MSiA 400 Lab Assignment 4

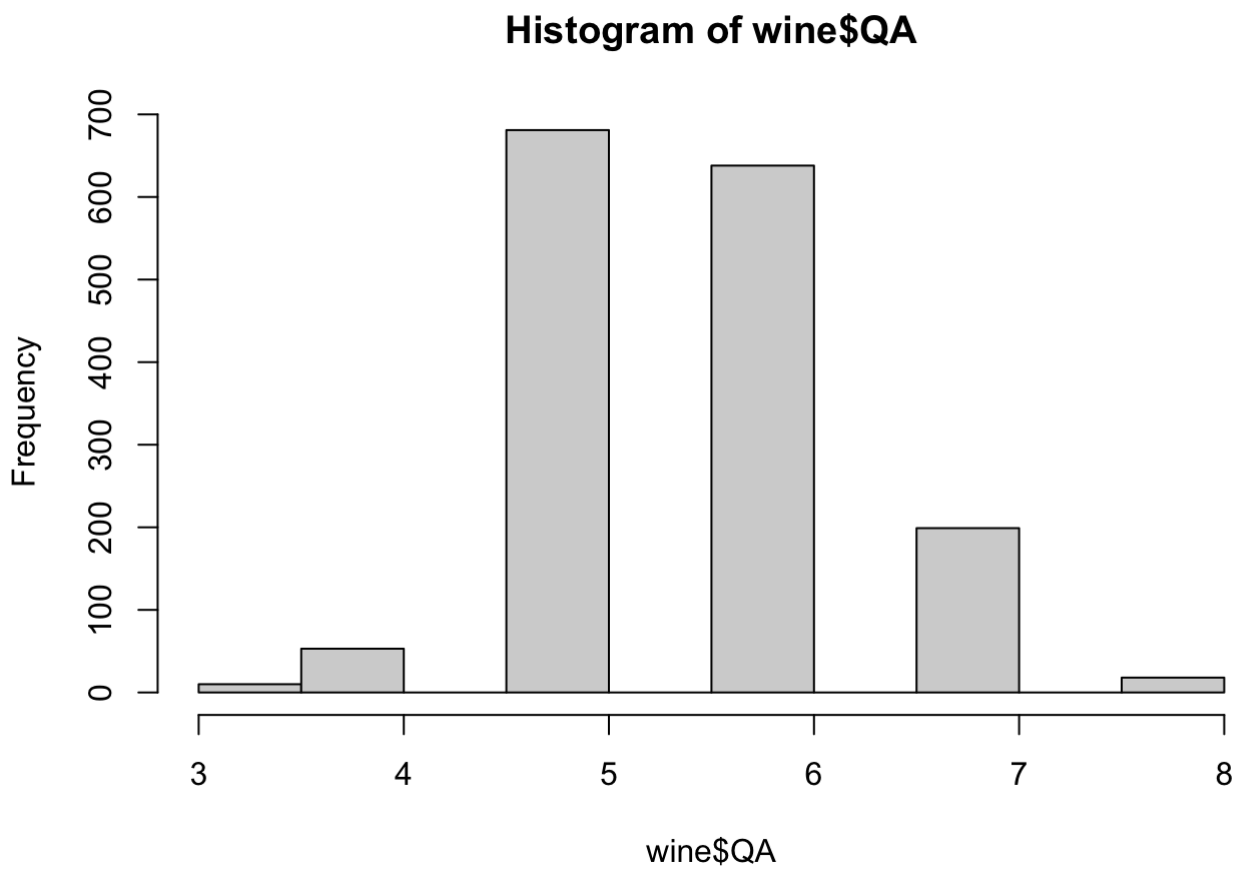
Shenglang Zhou: szg1224

Problem 1a)

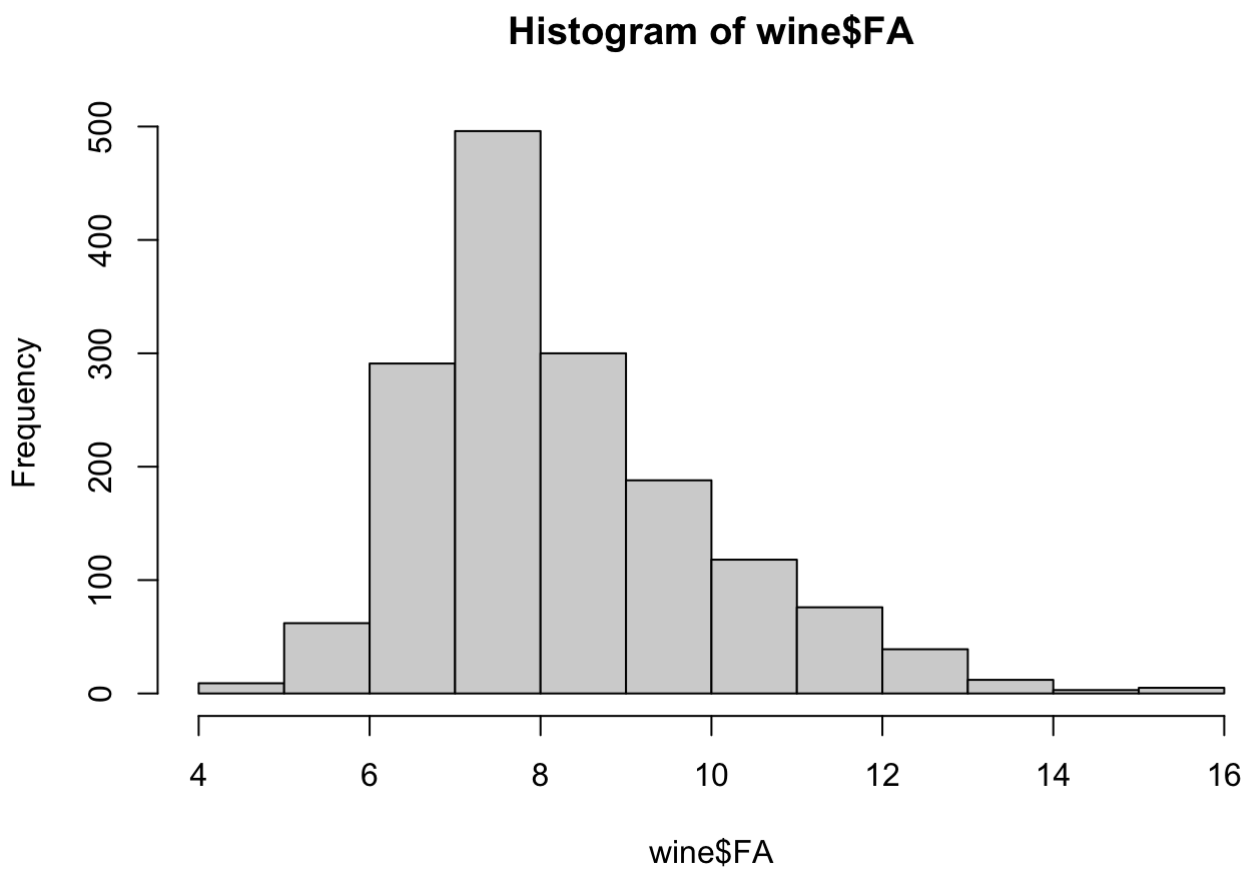
In this question, I will plot everyone of them using histogram

```
setwd("/Users/dylanchou/Desktop/MSiA400")
wine = read.delim("redwine.txt")
```

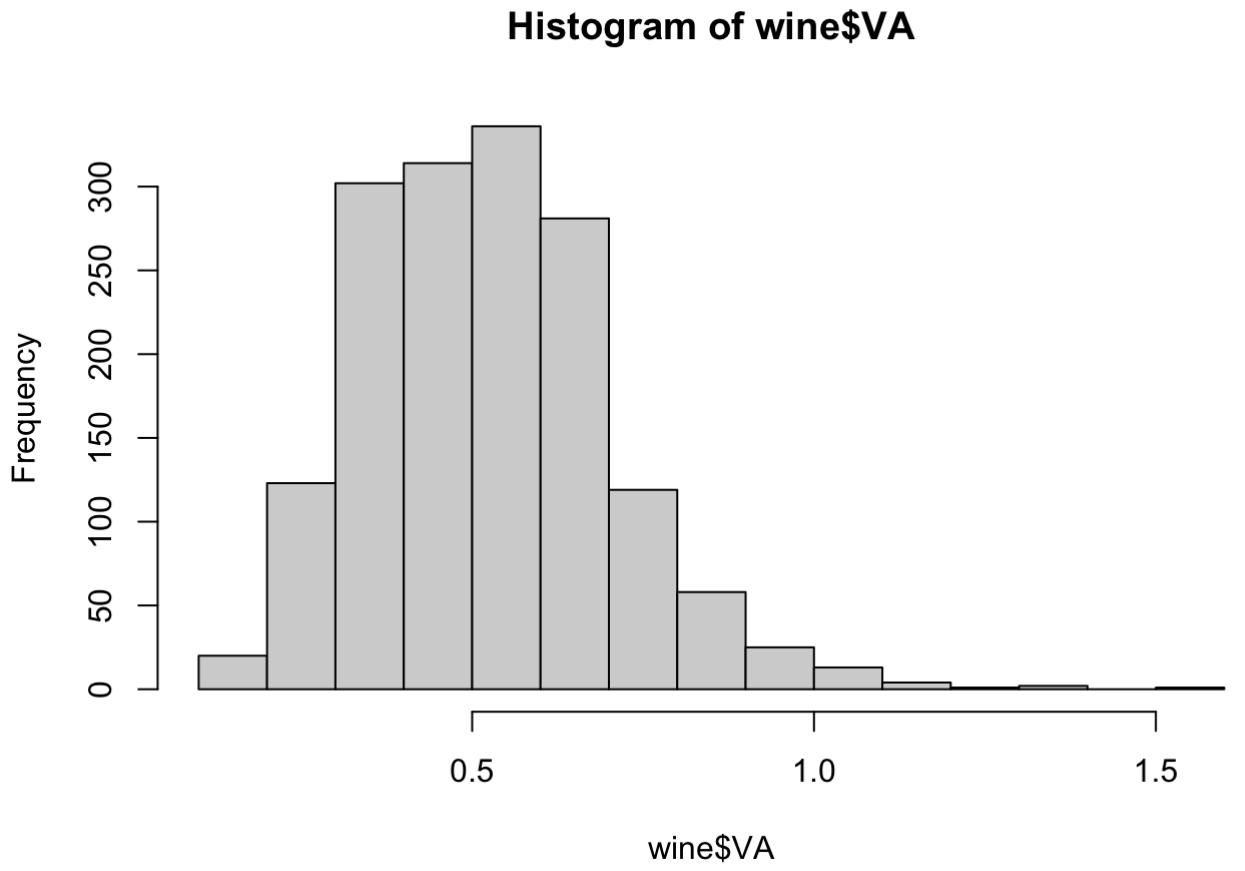
```
hist(wine$QA)
```



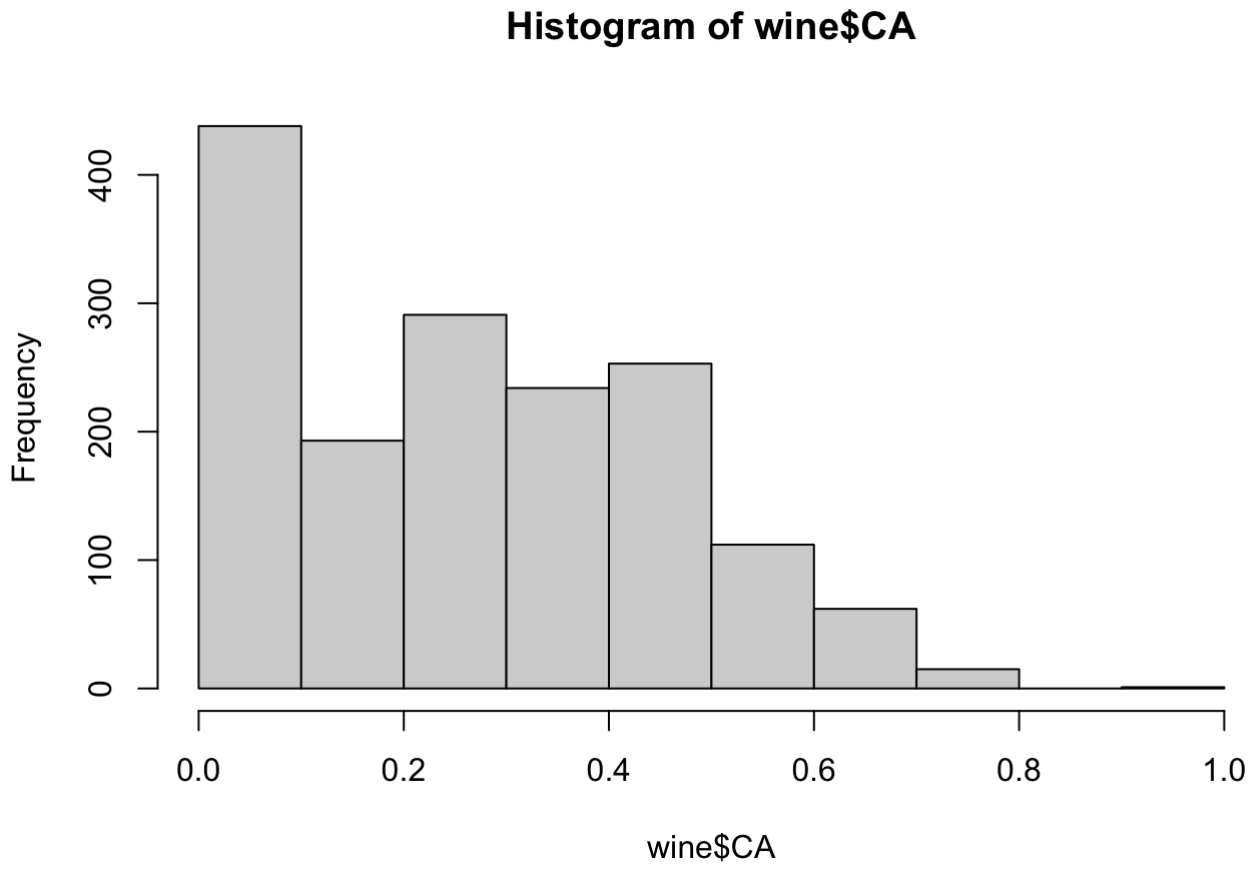

```
hist(wine$FA)
```


Histogram of wine$FA
Frequency
0
100
200
300
400
500
4
6
8
10
12
14
16
wine$FA

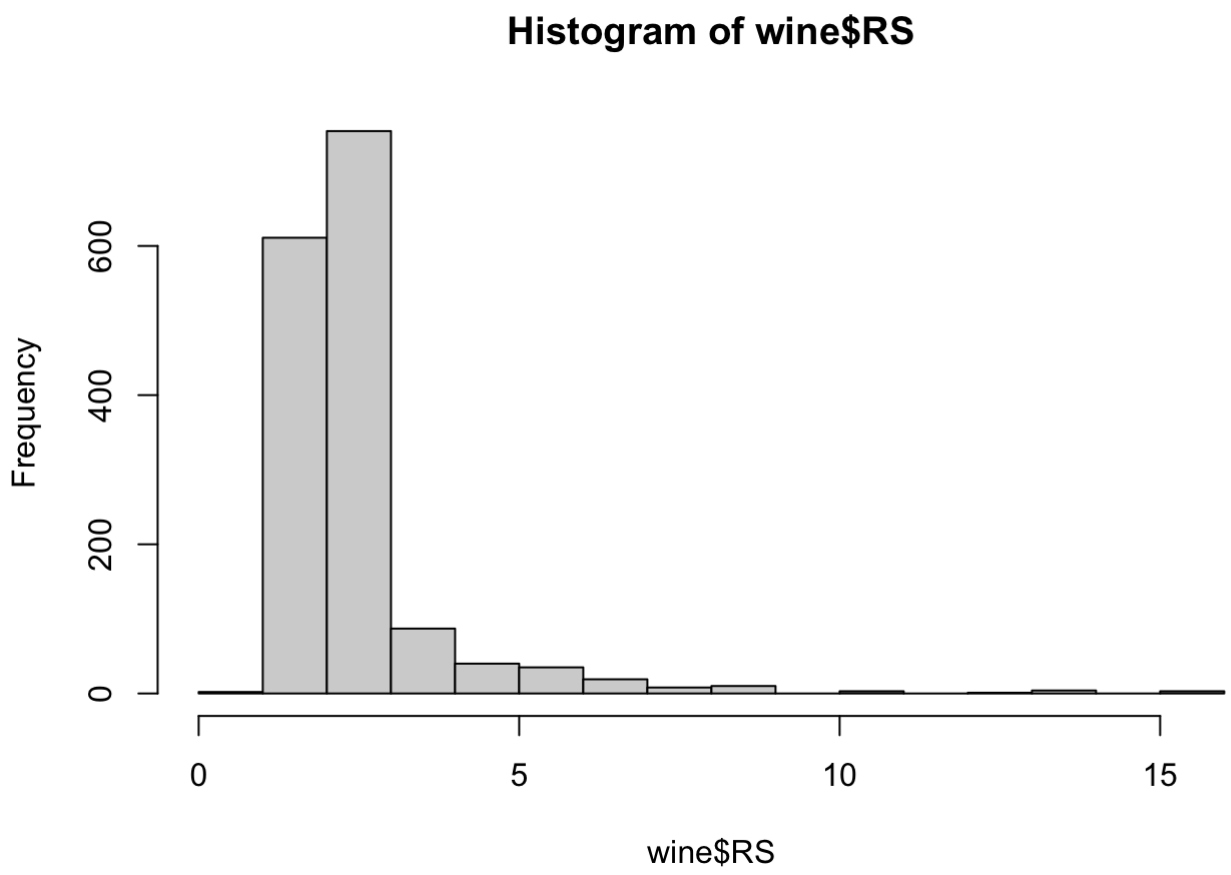
```
hist(wine$VA)
```


Histogram of wine$VA
Frequency
0
50
100
150
200
250
300
0.5
1.0
1.5
wine$VA

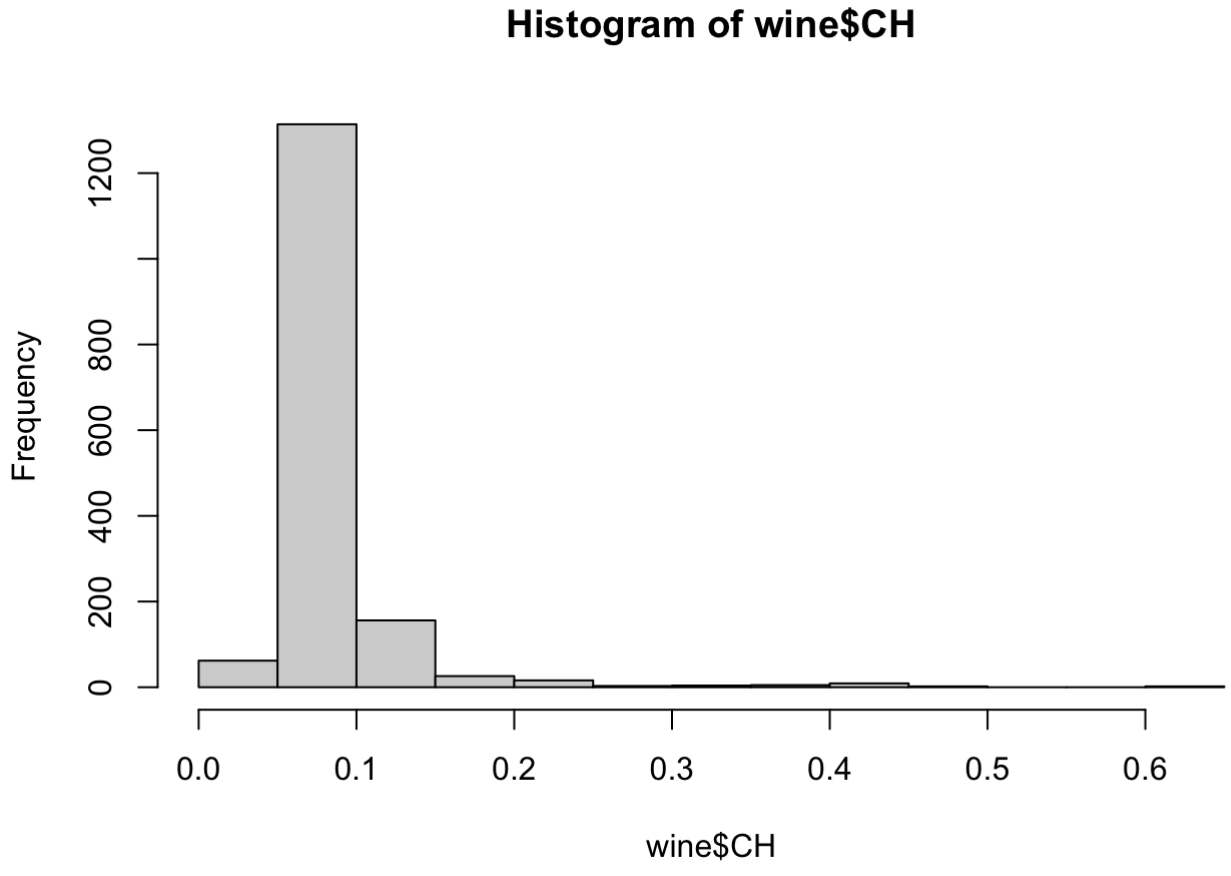
```
hist(wine$CA)
```



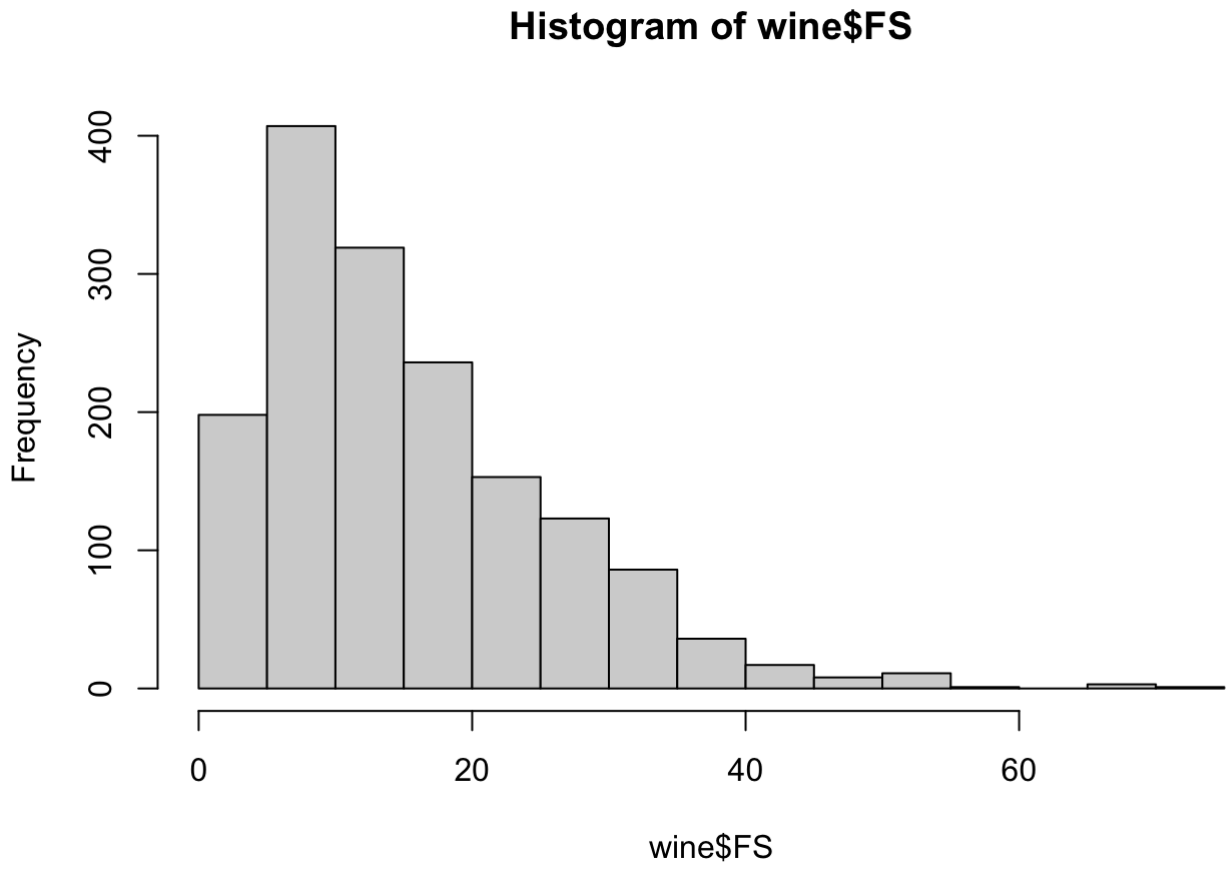

```
hist(wine$RS)
```


Histogram of wine$RS
Frequency
0
200
400
600
0
5
10
15
wine$RS

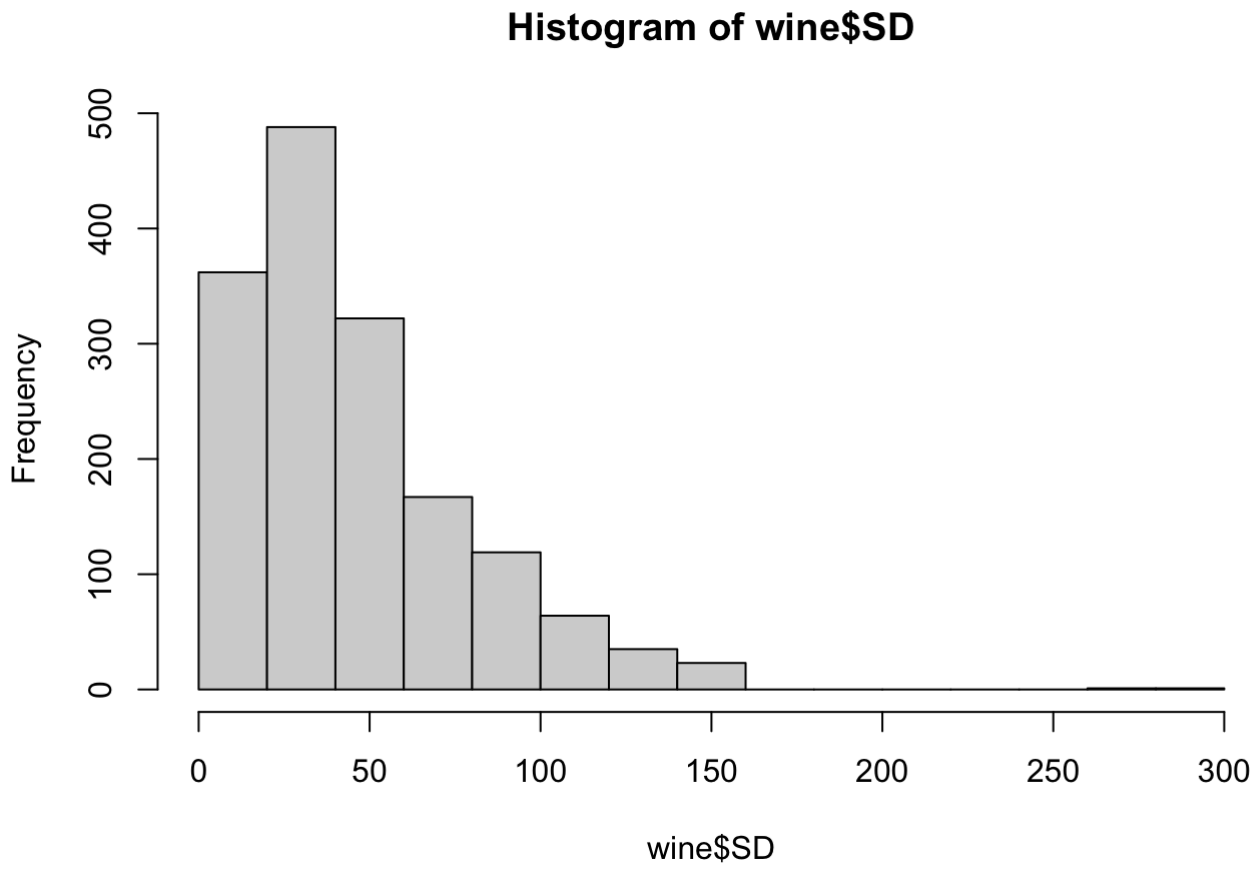
```
hist(wine$CH)
```


Histogram of wine$CH
Frequency
0
200
400
600
800
1200
0.0
0.1
0.2
0.3
0.4
0.5
0.6
wine$CH

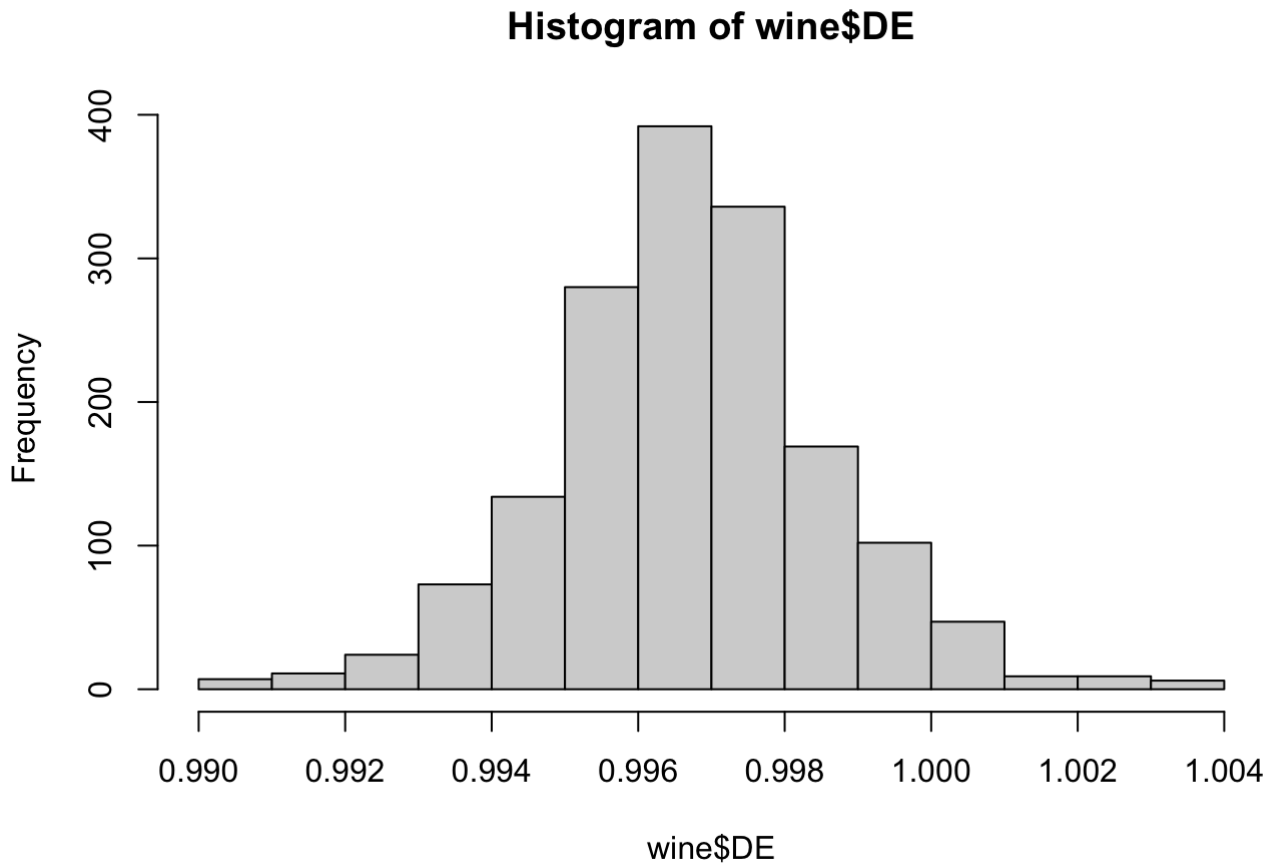
```
hist(wine$FS)
```



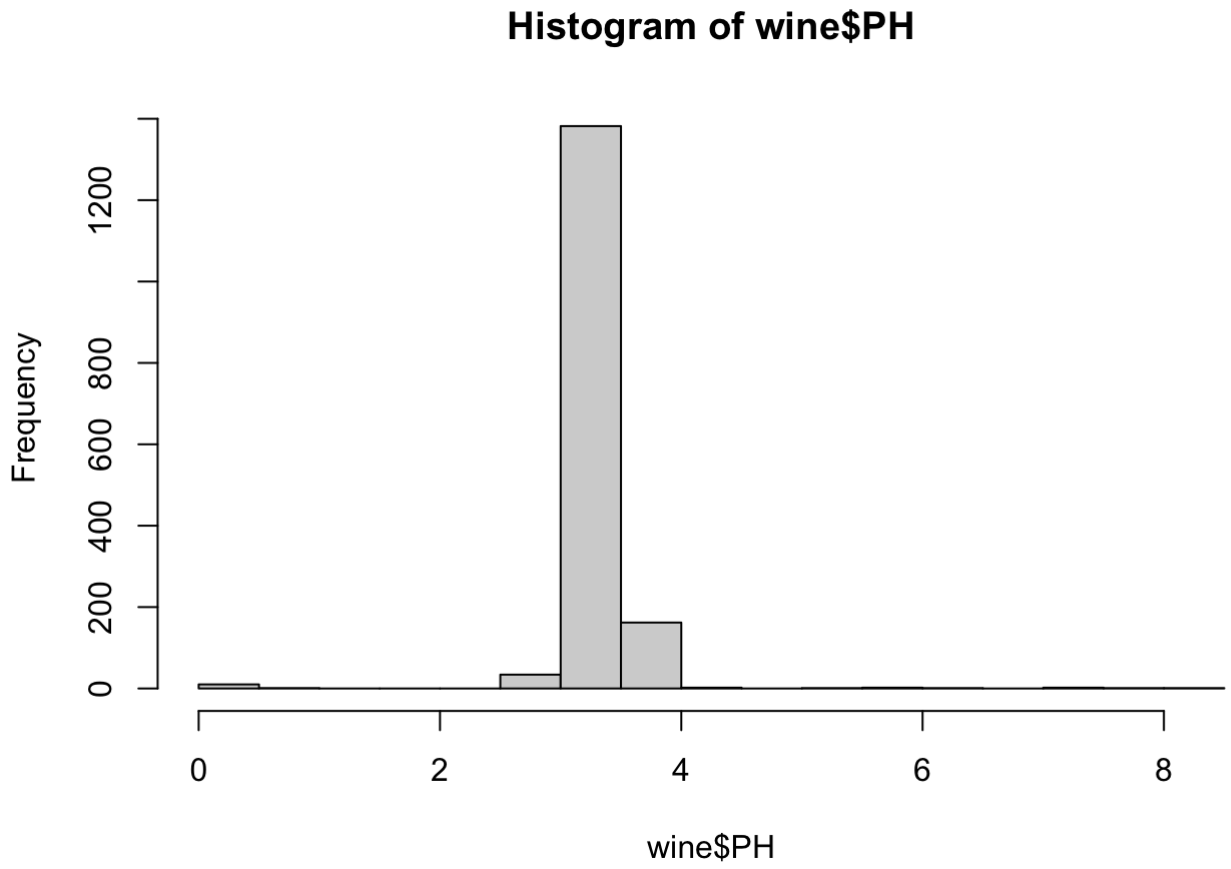

```
hist(wine$SD)
```


Histogram of wine$SD
Frequency
0
100
200
300
400
500
0
50
100
150
200
250
300
wine$SD

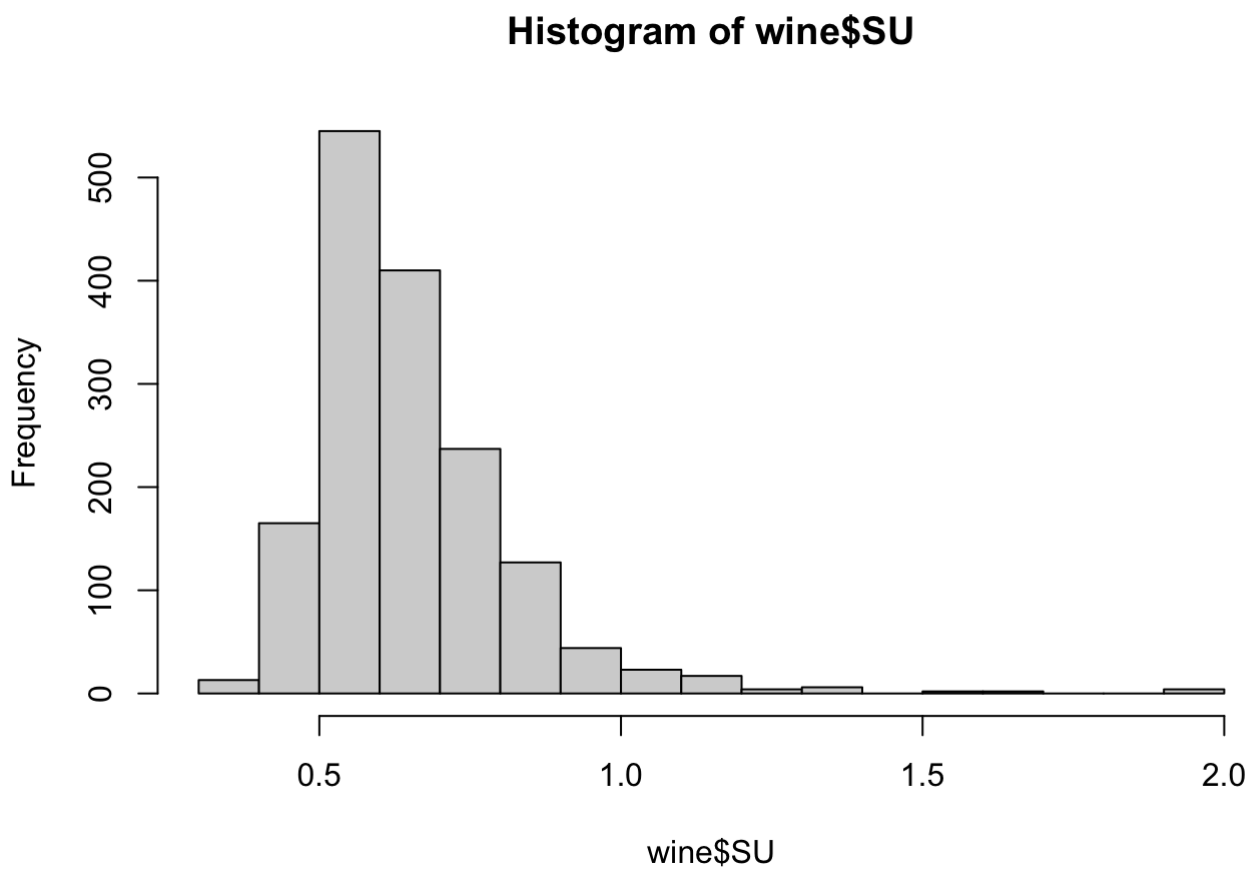
```
hist(wine$DE)
```


Histogram of wine$DE
Frequency
0
100
200
300
400
0.990
0.992
0.994
0.996
0.998
1.000
1.002
1.004
wine$DE

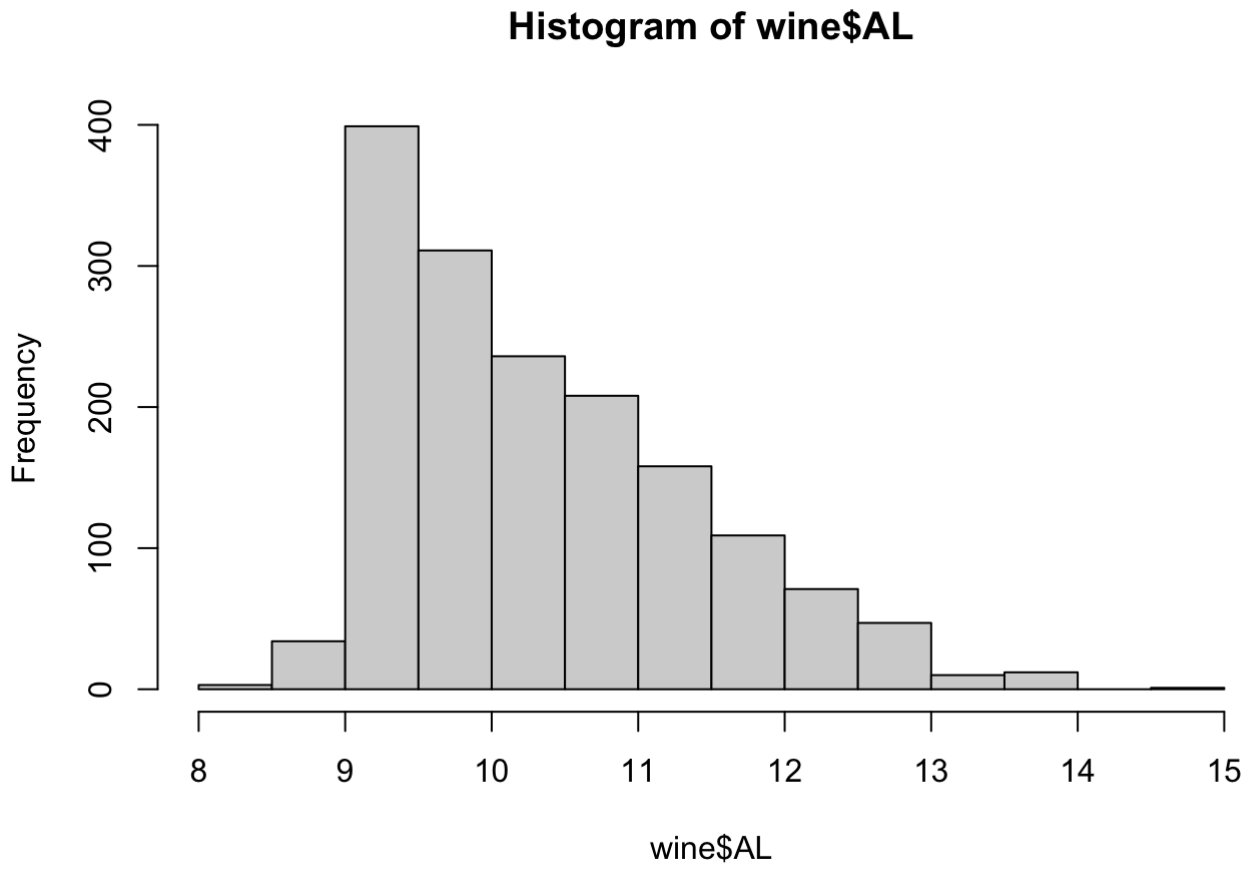
```
hist(wine$PH)
```




```
hist(wine$SU)
```

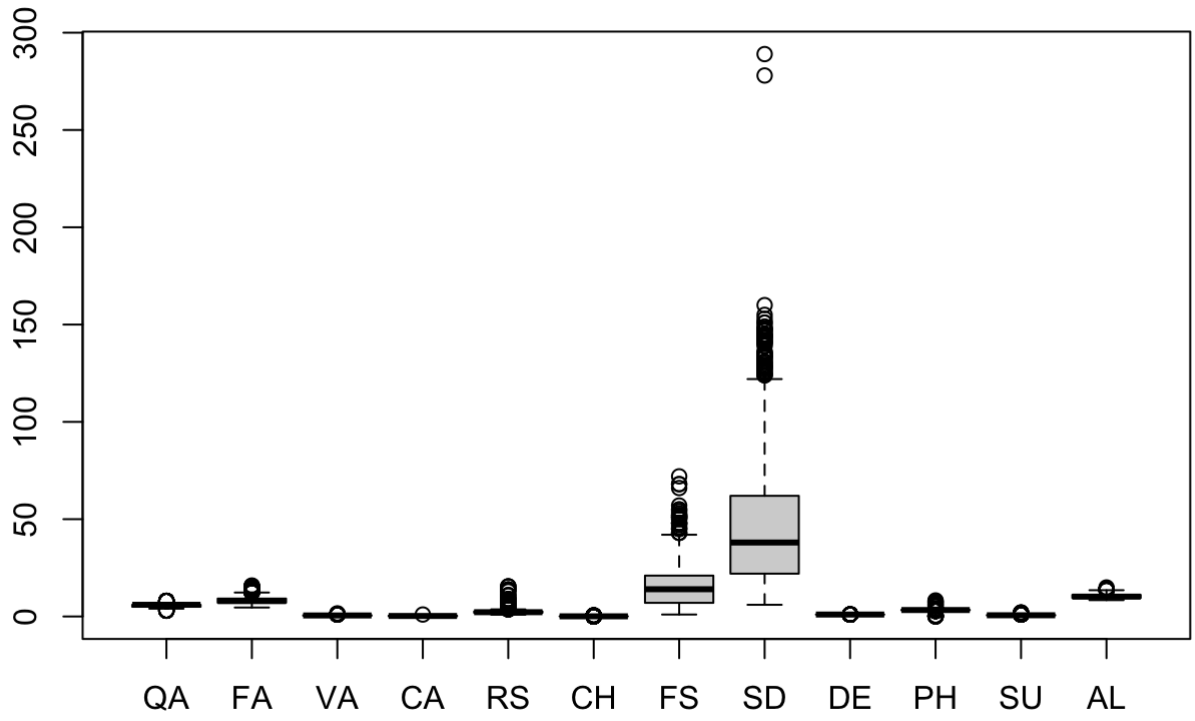

Histogram of wine$SU
Frequency
0
100
200
300
400
500
0.5
1.0
1.5
2.0
wine$SU

```
hist(wine$AL)
```

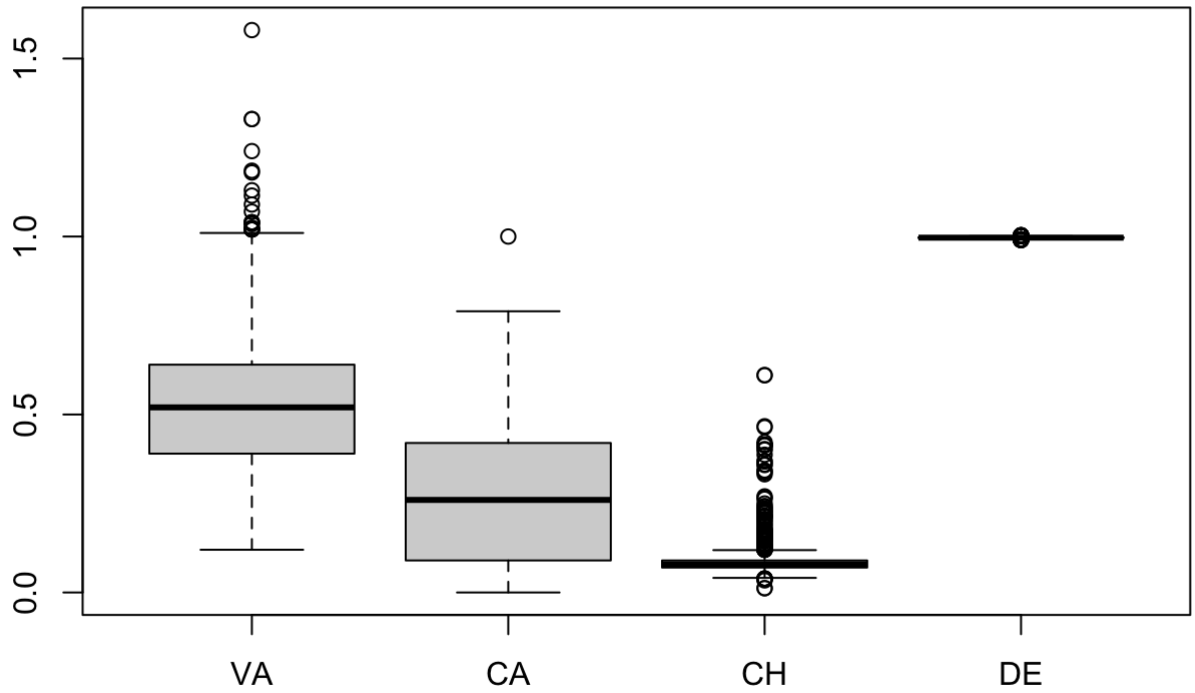

Histogram of wine$AL
Frequency
0
100
200
300
400
8
9
10
11
12
13
14
15
wine$AL

Problem 1b) First of all, we plot a overall box plot for all variables, looking for similar variables in terms of the data distribution.

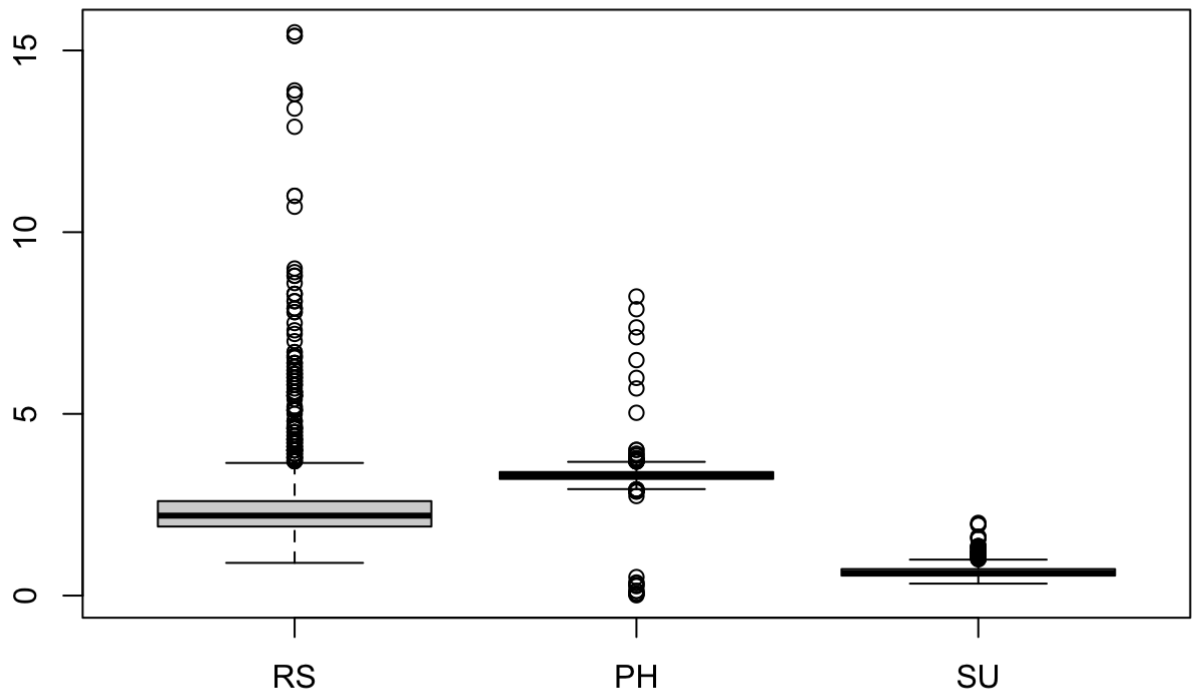
```
boxplot(wine)
```



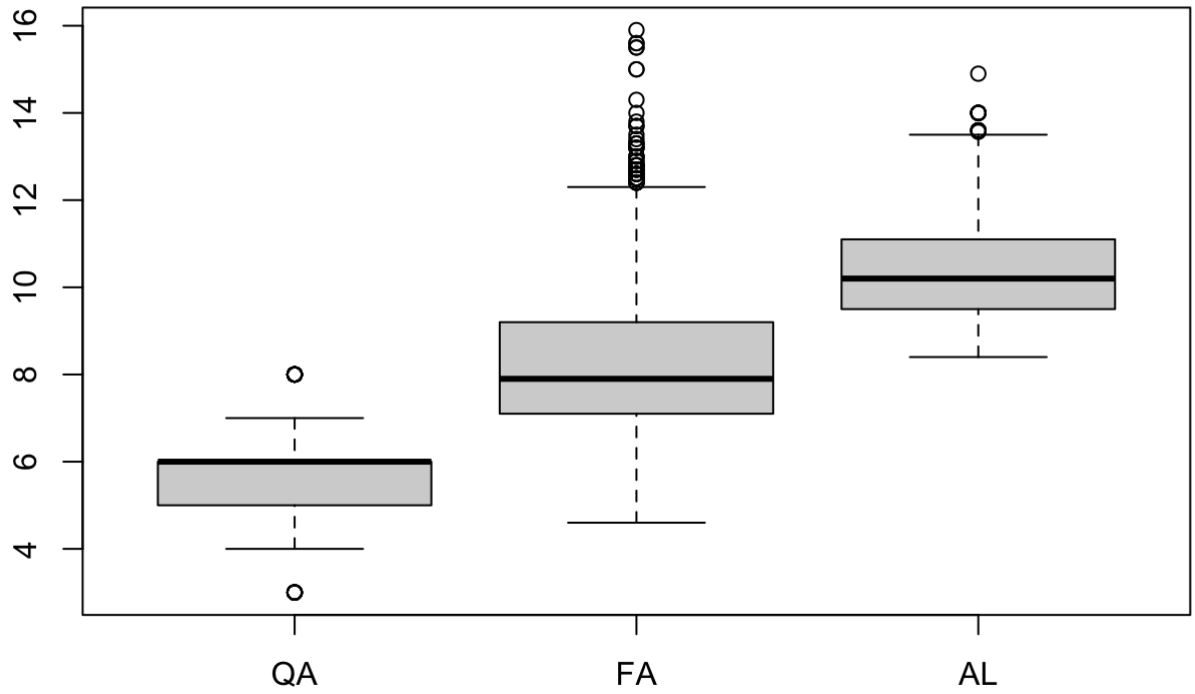

```
boxplot(wine %>% select(3,4,6,9))
```



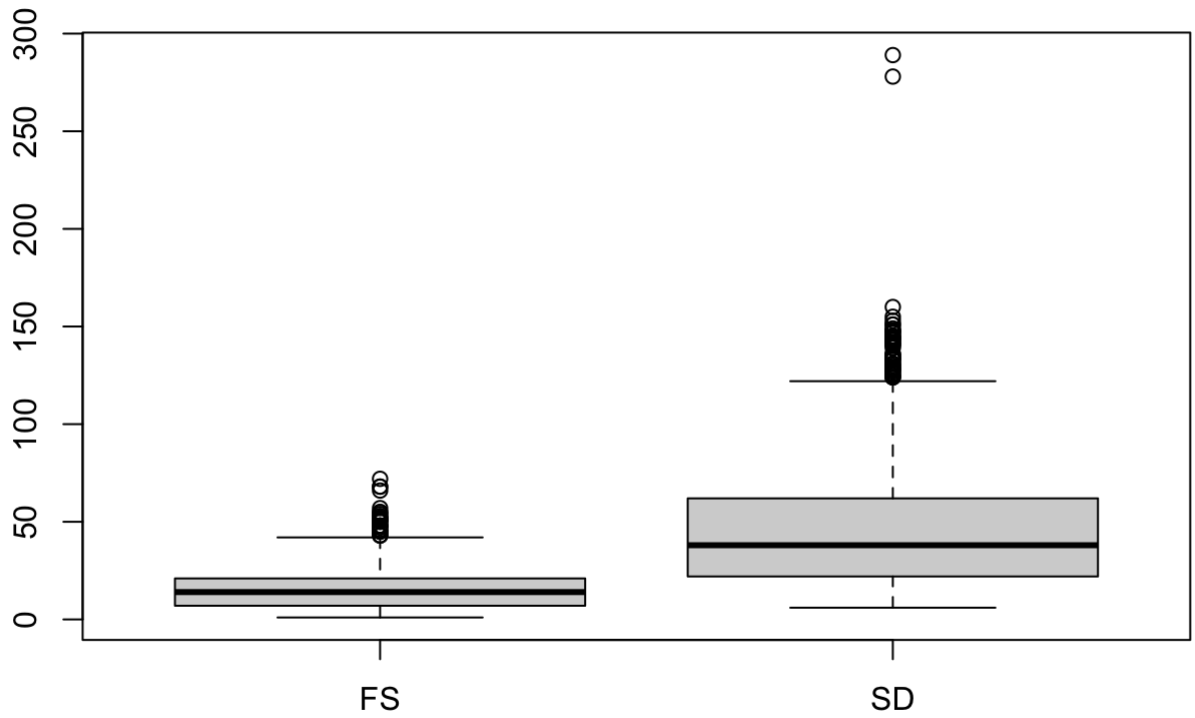

```
boxplot(wine %>% select(5,10,11))
```



```
boxplot(wine %>% select(1,2,12))
```




```
boxplot(wine %>% select(7,8))
```



The various box plots indicates that there are significant outliers among the variables.

Problem 1c)

```
library(e1071)
```

```
skewness(wine$QA)
```

```
## [1] 0.2173931
```

```
kurtosis(wine$QA)
```

```
## [1] 0.2879148
```

```
skewness(wine$FA)
```

```
## [1] 0.9809084
```

```
kurtosis(wine$FA)
```

```
## [1] 1.119699
```

```
skewness(wine$VA)
```

```
## [1] 0.6703331
```

```
kurtosis(wine$VA)
```

```
## [1] 1.212689
```

```
skewness(wine$CA)
```

```
## [1] 0.3177403
```

```
kurtosis(wine$CA)
```

```
## [1] -0.7930455
```

```
skewness(wine$RS,na.rm = TRUE)
```

```
## [1] 4.536234
```

```
kurtosis(wine$RS,na.rm = TRUE)
```

```
## [1] 28.41558
```

```
skewness(wine$CH)
```

```
## [1] 5.669694
```

```
kurtosis(wine$CH)
```

```
## [1] 41.52596
```

```
skewness(wine$FS)
```

```
## [1] 1.248222
```

```
kurtosis(wine$FS)
```

```
## [1] 2.007221
```

```
skewness(wine$SD,na.rm = TRUE)
```

```
## [1] 1.510903
```

```
kurtosis(wine$SD,na.rm = TRUE)
```

```
## [1] 3.851013
```

```
skewness(wine$DE)
```

```
## [1] 0.07115397
```

```
kurtosis(wine$DE)
```

```
## [1] 0.9225
```

```
skewness(wine$PH)
```

```
## [1] 0.7893898
```

```
kurtosis(wine$PH)
```

```
## [1] 69.51551
```

```
skewness(wine$SU)
```

```
## [1] 2.424118
```

```
kurtosis(wine$SU)
```

```
## [1] 11.66153
```

```
skewness(wine$AL)
```

```
## [1] 0.8592144
```

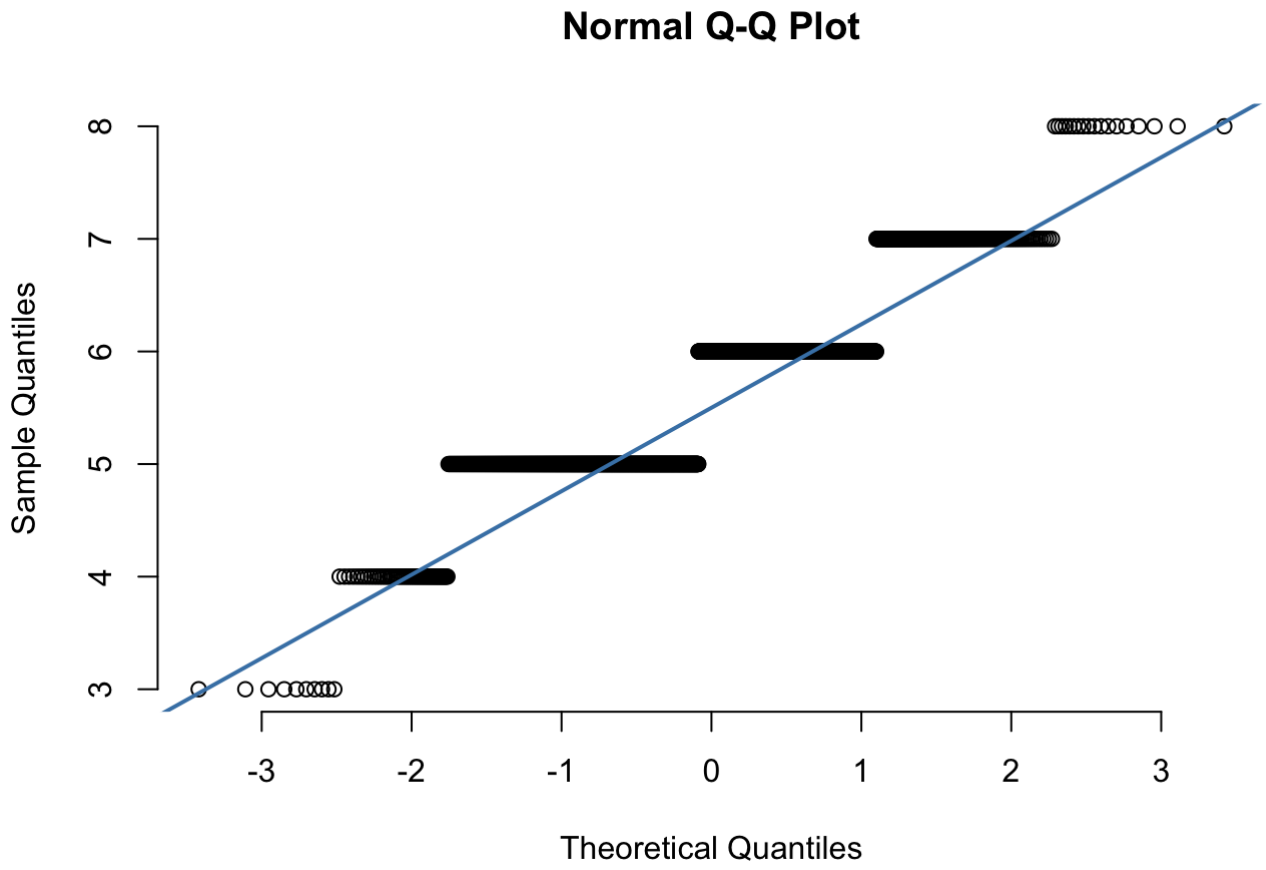
```
kurtosis(wine$AL)
```

```
## [1] 0.1916586
```

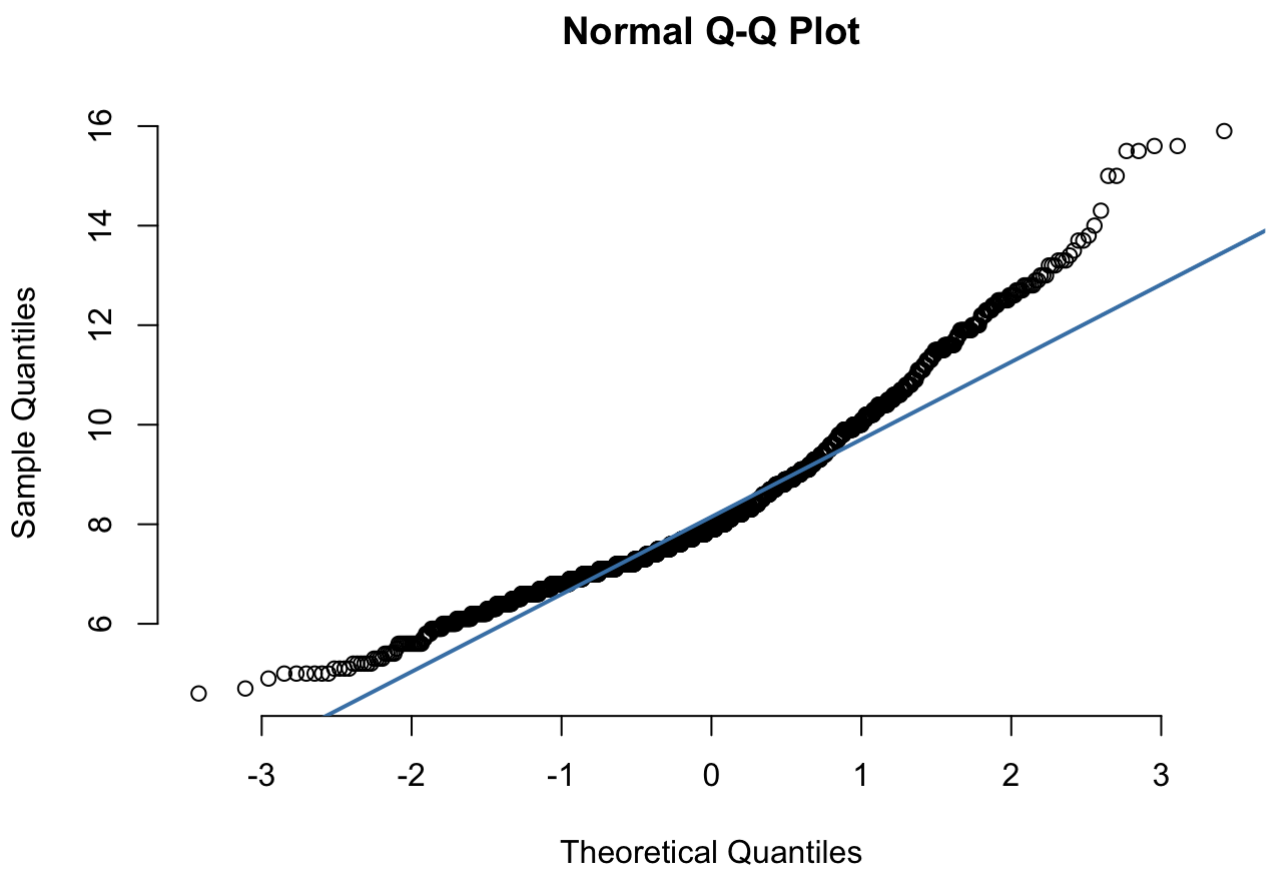
From the skewness we can tell no variables were left skewed. SU, SD, FS, CH, RS, were significantly right skewed while the others were slightly right skewed. From the kurtosis, SD, FA, VA, FS were around 3 and therefore likely mesokurtic. SU, PH, CH, RS were significantly greater than 3, so leptokurtic. AL, DE, CA, QA were significantly less than 3 so platykurtic.

1d) We can draw the QQ plot with the following code

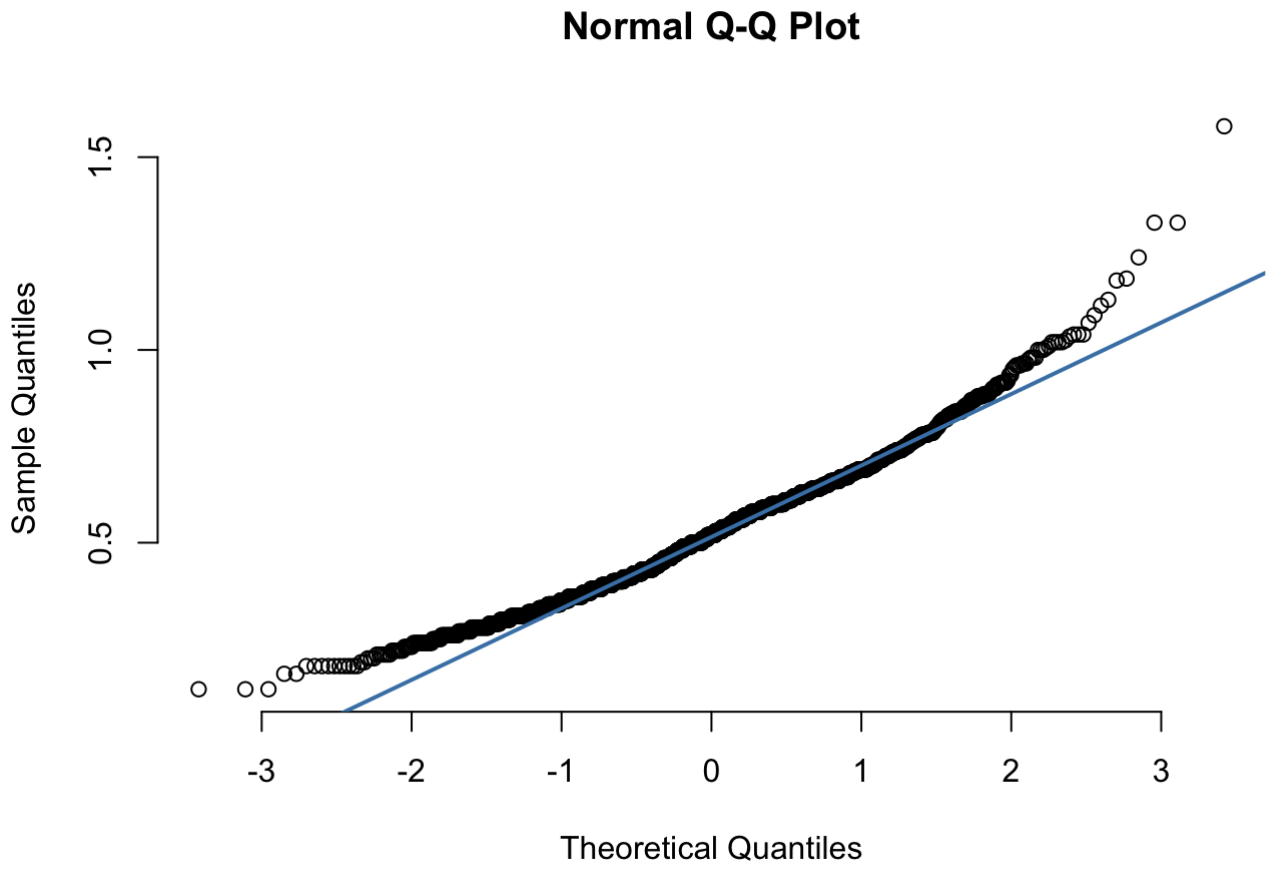
```
qqnorm(wine$QA, pch = 1, frame = FALSE)
qqline(wine$QA, col = "steelblue", lwd = 2)
```



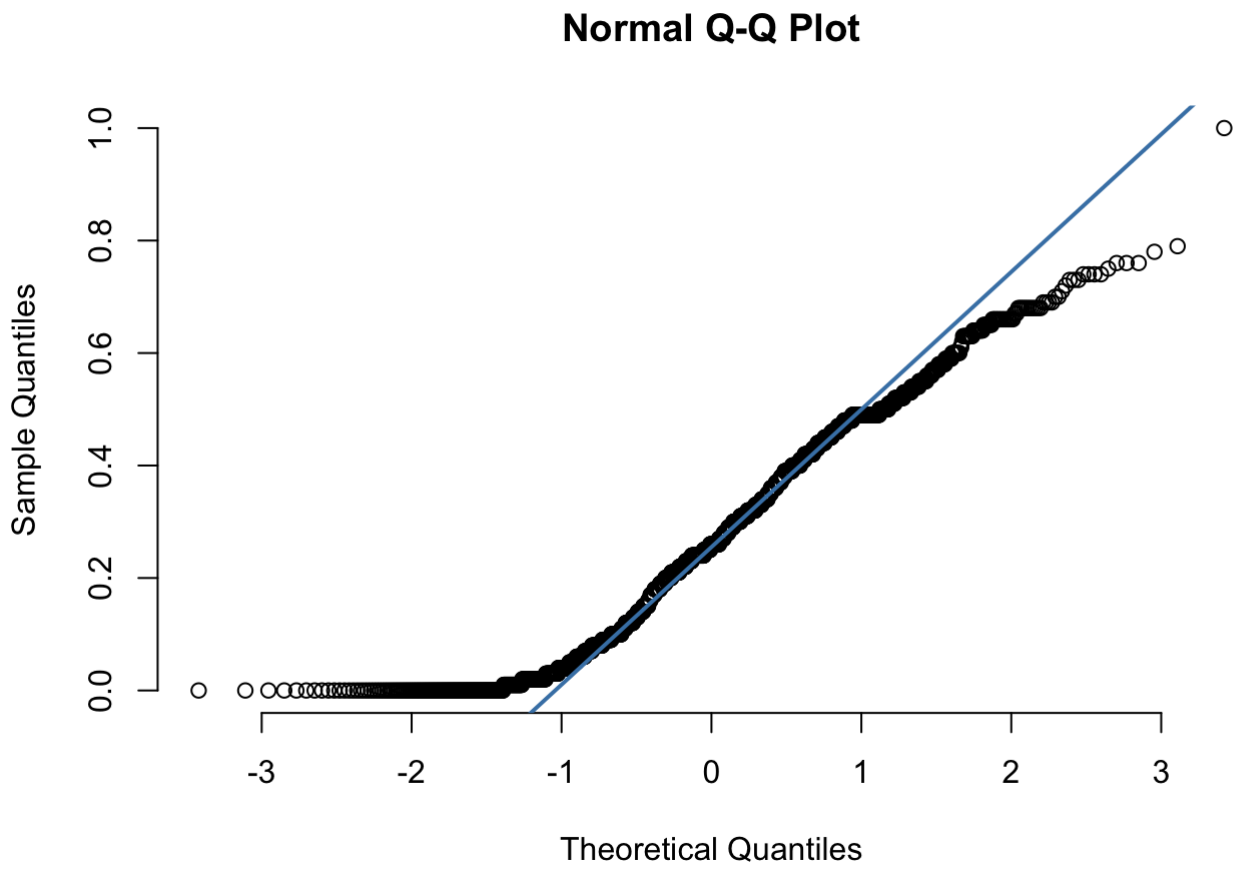

```
qqnorm(wine$FA, pch = 1, frame = FALSE)
qqline(wine$FA, col = "steelblue", lwd = 2)
```



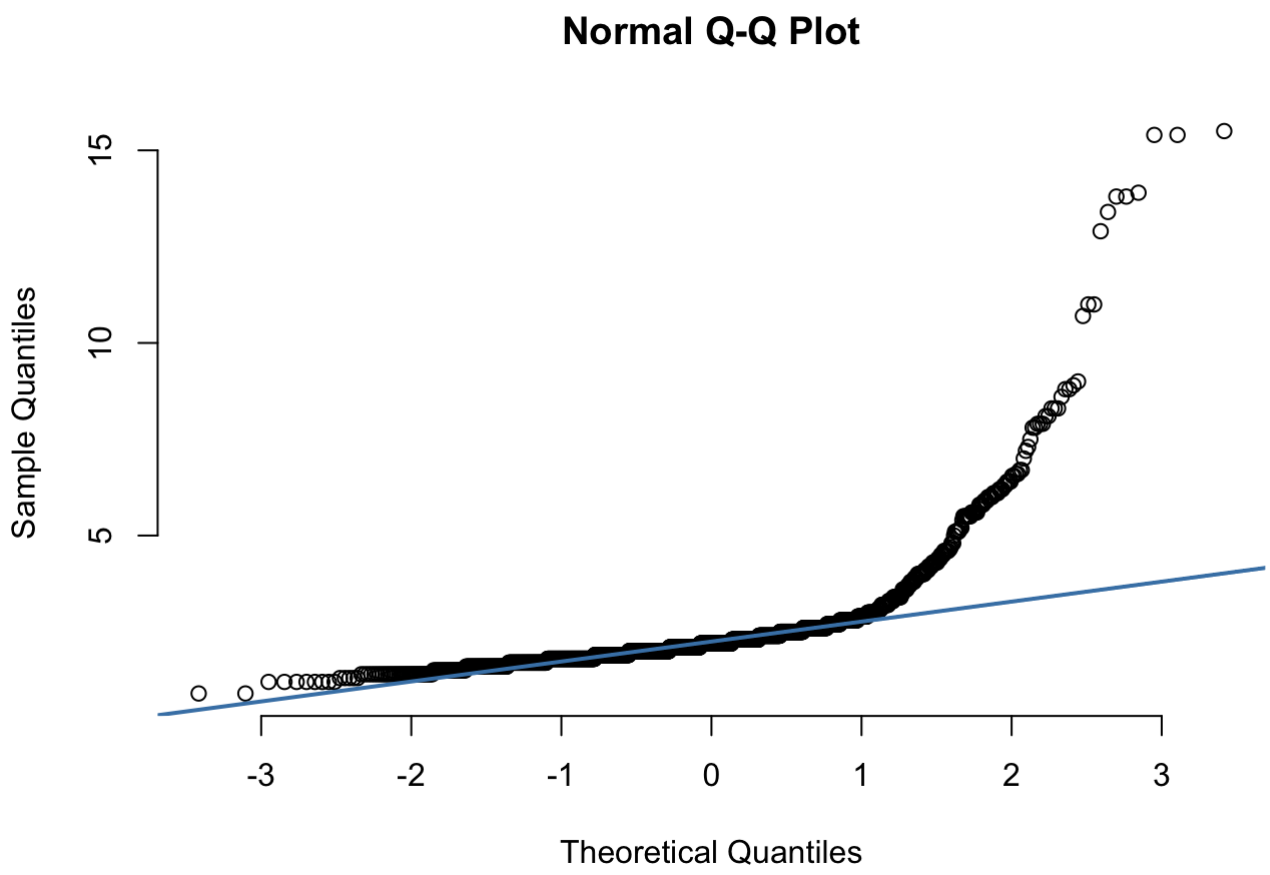

```
qqnorm(wine$VA, pch = 1, frame = FALSE)
qqline(wine$VA, col = "steelblue", lwd = 2)
```



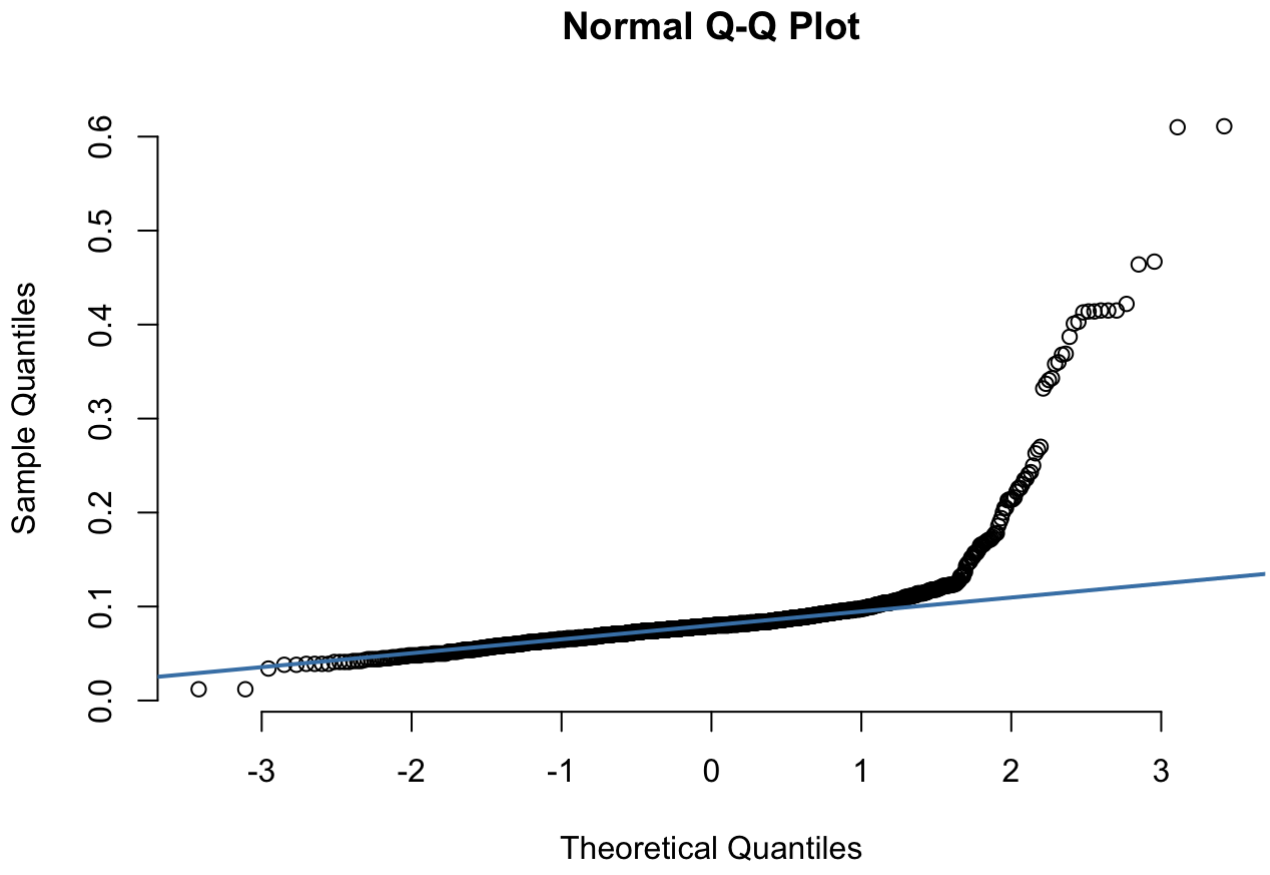
```
qqnorm(wine$CA, pch = 1, frame = FALSE)
qqline(wine$CA, col = "steelblue", lwd = 2)
```



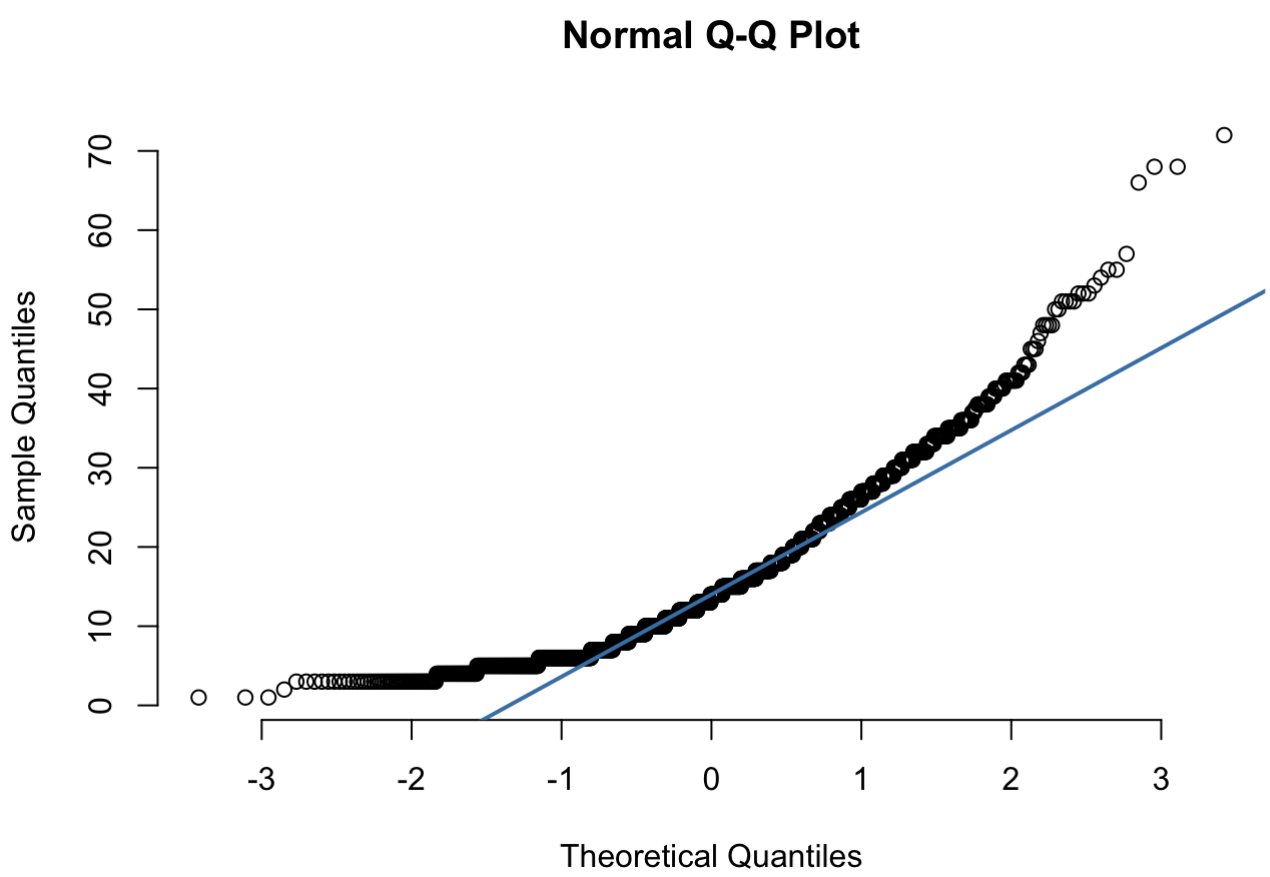

```
qqnorm(wine$RS, pch = 1, frame = FALSE)
qqline(wine$RS, col = "steelblue", lwd = 2)
```



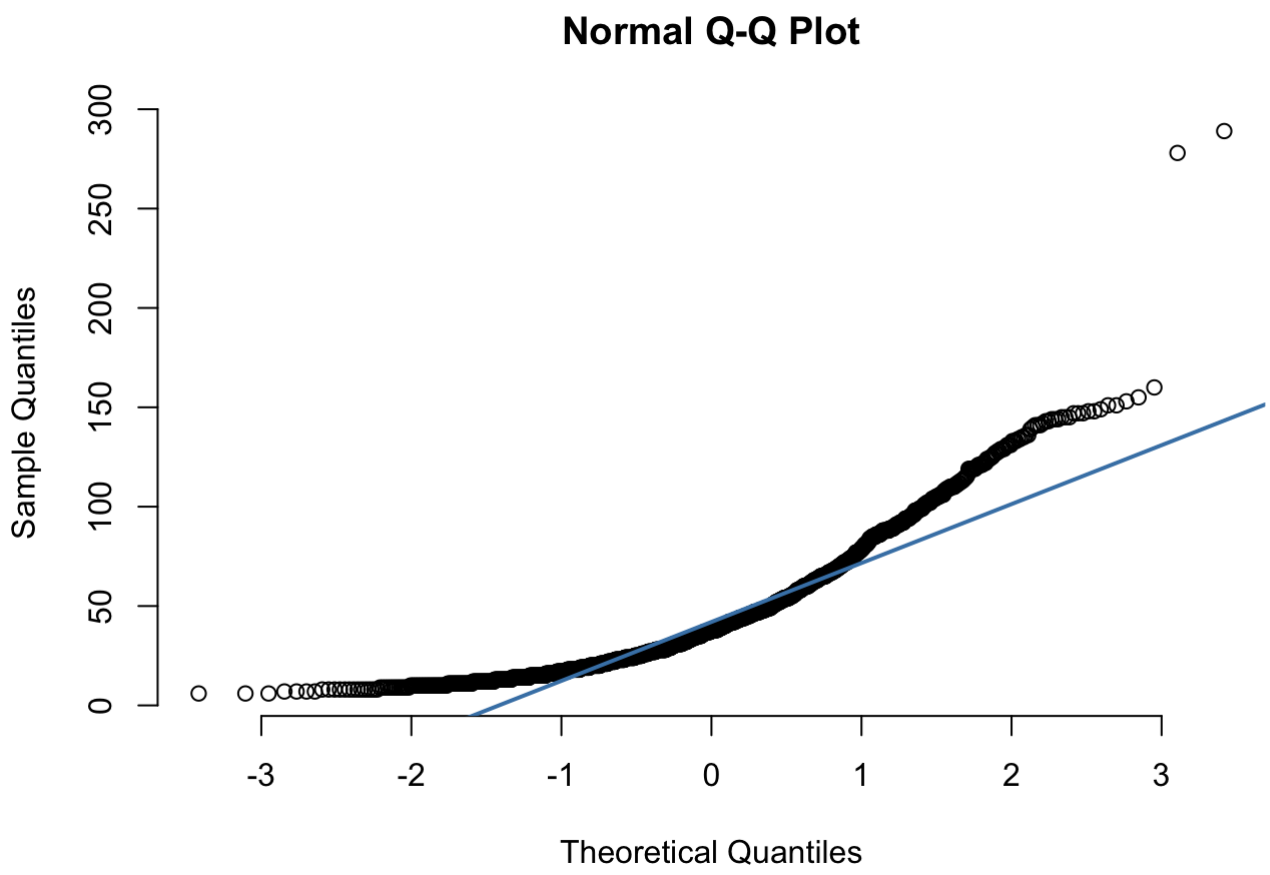

```
qqnorm(wine$CH, pch = 1, frame = FALSE)
qqline(wine$CH, col = "steelblue", lwd = 2)
```



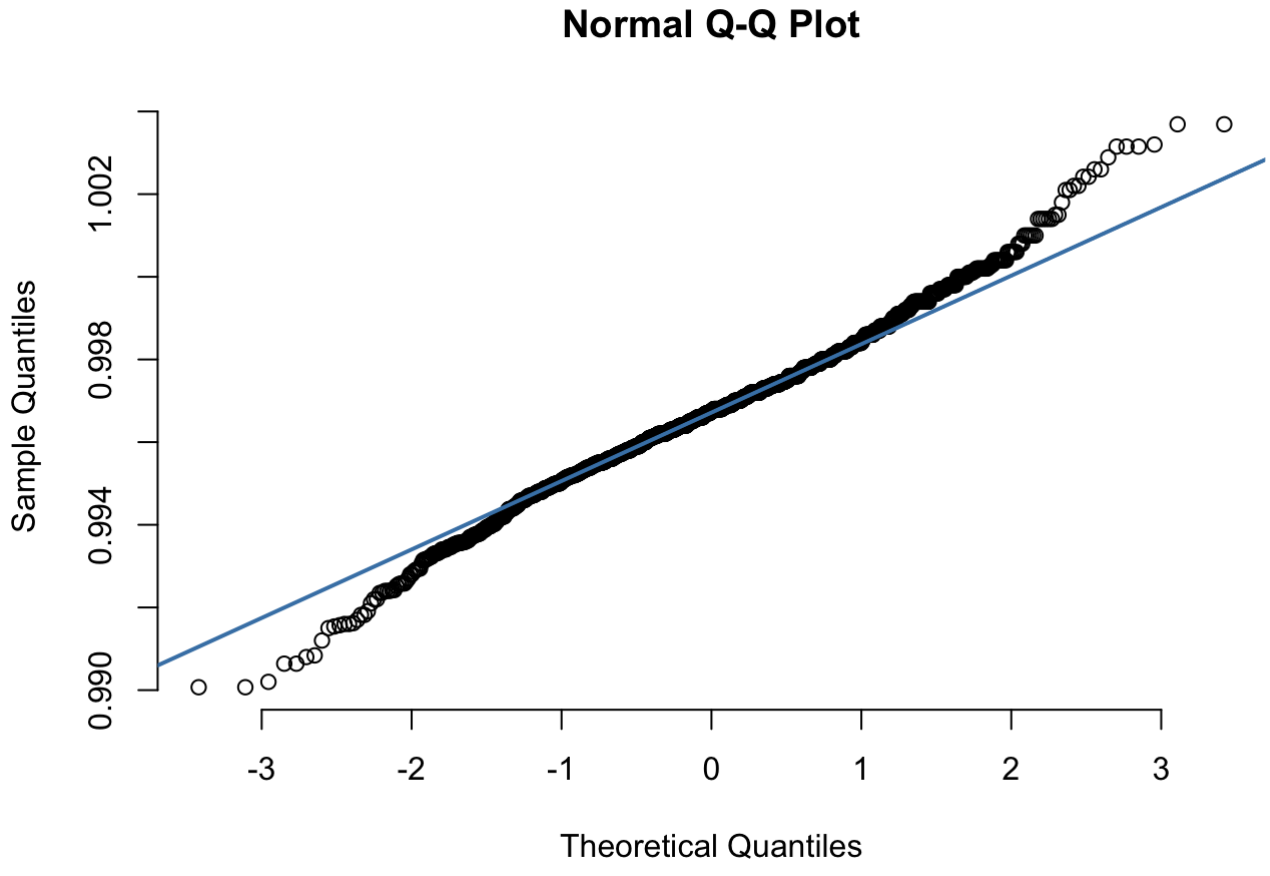
```
qqnorm(wine$FS, pch = 1, frame = FALSE)
qqline(wine$FS, col = "steelblue", lwd = 2)
```



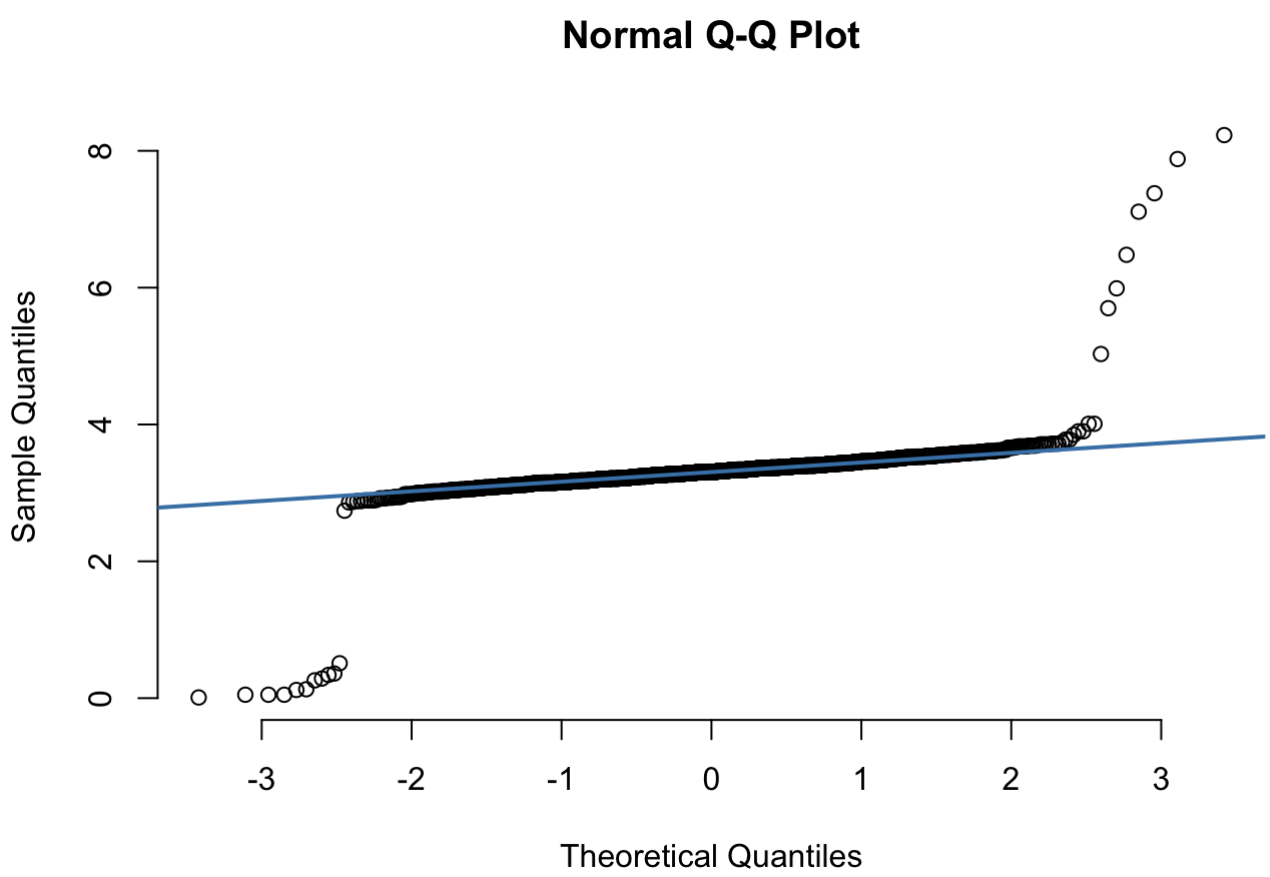

```
qqnorm(wine$SD, pch = 1, frame = FALSE)
qqline(wine$SD, col = "steelblue", lwd = 2)
```



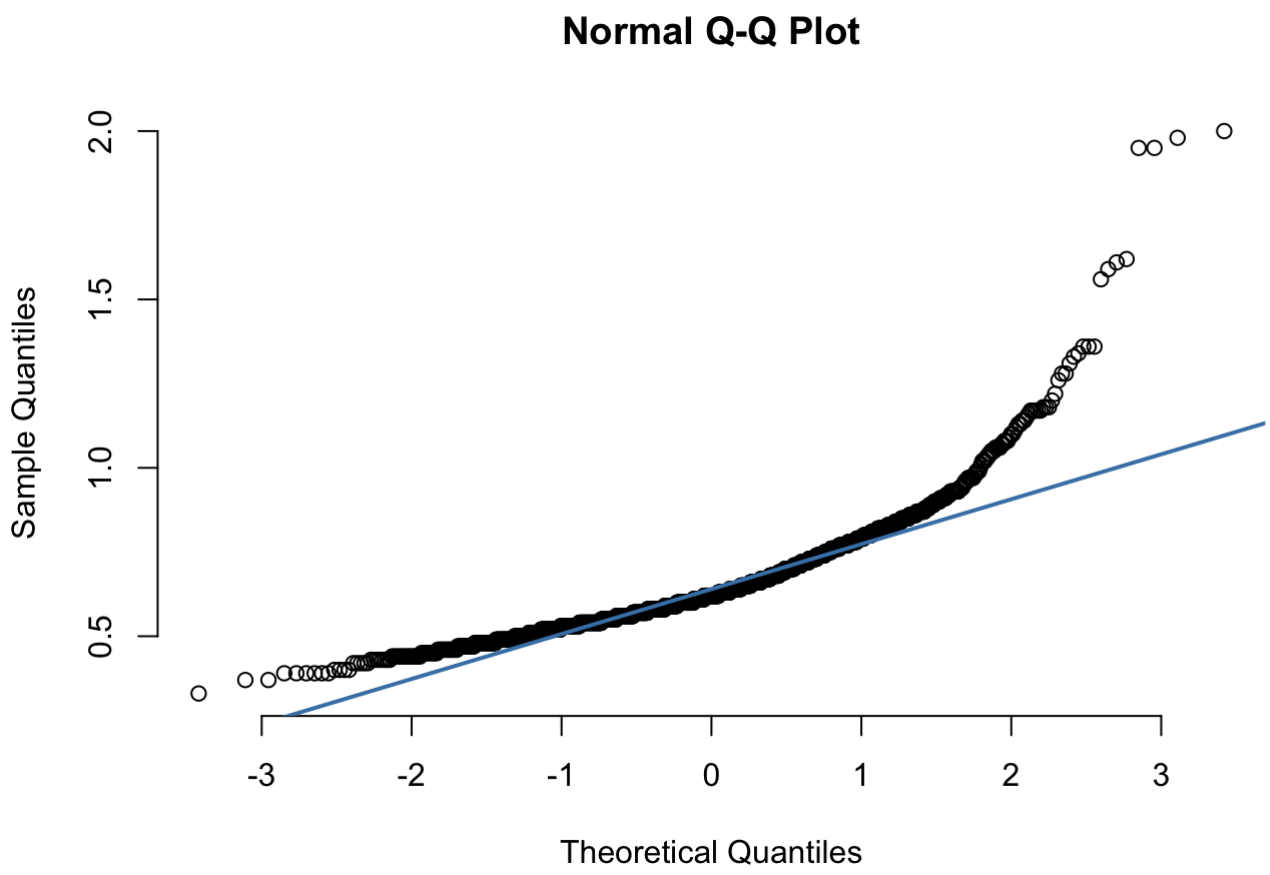

```
qqnorm(wine$DE, pch = 1, frame = FALSE)
qqline(wine$DE, col = "steelblue", lwd = 2)
```



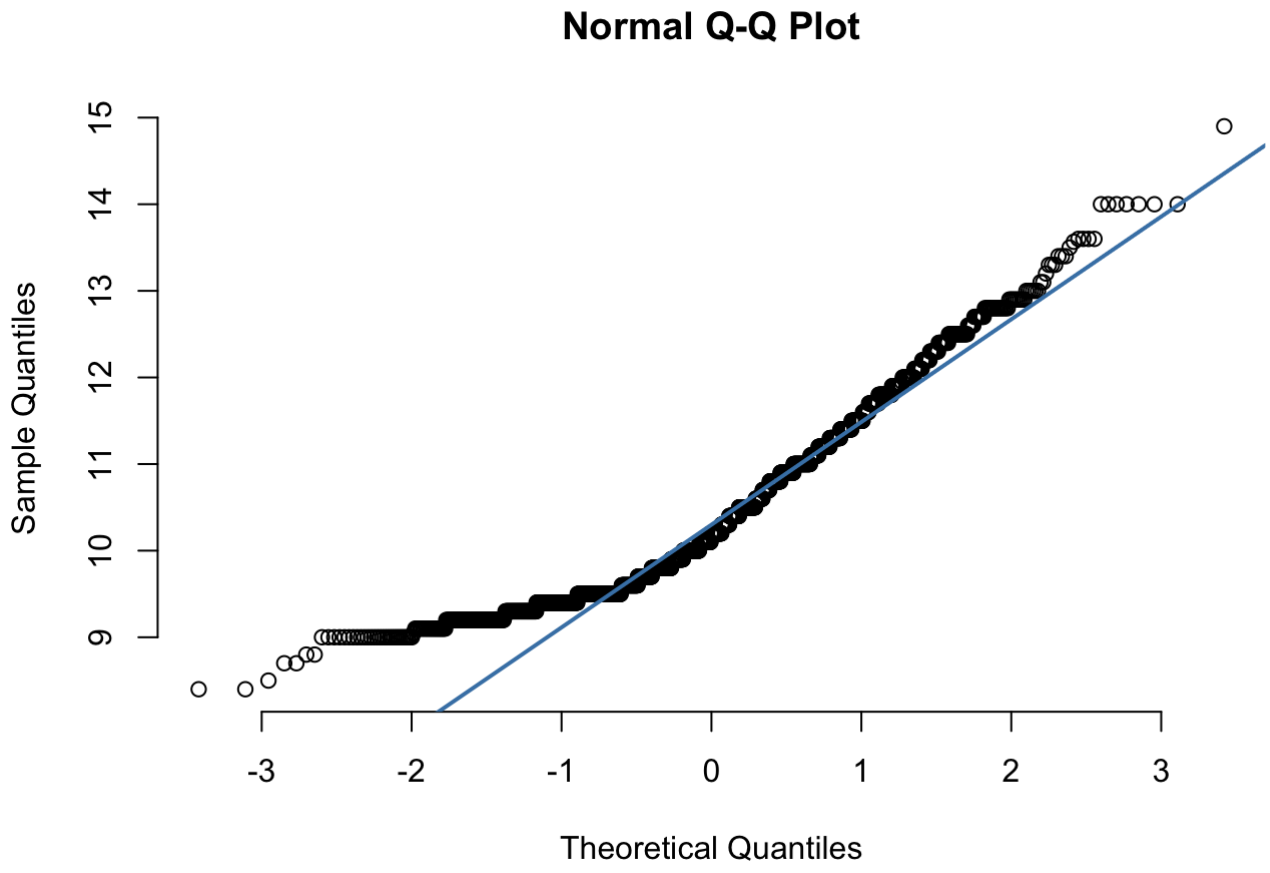
```
qqnorm(wine$PH, pch = 1, frame = FALSE)
qqline(wine$PH, col = "steelblue", lwd = 2)
```




```
qqnorm(wine$SU, pch = 1, frame = FALSE)
qqline(wine$SU, col = "steelblue", lwd = 2)
```




```
qqnorm(wine$AL, pch = 1, frame = FALSE)
qqline(wine$AL, col = "steelblue", lwd = 2)
```



The QQ plots confirm observations above, especially in terms of skew showing that none are left skewed. Additionally, the type of kurtosis is apparent for each variable in the given QQ plots.

Problem 2

2a) The following code can be used to check the missing values of the datase

```
colSums(is.na(wine))
```

```
## QA FA VA CA RS CH FS SD DE PH SU AL
##  0  0  0  0 22  0  0 17  0  0  0  0
```

```
#rowSums(is.na(wine))
sum(rowSums(is.na(wine)))
```

```
## [1] 39
```

```
length(rowSums(is.na(wine)))
```

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

2b)

Here we divide them into 5 folds.

```
set.seed(314159)
n = nrow(wine) # number of samples
```

```
library(caret)
```

```
nfolds = 5
folds = createFolds(1:n, k=nfolds)
```

Random Sampling

```
tMSE <- vector()
vMSE <- vector()
for (i in 1:nfolds){
      train = wine[-folds[[i]],]
      validation = wine[folds[[i]],]

      sampRS <- train$RS[!is.na(train$RS)]
      sampSD <- train$SD[!is.na(train$SD)]
      train$RS[is.na(train$RS)] <- sample(sampRS,length(train$RS[is.na(train$RS)]),re
place = TRUE)
      validation$RS[is.na(validation$RS)] <- sample(sampRS,length(validation$RS[is.na
(validation$RS)]),replace = TRUE)

      train$SD[is.na(train$SD)] <- sample(sampSD,length(train$SD[is.na(train$SD)]),re
place = TRUE)
      validation$SD[is.na(validation$SD)] <- sample(sampSD,length(validation$SD[is.na
(validation$SD)]),replace = TRUE)

      model <- lm(QA~., data = train)
      trainpredict = predict(model)
      valpredict = predict(model,newdata=validation)

      trainMSE <- mean((train$QA - trainpredict)^2)
      valMSE <- mean((validation$QA - valpredict)^2)

      tMSE <- append(tMSE,trainMSE)
      vMSE <- append(vMSE,valMSE)


}
print("Mean Training MSE")
```

```
## [1] "Mean Training MSE"
```

```
print(mean(tMSE))
```

```
## [1] 0.4172222
```

```
print("Mean Validation MSE")
```

```
## [1] "Mean Validation MSE"
```

```
print(mean(vMSE))
```

```
## [1] 0.4257731
```

2c) Most common value

```
tMSE <- vector()
vMSE <- vector()
for (i in 1:nfolds){
      train = wine[-folds[[i]],]
      validation = wine[folds[[i]],]

      sampRS <- train$RS[!is.na(train$RS)]
      sampSD <- train$SD[!is.na(train$SD)]

      cRS <- table(sampRS)
      RScom <- as.numeric(names(cRS)[which(cRS==max(cRS))])

      cSD <- table(sampSD)
      SDcom <- as.numeric(names(cSD)[which(cSD==max(cSD))])

      train$RS[is.na(train$RS)] <- RScom
      validation$RS[is.na(validation$RS)] <- RScom



      train$SD[is.na(train$SD)] <- SDcom
      validation$SD[is.na(validation$SD)] <- SDcom

      model <- lm(QA~., data = train)
      trainpredict = predict(model)
      valpredict = predict(model,newdata=validation)

      trainMSE <- mean((train$QA - trainpredict)^2)
      valMSE <- mean((validation$QA - valpredict)^2)

      tMSE <- append(tMSE,trainMSE)
      vMSE <- append(vMSE,valMSE)

      print(length(validation$QA))
      print(length(valpredict))


}
```

```
## [1] 320
## [1] 320
## [1] 320
## [1] 320
## [1] 320
## [1] 320
## [1] 319
## [1] 319
## [1] 320
## [1] 320
```

```
print("Mean Training MSE")
```

```
## [1] "Mean Training MSE"
```

```
print(mean(tMSE))
```

```
## [1] 0.4171998
```

```
print("Mean Validation MSE")
```

```
## [1] "Mean Validation MSE"
```

```
print(mean(vMSE))
```

```
## [1] 0.4252296
```

2d) Average Value

```
tMSE <- vector()
vMSE <- vector()
for (i in 1:nfolds){
      train = wine[-folds[[i]],]
      validation = wine[folds[[i]],]

      sampRS <- train$RS[!is.na(train$RS)]
      sampSD <- train$SD[!is.na(train$SD)]

      meanRS <- mean(sampRS)
      meanSD <- mean(sampSD)

      train$RS[is.na(train$RS)] <- meanRS
      validation$RS[is.na(validation$RS)] <- meanRS

      train$SD[is.na(train$SD)] <- meanSD
      validation$SD[is.na(validation$SD)] <- meanSD

      model <- lm(QA~., data = train)

      trainpredict = predict(model)
      valpredict = predict(model,newdata=validation)

      trainMSE <- mean((train$QA - trainpredict)^2)
      valMSE <- mean((validation$QA - valpredict)^2)

      tMSE <- append(tMSE,trainMSE)
      vMSE <- append(vMSE,valMSE)


}
print("Mean Training MSE")
```

```
## [1] "Mean Training MSE"
```

```
print(mean(tMSE))
```

```
## [1] 0.4170898
```

```
print("Mean Validation MSE")
```

```
## [1] "Mean Validation MSE"
```

```
print(mean(vMSE))
```

```
## [1] 0.4250095
```

2e KNN

```
library(DMwR2)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
tMSE <- vector()
vMSE <- vector()
for (i in 1:nfolds){
      train = wine[-folds[[i]],]
      validation = wine[folds[[i]],]

      train <- knnImputation(train,k=5)
      validation <- knnImputation(validation,k=5)

      model <- lm(QA~., data = train)

      trainpredict = predict(model)
      valpredict = predict(model,newdata=validation)

      trainMSE <- mean((train$QA - trainpredict)^2)
      valMSE <- mean((validation$QA - valpredict)^2)

      tMSE <- append(tMSE,trainMSE)
      vMSE <- append(vMSE,valMSE)


}
print("Mean Training MSE")
```

```
## [1] "Mean Training MSE"
```

```
print(mean(tMSE))
```

```
## [1] 0.4170197
```

```
print("Mean Validation MSE")
```

```
## [1] "Mean Validation MSE"
```

```
print(mean(vMSE))
```

```
## [1] 0.4249027
```

2f) MICE

Here we can load mice to help us do the data imputation

```
library(mice)
```

```
##
## Attaching package: 'mice'
```

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

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
tMSE <- vector()
vMSE <- vector()
for (i in 1:nfolds){
      #Ignore rows for the test set
      ignore = ifelse(1:nrow(wine) %in% folds[[i]], TRUE, FALSE)
      wineCopy <- wine
      wineCopy <- complete(mice(wineCopy, ignore=ignore,seed=12345, print=F,meth='pm
m'))
      train = wineCopy[-folds[[i]],]
      validation = wineCopy[folds[[i]],]

      model <- lm(QA~., data = train)

      trainpredict = predict(model)
      valpredict = predict(model,newdata=validation)

      trainMSE <- mean((train$QA - trainpredict)^2)
      valMSE <- mean((validation$QA - valpredict)^2)



      tMSE <- append(tMSE,trainMSE)
      vMSE <- append(vMSE,valMSE)


}
```

```
## Warning: Number of logged events: 50

## Warning: Number of logged events: 50

## Warning: Number of logged events: 50

## Warning: Number of logged events: 50

## Warning: Number of logged events: 50
```

```
print("Mean Training MSE")
```

```
## [1] "Mean Training MSE"
```

```
print(mean(tMSE))
```

```
## [1] 0.4173353
```

```
print("Mean Validation MSE")
```

```
## [1] "Mean Validation MSE"
```

```
print(mean(vMSE))
```

```
## [1] 0.4243435
```

2g) Remove NA

We use the above methods to remove the NA values by imputation

```
tMSE <- vector()
vMSE <- vector()
for (i in 1:nfolds){
      train = wine[-folds[[i]],]
      validation = wine[folds[[i]],]

      train <- na.omit(train)
      validation <- na.omit(validation)

      model <- lm(QA~., data = train)

      trainpredict = predict(model)
      valpredict = predict(model,newdata=validation)

      trainMSE <- mean((train$QA - trainpredict)^2)
      valMSE <- mean((validation$QA - valpredict)^2)

      tMSE <- append(tMSE,trainMSE)
      vMSE <- append(vMSE,valMSE)


}
print("Mean Training MSE")
```

```
## [1] "Mean Training MSE"
```

```
print(mean(tMSE))
```

```
## [1] 0.4199708
```

```
print("Mean Validation MSE")
```

```
## [1] "Mean Validation MSE"
```

```
print(mean(vMSE))
```

```
## [1] 0.4277329
```

2h) Which method for handling missing values performs best? Why may this be?

Removing the NA's performs the best. It looks like data imputation does not offer us enough insights. This may be because of the fact that only a small percentage of rows had to be ommitted and the model could be trained on the remaining true values of the dataset.