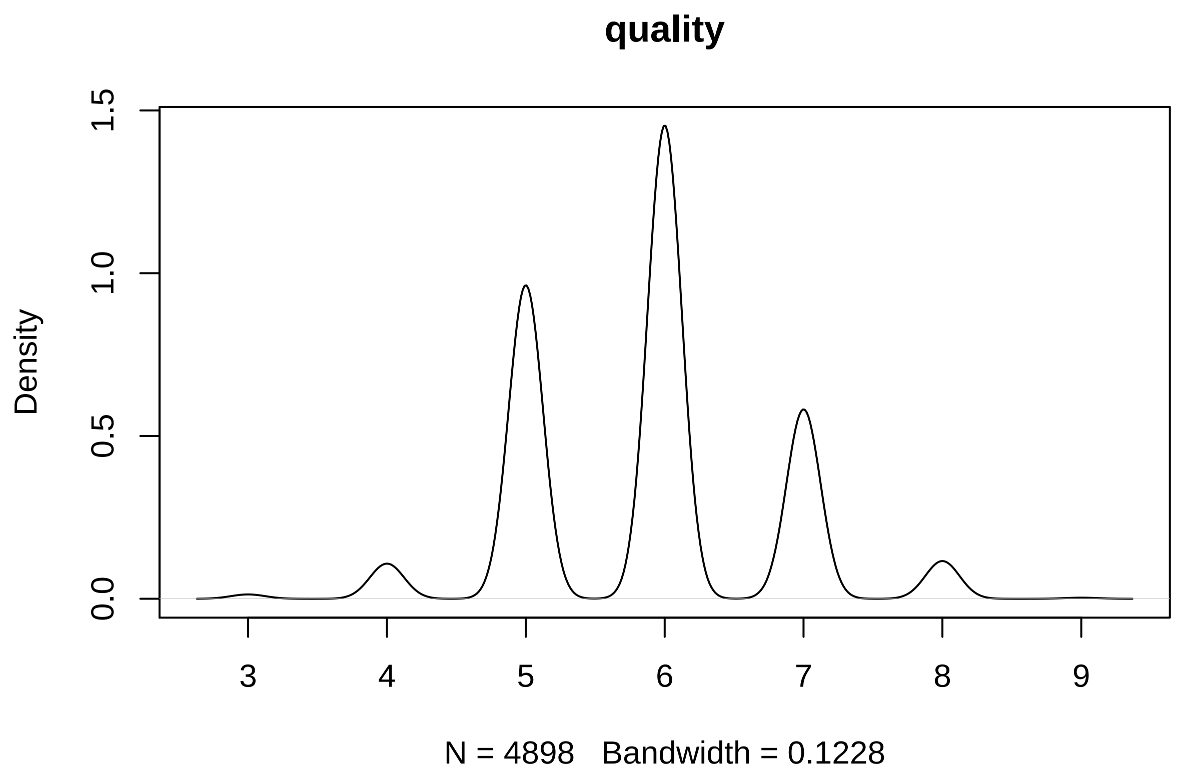

total sulfur dioxide
Density
0.000
0.004
0.008
0
100
200
300
400
N = 4898 Bandwidth = 6.992


density
Density
0
20
40
60
80
100
0.99
1.00
1.01
1.02
1.03
1.04
N = 4898   Bandwidth = 0.0004921


pH
Density
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2.8
3.0
3.2
3.4
3.6
3.8
N = 4898   Bandwidth = 0.02333


sulphates
Density
0
1
2
3
4
0.2
0.4
0.6
0.8
1.0
N = 4898   Bandwidth = 0.01719


alcohol
Density
0.0
0.1
0.2
0.3
8
10
12
14
N = 4898   Bandwidth = 0.2025



```
print("RED WINES")
```

```
## [1] "RED WINES"
```

```
print("==================")
```

```
## [1] "=================="
```

```
for( i in colnames(winequality_red))
{
  print(i)
  print(var(winequality_red[[i]],na.rm=TRUE))
  print("----------")
}
```

```
## [1] "fixed acidity"
## [1] 3.031416
## [1] "----------"
## [1] "volatile acidity"
## [1] 0.03206238
## [1] "----------"
## [1] "citric acid"
## [1] 0.03794748
## [1] "----------"
## [1] "residual sugar"
## [1] 1.987897
## [1] "----------"
## [1] "chlorides"
## [1] 0.002215143
## [1] "----------"
## [1] "free sulfur dioxide"
## [1] 109.4149
## [1] "----------"
## [1] "total sulfur dioxide"
## [1] 1082.25
## [1] "----------"
## [1] "density"
## [1] 3.562029e-06
## [1] "----------"
## [1] "pH"
## [1] 0.02383518
## [1] "----------"
## [1] "sulphates"
## [1] 0.02873262
## [1] "----------"
## [1] "alcohol"
## [1] 1.135647
## [1] "----------"
## [1] "quality"
## [1] 0.6521684
## [1] "----------"
```

```r
print("WHITE WINES")
```

```
## [1] "WHITE WINES"
```

```r
print("==================")
```

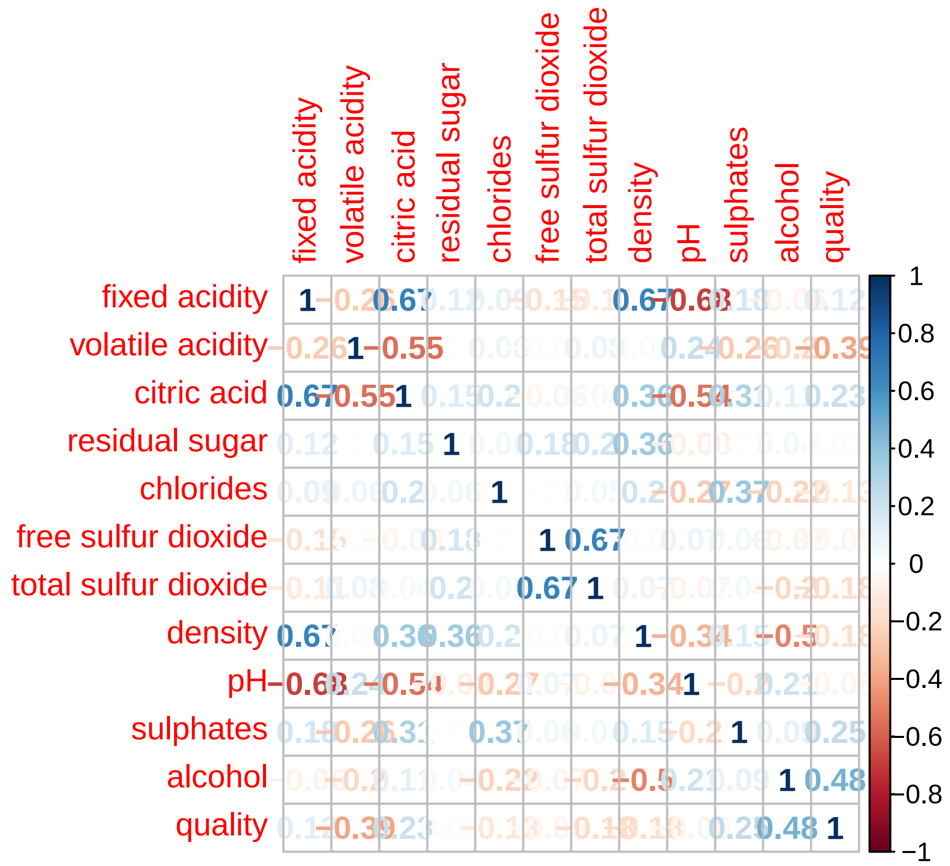
```
## [1] "=================="
```

```r
for( i in colnames(winequality_red))
{
  print(i)
  print(var(winequality_red[[i]],na.rm=TRUE))
  print("----------")
}
```

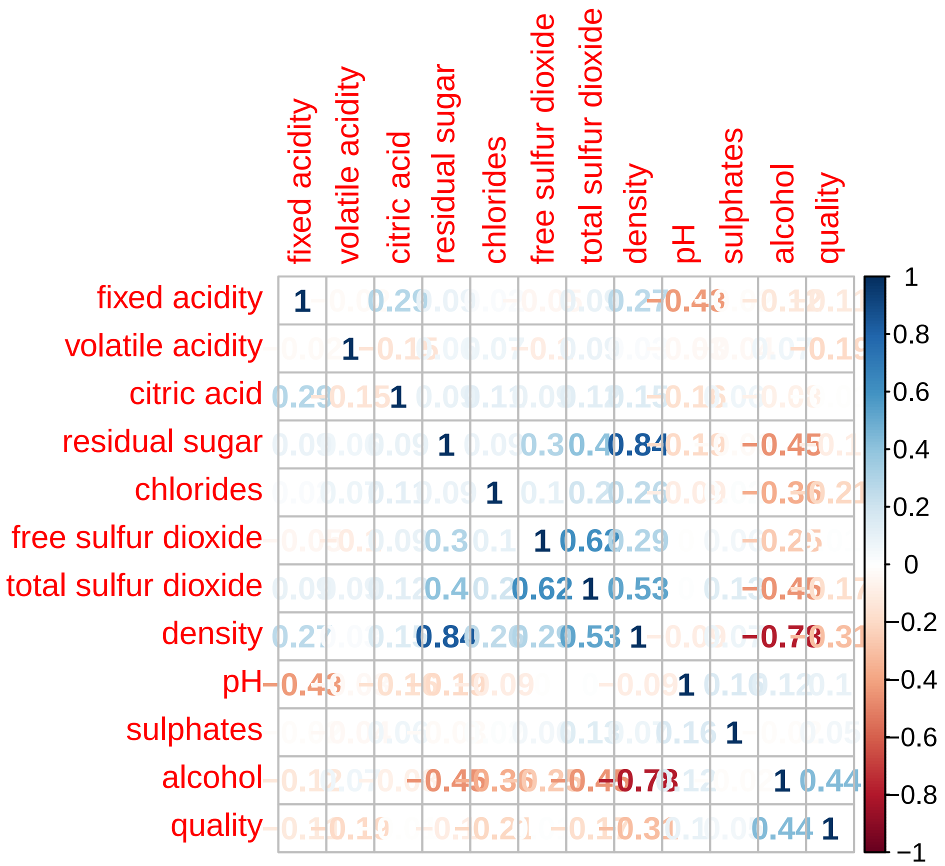
```
## [1] "fixed acidity"
## [1] 3.031416
## [1] "----------"
## [1] "volatile acidity"
## [1] 0.03206238
## [1] "----------"
## [1] "citric acid"
## [1] 0.03794748
## [1] "----------"
## [1] "residual sugar"
## [1] 1.987897
## [1] "----------"
## [1] "chlorides"
## [1] 0.002215143
## [1] "----------"
## [1] "free sulfur dioxide"
## [1] 109.4149
## [1] "----------"
## [1] "total sulfur dioxide"
## [1] 1082.25
## [1] "----------"
## [1] "density"
## [1] 3.562029e-06
## [1] "----------"
## [1] "pH"
## [1] 0.02383518
## [1] "----------"
## [1] "sulphates"
## [1] 0.02873262
## [1] "----------"
## [1] "alcohol"
## [1] 1.135647
## [1] "----------"
## [1] "quality"
## [1] 0.6521684
## [1] "----------"
```

EDA Question 2- Variance of each column

```r
corrplot(cor(na.omit(winequality_red)),method = 'number')
```



```r
corrplot(cor(winequality_white),method = 'number')
```


fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

EDA Question 3- Feature Selection

Having explored both data set's I will pick red wines as it has a more interesting set of correlations between it's features and generally higher variability overall since it's a larger dataset.

Based on the results of the correlation analysis it's clear that the best features with the highest correlation in the data set chosen are the "Volatile Acidity" feature and the "Alcohol" feature. Using these two, I will prepare the data set for analysis.

```r
wq_red_df <- data.frame(winequality_red$`volatile acidity`,
                        winequality_red$`alcohol`,
                        winequality_red$`quality`)
```

```r
data_size = length(wq_red_df$winequality_red..volatile.acidity.)
train_val = .8*data_size
test_val = train_val + 1

response_var_train <- winequality_red$`quality`[1:train_val]
response_var_test <- winequality_red$`quality`[test_val:data_size]

wq_data <- data.frame(rep(1,length(winequality_red$`quality`)),
                              wq_red_df$winequality_red..volatile.acidity.,
                              wq_red_df$winequality_red.alcohol)
wq_data_matrix <- data.matrix(wq_data)

wq_data_matrix_train <- wq_data_matrix[1:train_val,]
wq_data_matrix_test <- wq_data_matrix[test_val:data_size,]
```

Analysis

Analysis Question 1- Multivariate Linear Regression

As per eqution (5.35) from the I2ML course text book:

$$r^t = w_0 + w_1 x_1^t + w_2 x_2^t + ... + w_d x_d^t + \epsilon$$

Now I train the model to determine the weights of the regression. This is based off of the derived formula from the book (5.39):

```r
weights = ginv(t(wq_data_matrix_train) %*% wq_data_matrix_train) %*%
          t(wq_data_matrix_train) %*% response_var_train
```

Calculating Epsilon:

```r
response_on_training = vector()

for(i in 1:length(response_var_train))
{
  response_on_training[i] = weights[1] +
    wq_data_matrix_train[i,2]*weights[2] +
    wq_data_matrix_train[i,3]*weights[3]
}

#this is the expected error
epsilon = 0
for(i in 1:length(response_var_train))
{
 epsilon = epsilon + response_on_training[i] - weights[1] -
    wq_data_matrix_train[i,2]*weights[2] -
    wq_data_matrix_train[i,3]*weights[3]
}
epsilon = (0.5)*(epsilon^2)
```

An explanation of the vector that is produced. The first value is

$$w_0$$

which is the intercept. The rest of the values are the ordered weights of the variables.

With my weights produced, I can now test my model against the test data:

```r
response_test = vector()
for(i in 1:length(response_var_test))
{
  response_test[i] = weights[1] +
    wq_data_matrix_test[i,2]*weights[2] +
    wq_data_matrix_test[i,3]*weights[3]
}
```

Finally, to test the performance of the model, I will calculate the RMSE and the R-Squared:

```r
#RMSE
rmse_of_model = sqrt(sum((response_test-response_var_test)^2))

#R-Squared
```

```r
ssr = sum((response_test-response_var_test)^2)
ss_total = sum((response_test-mean(response_test))^2)
rsquared = (ss_total - ssr)/ss_total

print("RMSE:")
```

```
## [1] "RMSE:"
```

```r
print(rmse_of_model)
```

```
## [1] 12.08537
```

```r
print("R-Squared:")
```

```
## [1] "R-Squared:"
```

```r
print(rsquared)
```

```
## [1] -1.638753
```

As can be seen by the above values, it's quite clear that this model performs very poorly. As this model does have an intercept as mentioned above, the RSquared value being negative is an indication of an extremely poor fit to the data.

Analysis Question 2- Multivariate Quadratic Regression (Higher Order Polynomial)

I will attempt to see if a quadratic regression with a higher order of polynomial can produce a better result.

As before I will first train the data.

```
wq_data_matrix_train =  cbind(wq_data_matrix_train,wq_data_matrix_train[,2]^2)
wq_data_matrix_train =  cbind(wq_data_matrix_train,wq_data_matrix_train[,2]*wq_data_matrix_train[,3])
wq_data_matrix_train =  cbind(wq_data_matrix_train,wq_data_matrix_train[,3]^2)

weights = ginv(t(wq_data_matrix_train) %*% wq_data_matrix_train) %*%
          t(wq_data_matrix_train) %*% response_var_train
```

Caclulating epsilon:

```
response_on_training = vector()

for(i in 1:length(response_var_train))
{
  response_on_training[i] = weights[1] +
    wq_data_matrix_train[i,2]*weights[2] +
    wq_data_matrix_train[i,3]*weights[3] +
    wq_data_matrix_train[i,4]*weights[4] +
    wq_data_matrix_train[i,5]*weights[5] +
    wq_data_matrix_train[i,6]*weights[6]
}

#this is the expected error
epsilon = 0
for(i in 1:length(response_var_train))
{
 epsilon = epsilon + response_on_training[i] - weights[1] -
    wq_data_matrix_train[i,2]*weights[2] -
    wq_data_matrix_train[i,3]*weights[3] -
    wq_data_matrix_train[i,4]*weights[4] -
    wq_data_matrix_train[i,5]*weights[5] -
    wq_data_matrix_train[i,6]*weights[6]

}
epsilon = (0.5)*(epsilon^2)
```

With my weights produced, I can now test my model against the test data:

```
response_test = vector()

wq_data_matrix_test =  cbind(wq_data_matrix_test,wq_data_matrix_test[,2]^2)
wq_data_matrix_test =  cbind(wq_data_matrix_test,wq_data_matrix_test[,2]*wq_data_matrix_test[,3])
wq_data_matrix_test =  cbind(wq_data_matrix_test,wq_data_matrix_test[,3]^2)

for(i in 1:length(response_var_test))
{
  response_test[i] = weights[1] +
    wq_data_matrix_test[i,2]*weights[2] +
    wq_data_matrix_test[i,3]*weights[3] +
    wq_data_matrix_test[i,4]*weights[4] +
    wq_data_matrix_test[i,5]*weights[5] +
    wq_data_matrix_test[i,6]*weights[6]
}
```

Finally, to test the performance of the model, I will calculate the RMSE and the R-Squared:

```
#RMSE
rmse_of_model = sqrt(sum((response_test-response_var_test)^2))

#R-Squared
ssr = sum((response_test-response_var_test)^2)
ss_total = sum((response_test-mean(response_test))^2)
rsquared = (ss_total - ssr)/ss_total

print("RMSE:")
```

```
## [1] "RMSE:"
```

```
print(rmse_of_model)
```

```
## [1] 12.11713
```

```
print("R-Squared:")
```

```
## [1] "R-Squared:"
```

```
print(rsquared)
```

```
## [1] -1.619055
```

Analysis Question 3 - Which order gives best performance

I did not have time to plot or look at any order greater than 2. That being said, an order of two didn't majorily improve performance so I'm not expecting test error to go down much further with higher orders. That also being said, as per Ch 4 in the book we have a bias-variance trade off to consider. The higher the fit, the more susceptible it will be to variance and thus will not generalise well. I think a linear model is appropriate here given the data.