

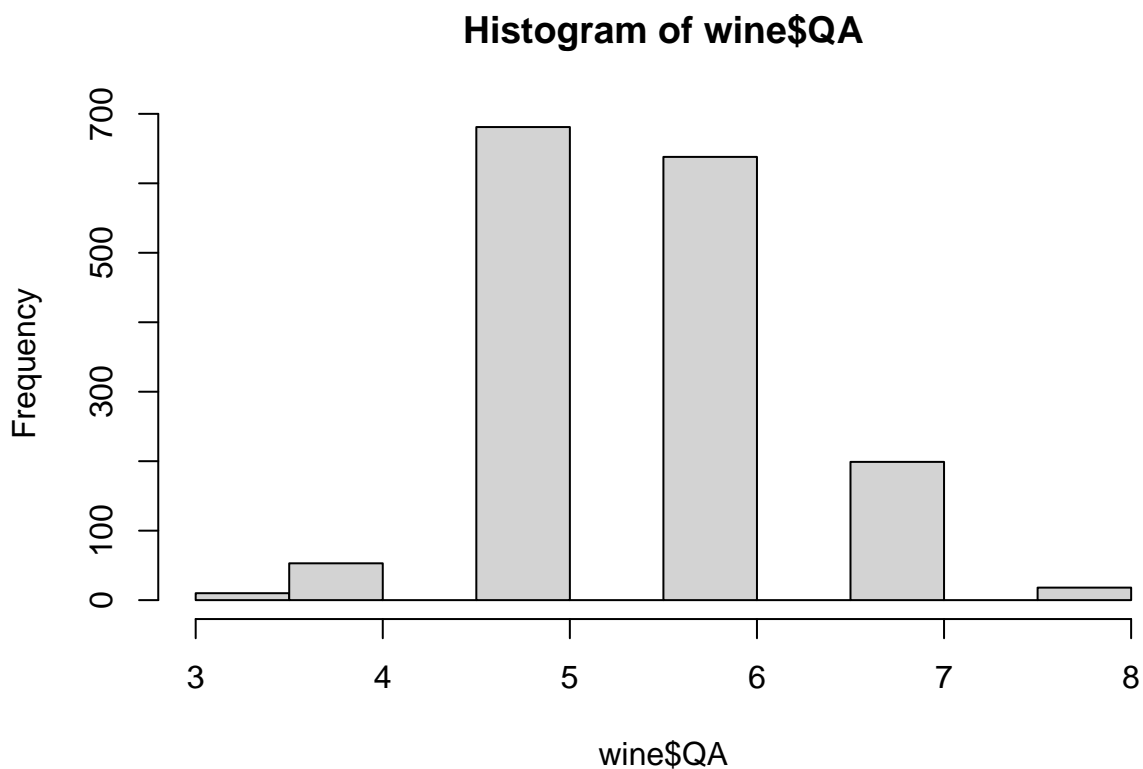
MSiA 400 Lab Assignment 4

Matt Ko: mjk3551

Problem 1a) For each of the 12 variables, plot the distribution using either a histogram or stem-and-leaf plot, whichever you deem more appropriate (for each variable).

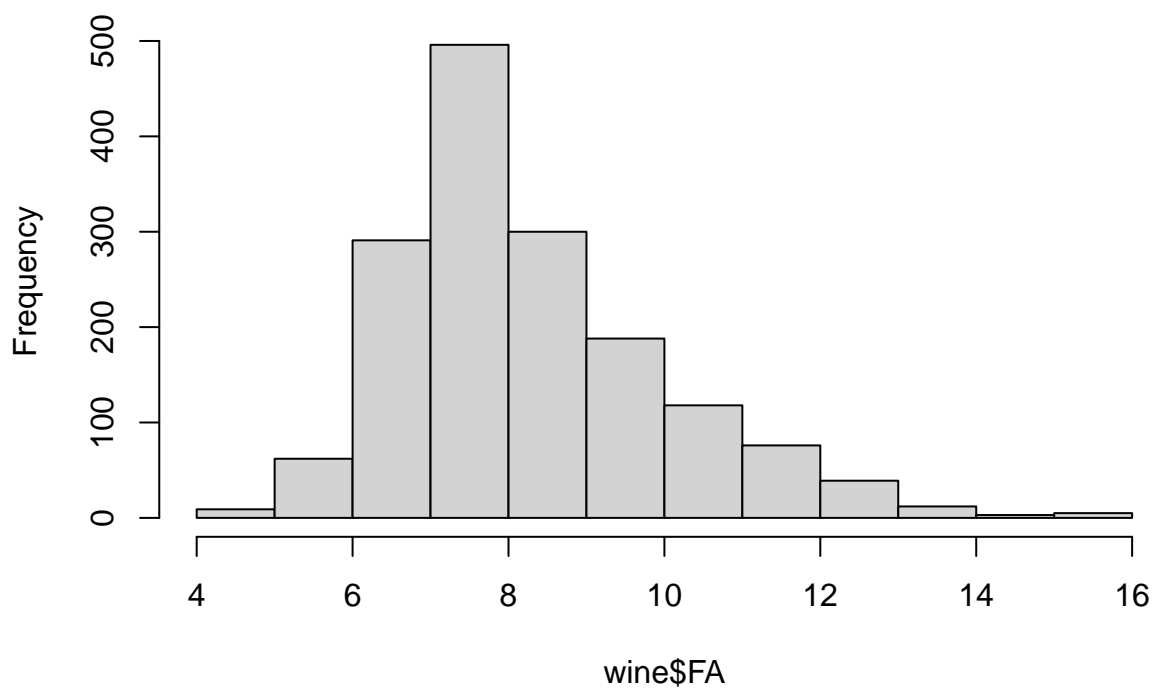
```
wine = read.delim("redwine.txt")
```

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



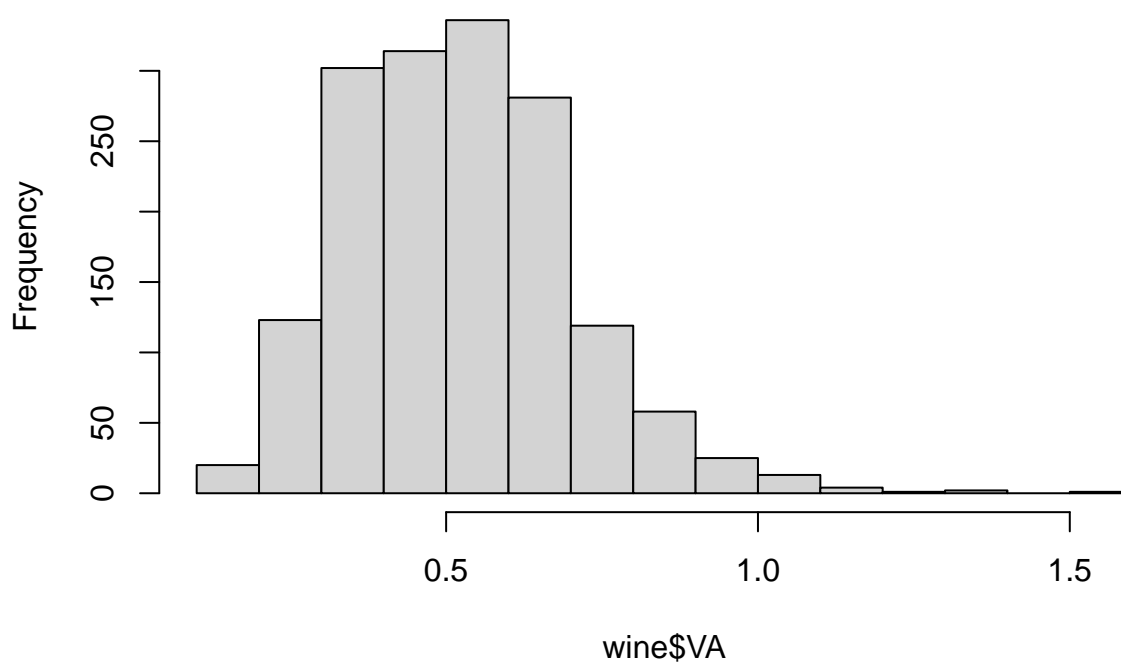
```
hist(wine$FA)
```

Histogram of wine\$FA



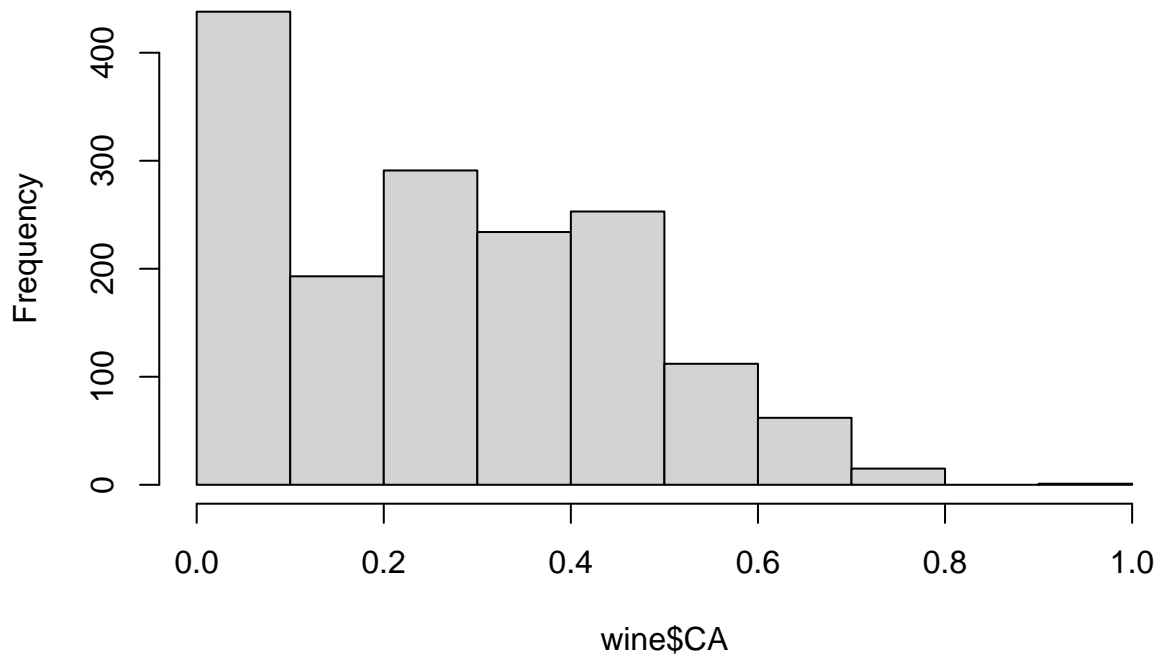
```
hist(wine$VA)
```

Histogram of wine\$VA



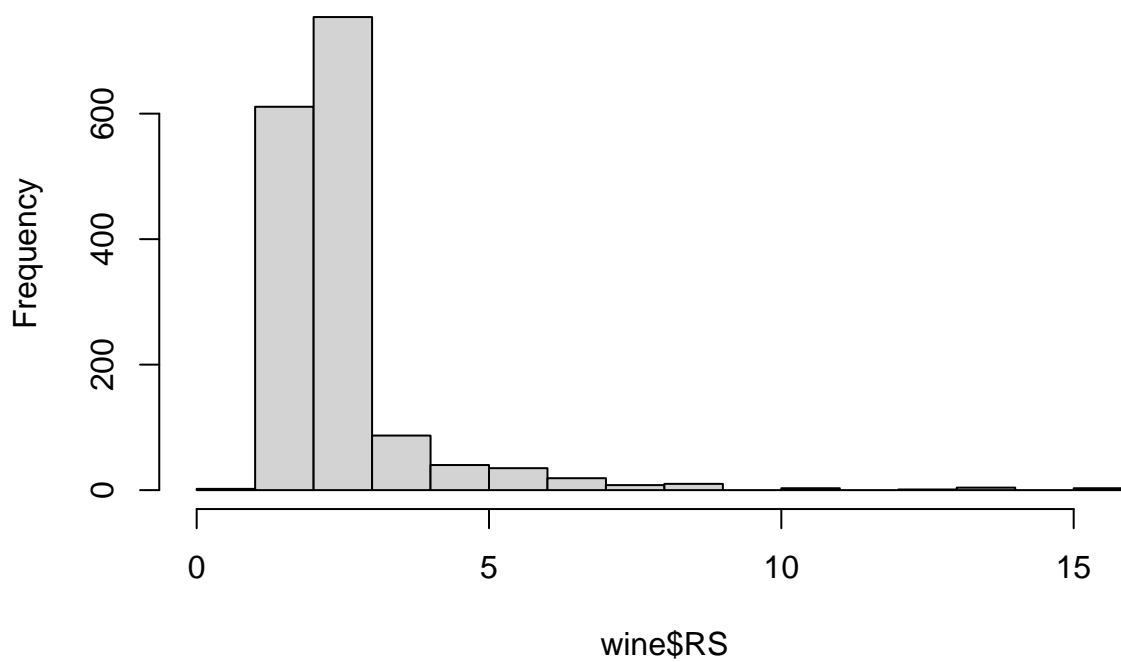
```
hist(wine$CA)
```

Histogram of wine\$CA



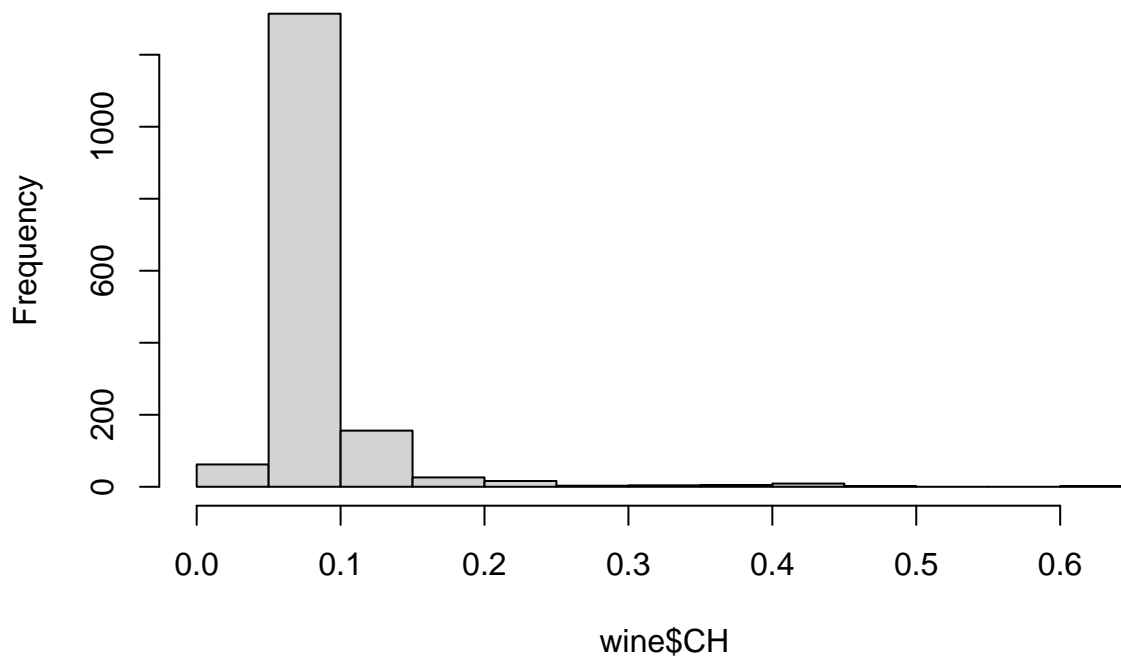
```
hist(wine$RS)
```

Histogram of wine\$RS



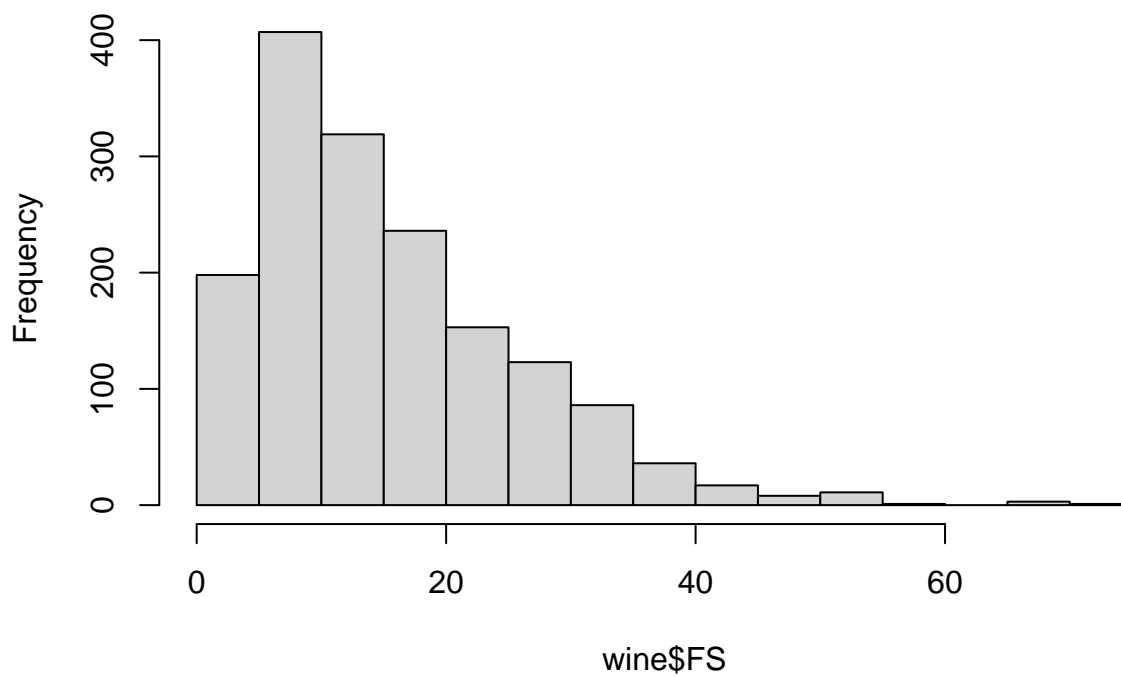
```
hist(wine$CH)
```

Histogram of wine\$CH



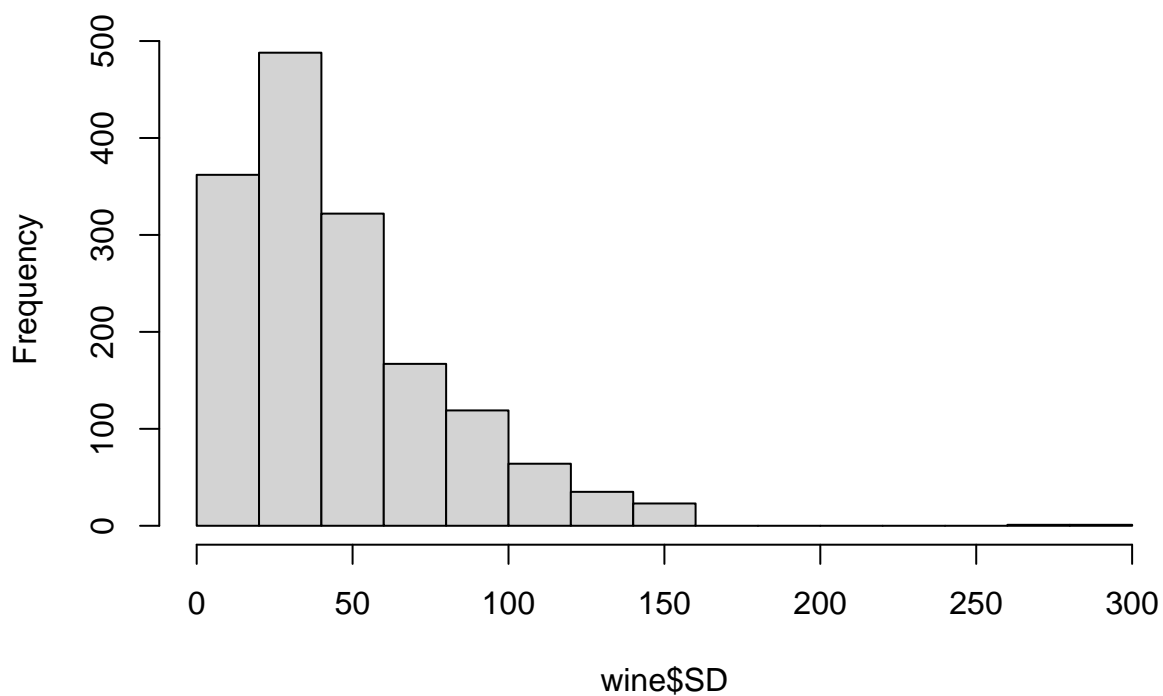
```
hist(wine$FS)
```

Histogram of wine\$FS



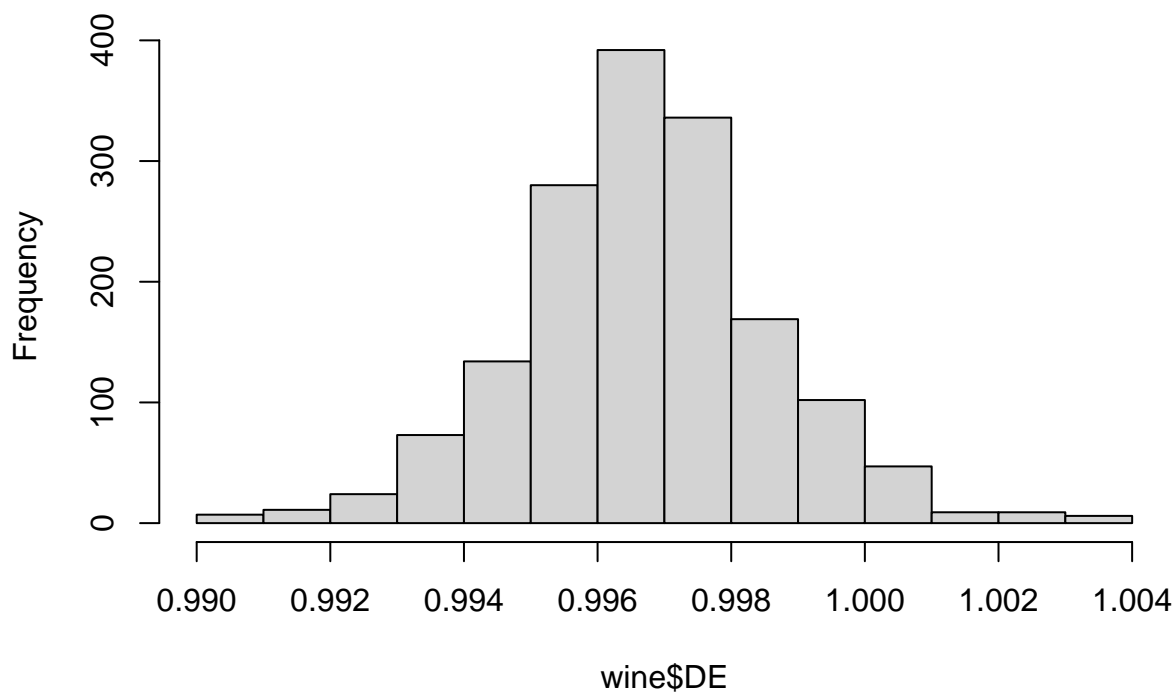
```
hist(wine$SD)
```

Histogram of wine\$SD



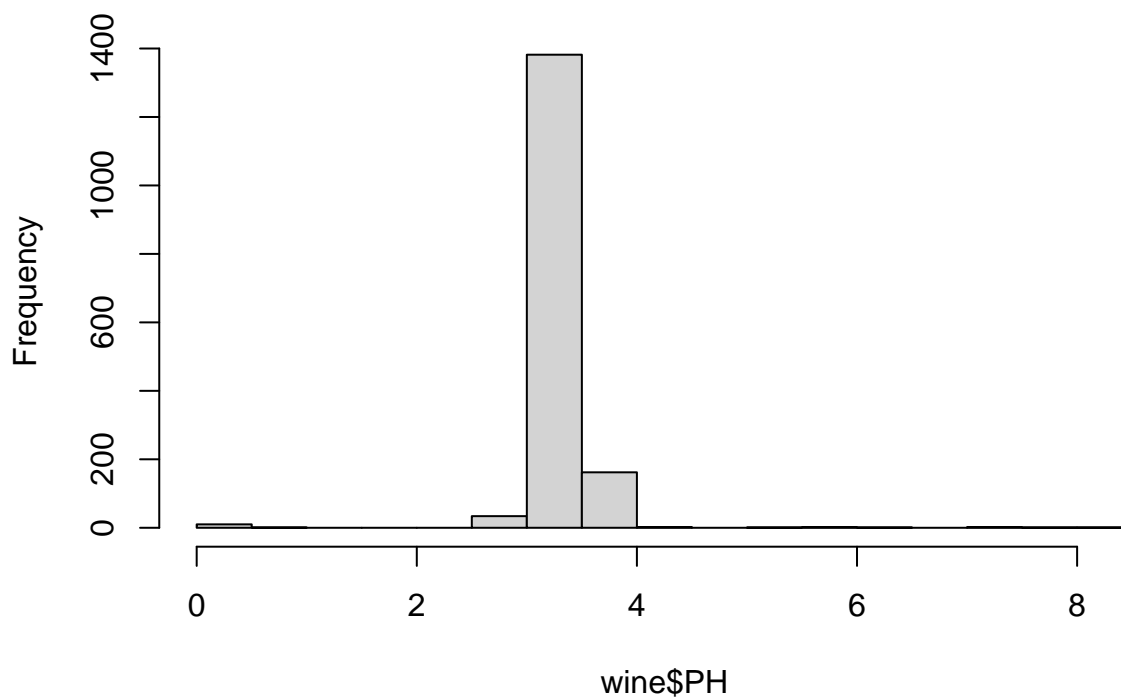
```
hist(wine$DE)
```

Histogram of wine\$DE



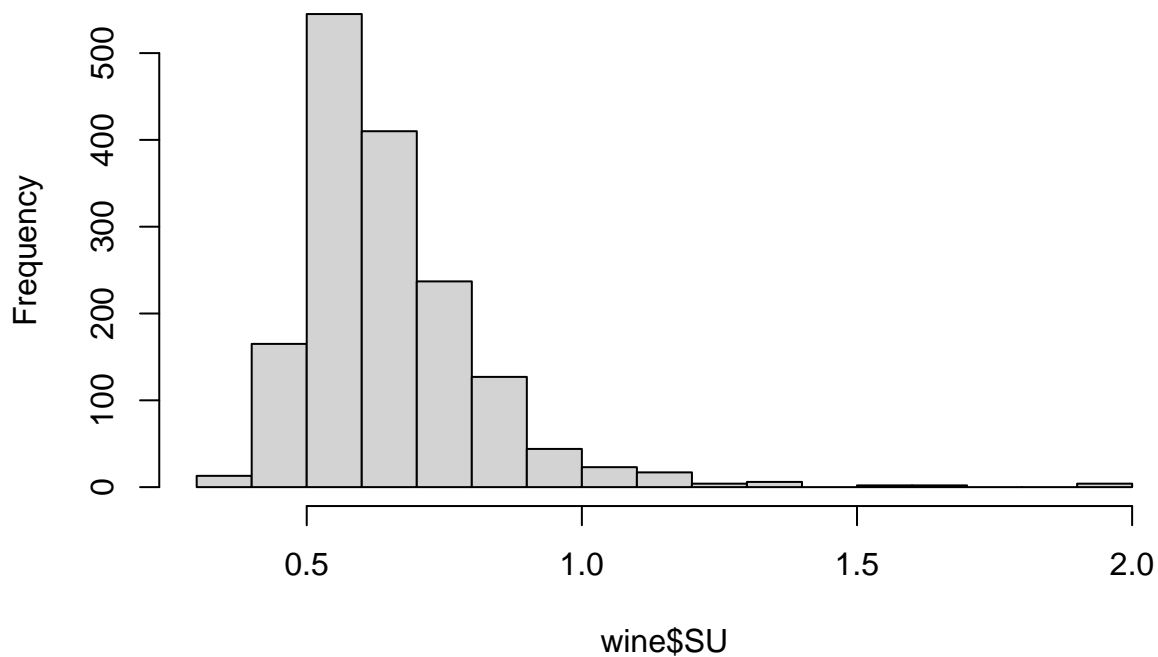
```
hist(wine$PH)
```

Histogram of wine\$PH



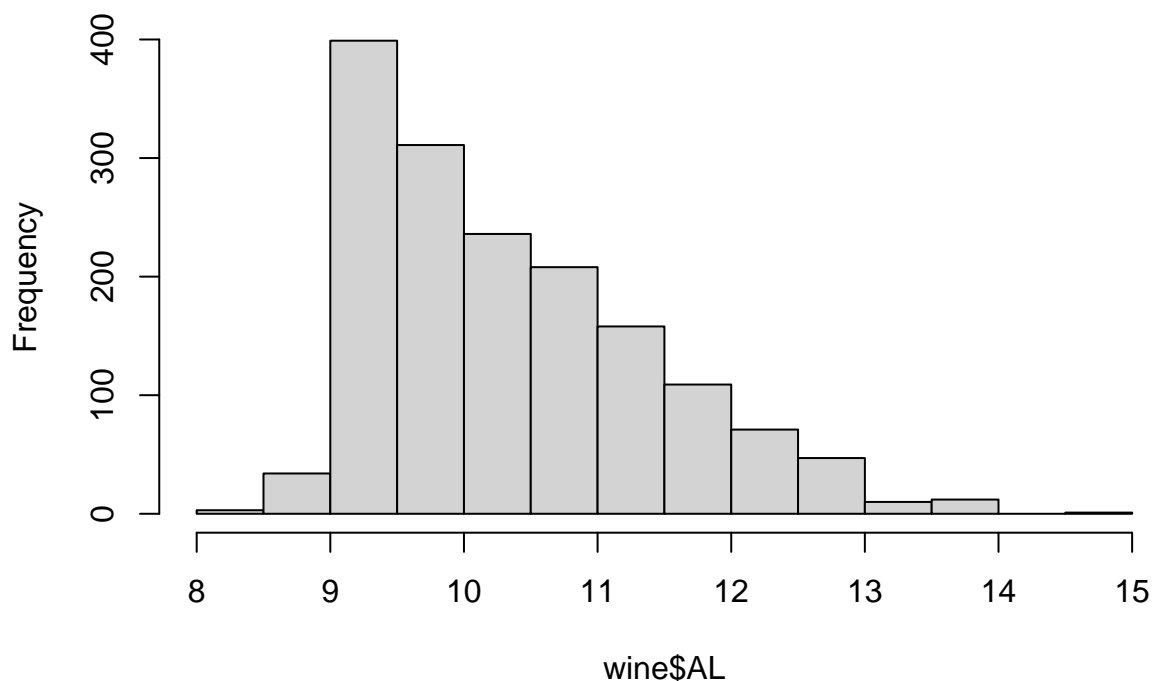
```
hist(wine$SU)
```

Histogram of wine\$SU



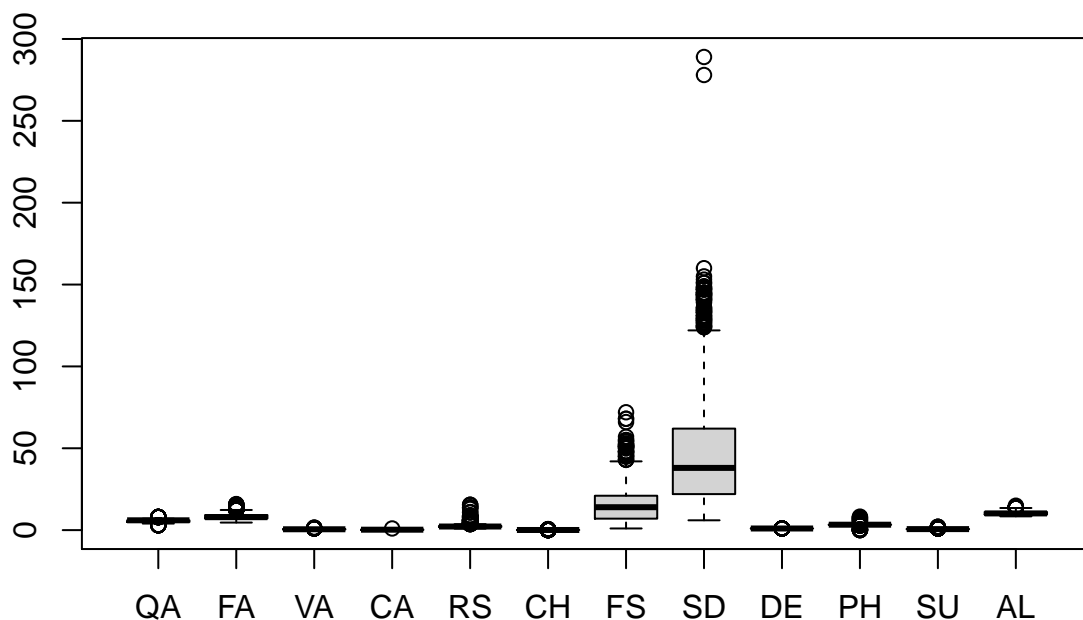
```
hist(wine$AL)
```

Histogram of wine\$AL

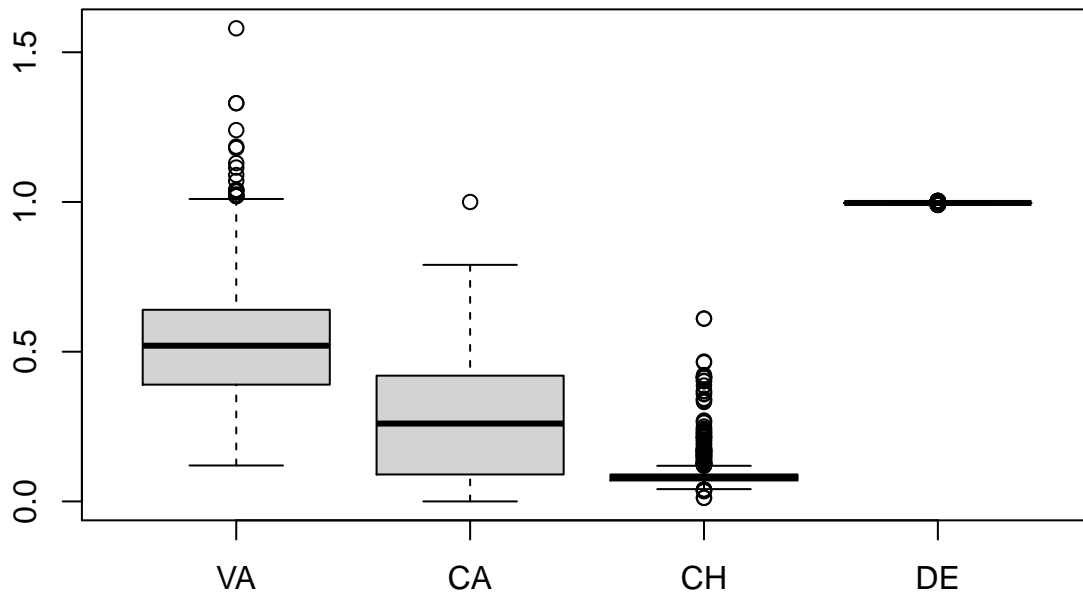


Problem 1b) For each variable, plot the distribution using a box-and-whisker plot. Are there any significant outliers? Note: some variables have comparable scales, while others do not.

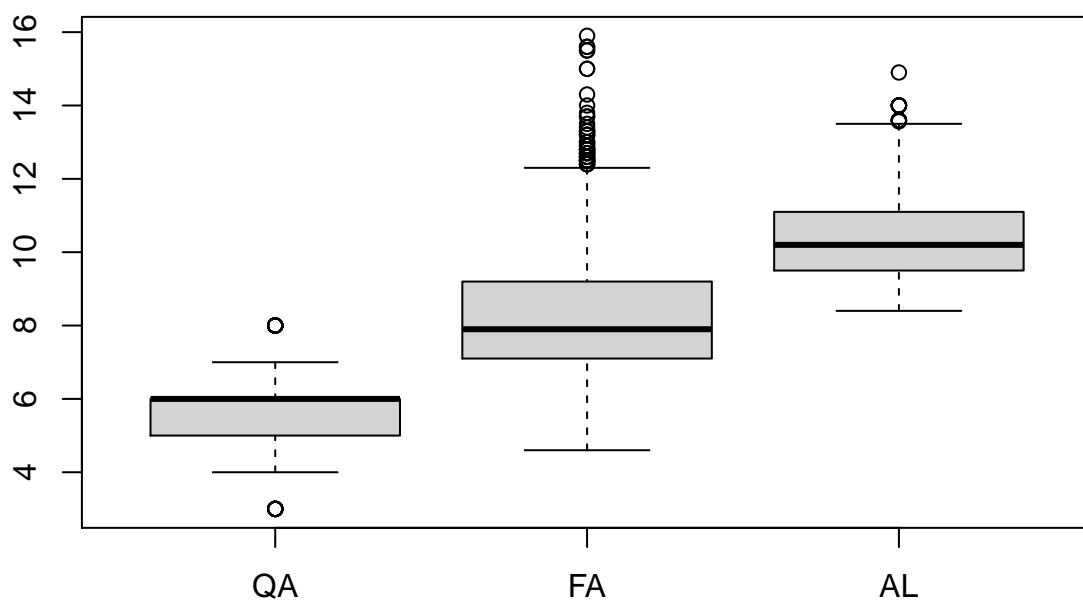
```
boxplot(wine)
```



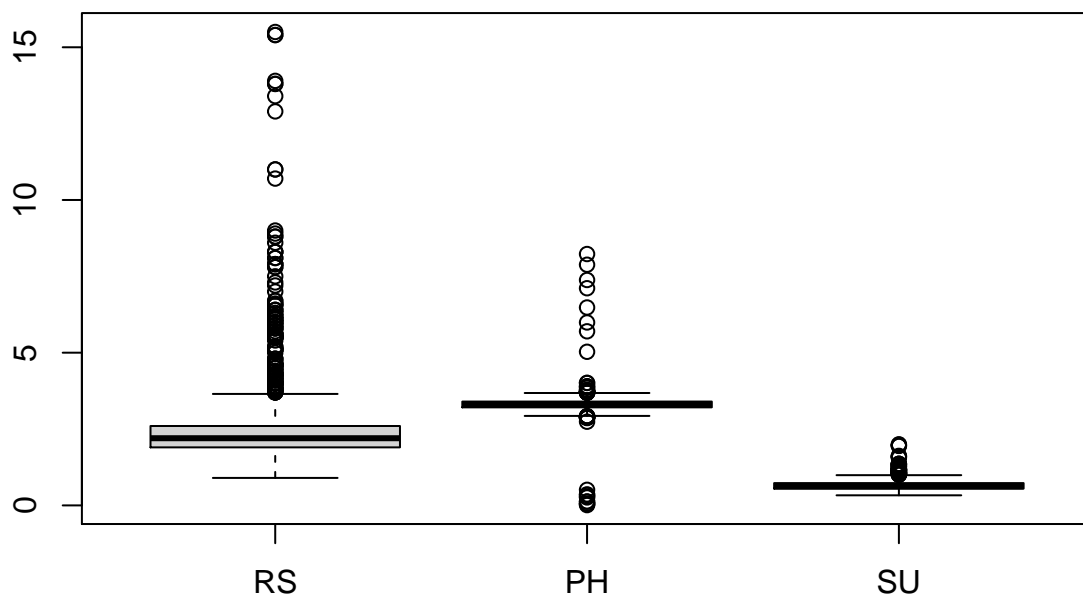
```
boxplot(wine %>% select(3,4,6,9))
```



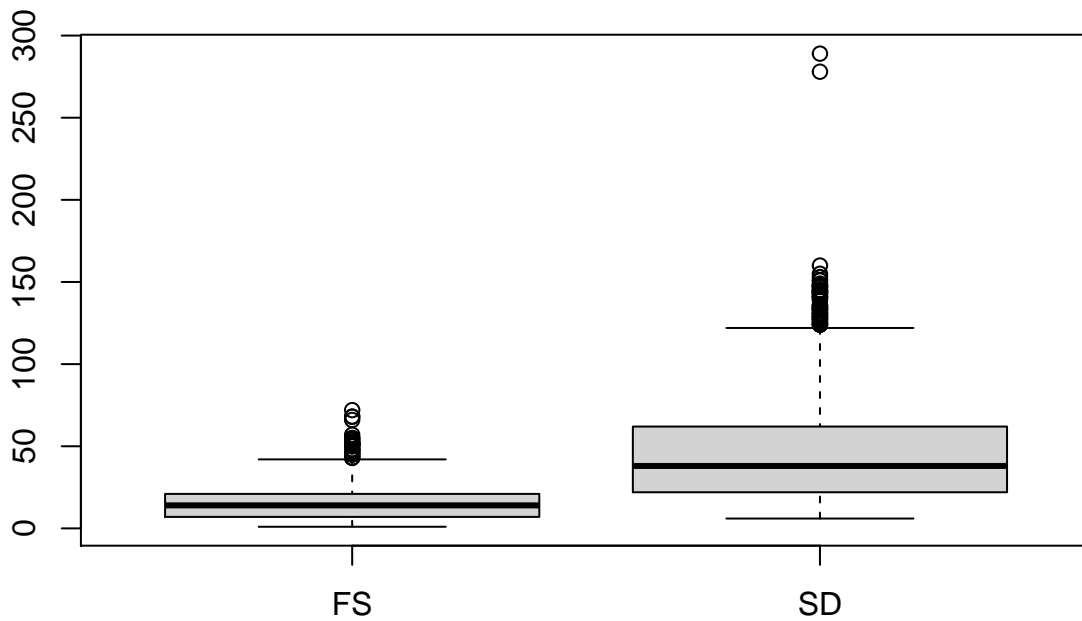
```
boxplot(wine %>% select(1,2,12))
```



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




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



The various box plots indicate 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
```

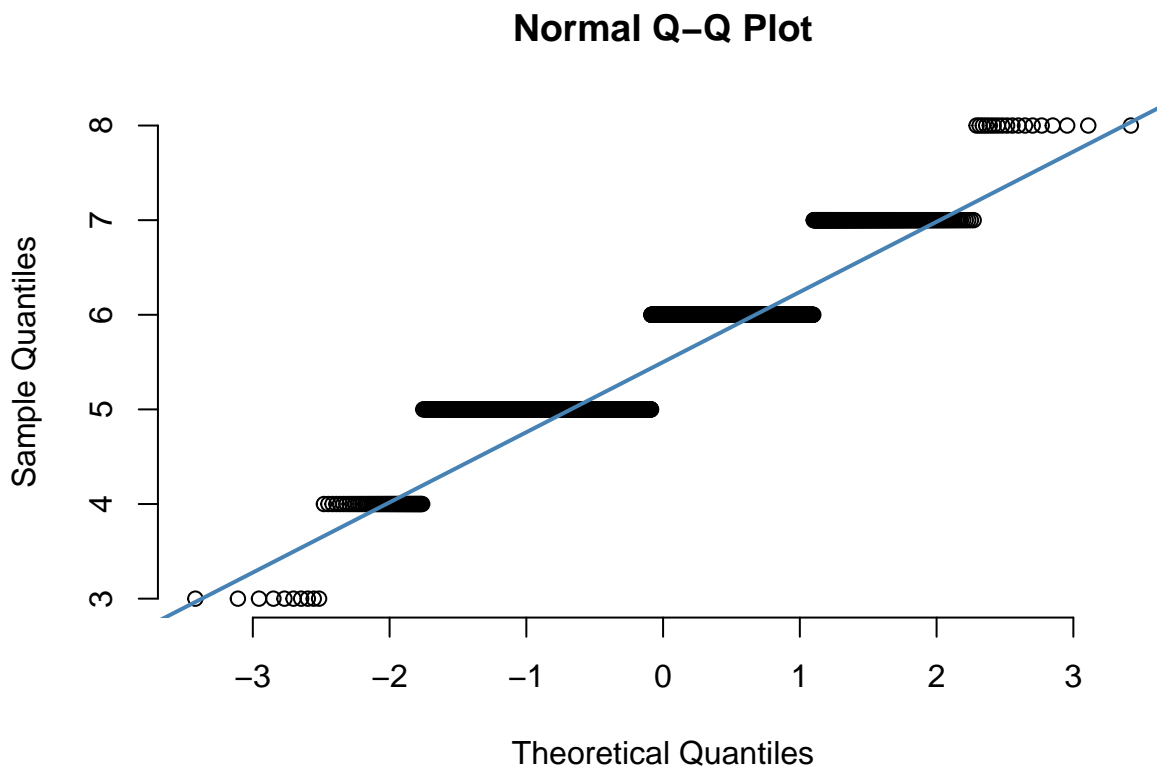
```
kurtosis(wine$AL)
```

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

No variables were left skewed. SU, SD, FS, CH, RS, were significantly right skewed while the others were slightly right skewed. 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.

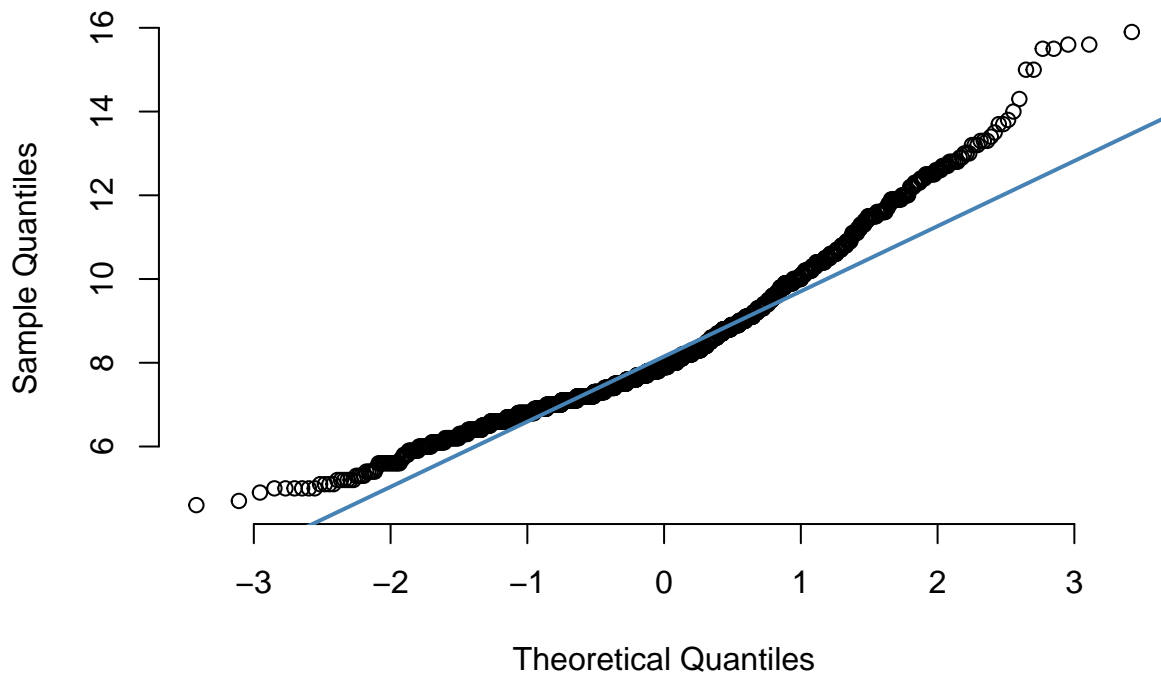
1d) For each variable, display the Q-Q plot. Do they confirm your observations from previous parts of Problem 1?

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



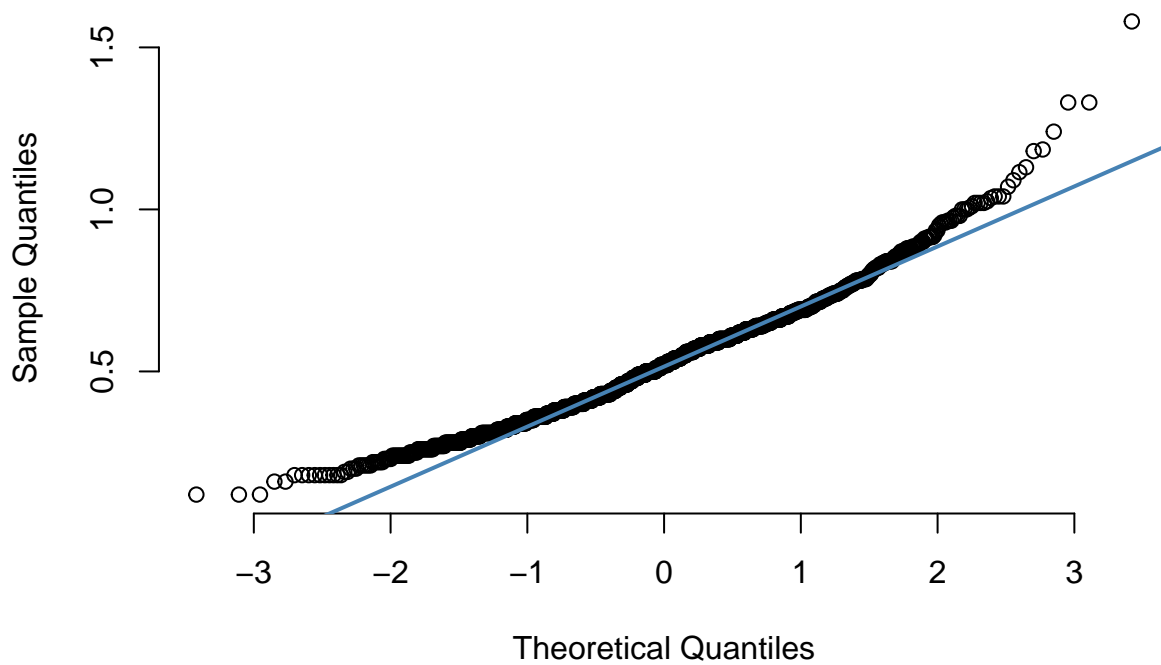
```
qqnorm(wine$FA, pch = 1, frame = FALSE)
qqline(wine$FA, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



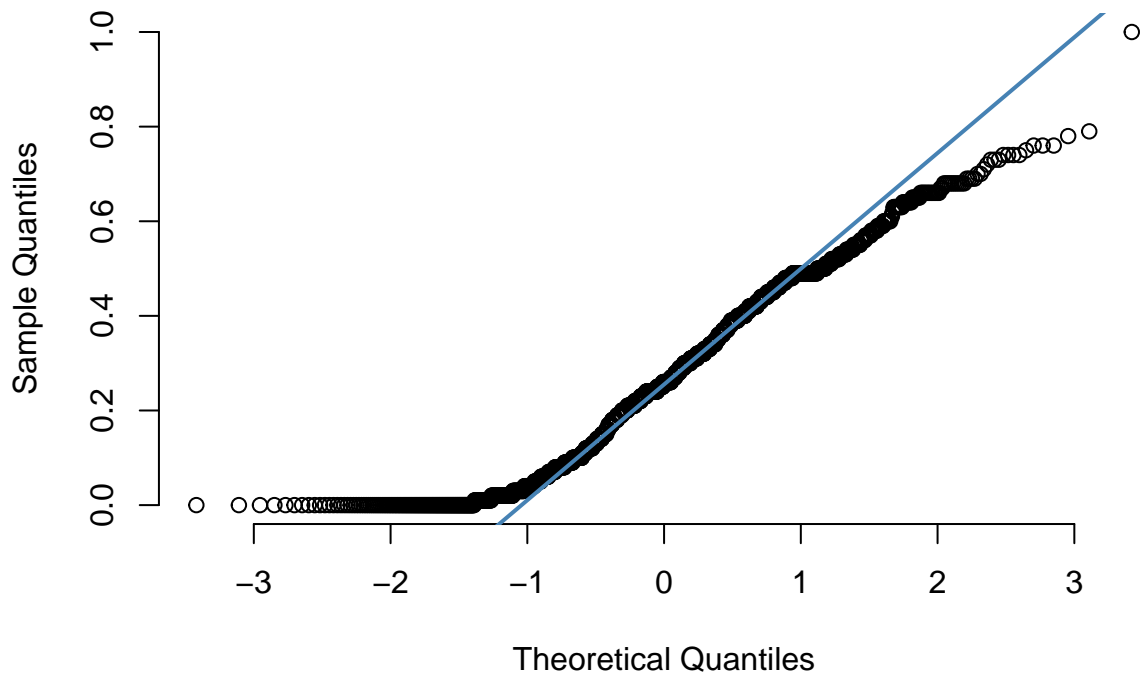
```
qqnorm(wine$VA, pch = 1, frame = FALSE)
qqline(wine$VA, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



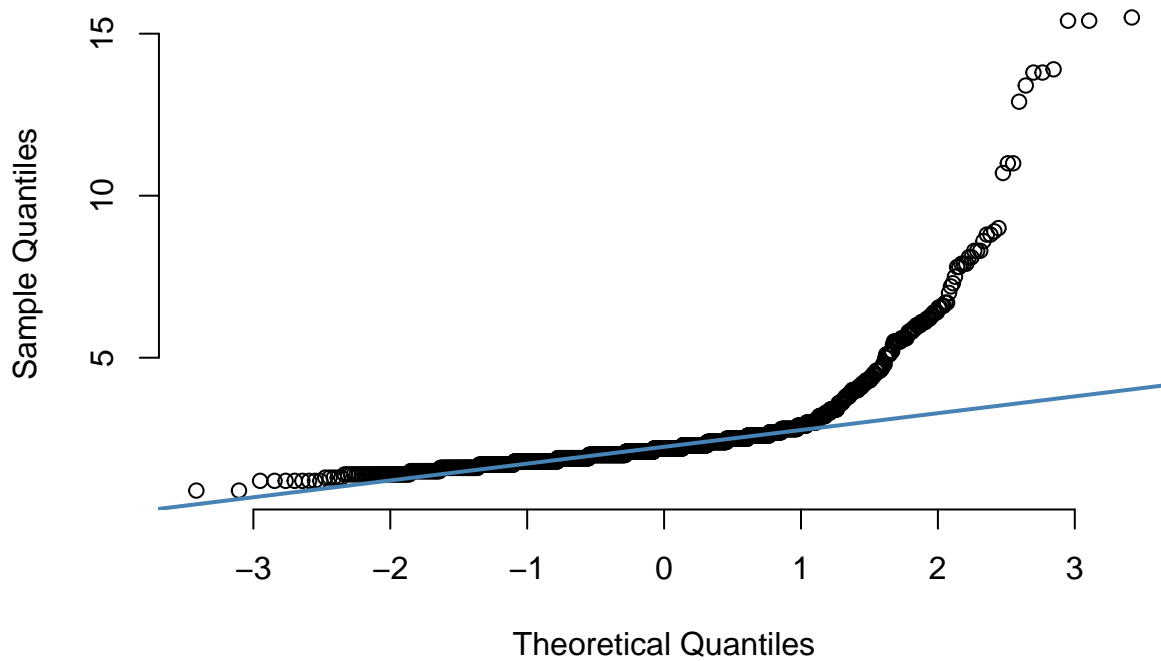
```
qqnorm(wine$CA, pch = 1, frame = FALSE)
qqline(wine$CA, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



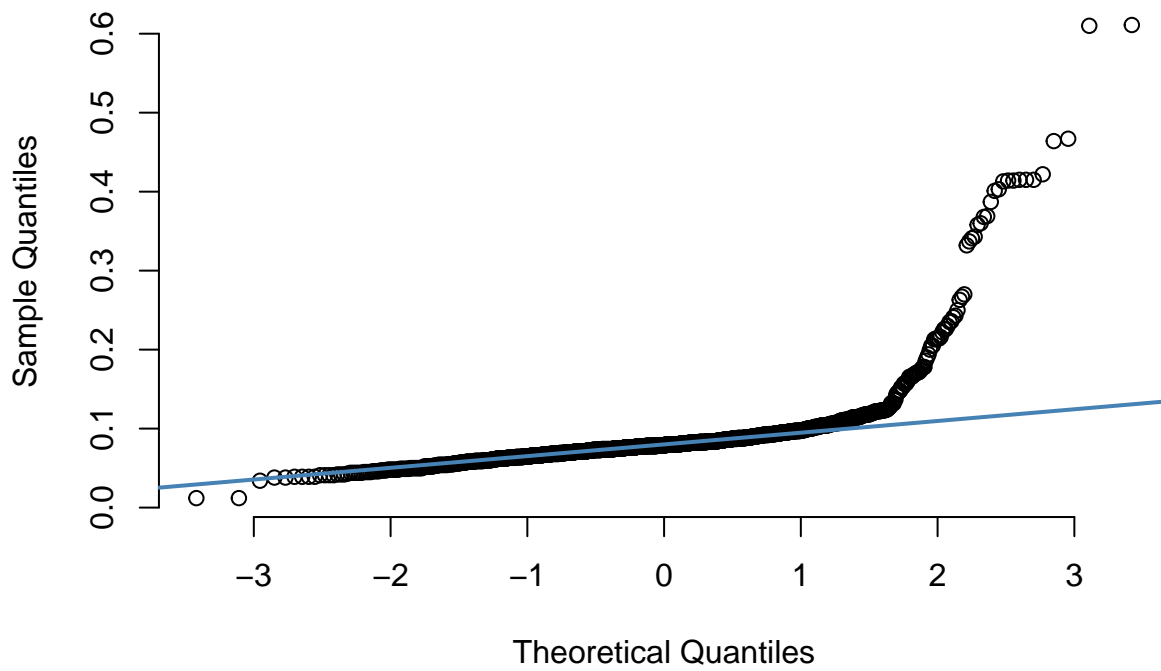
```
qqnorm(wine$RS, pch = 1, frame = FALSE)
qqline(wine$RS, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



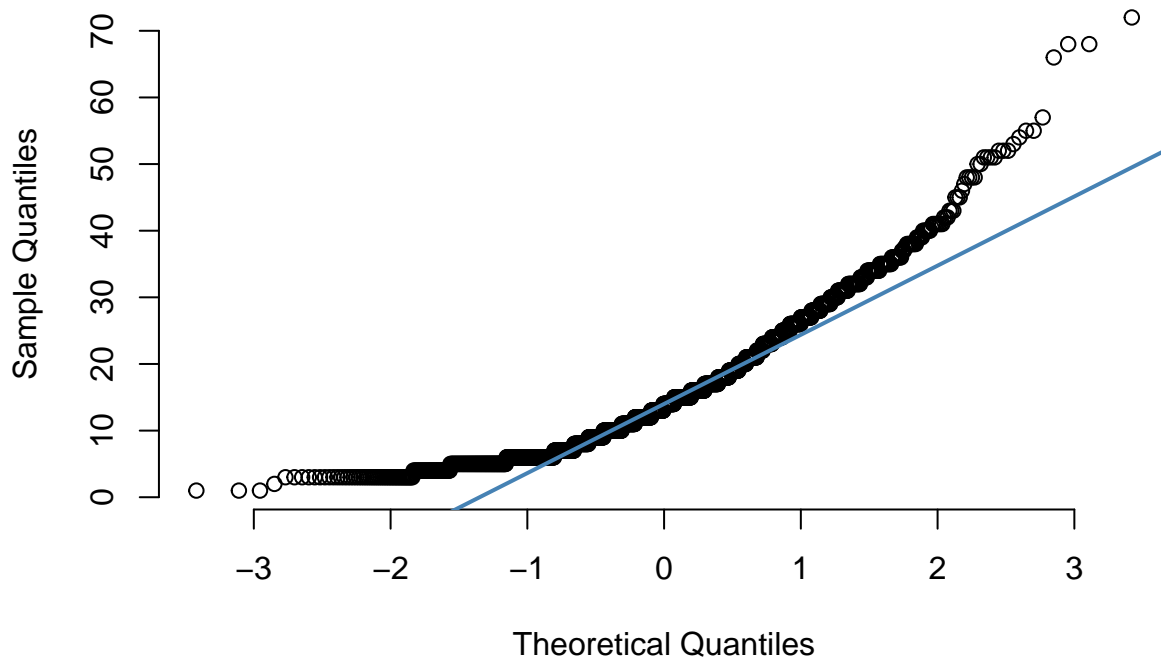
```
qqnorm(wine$CH, pch = 1, frame = FALSE)
qqline(wine$CH, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



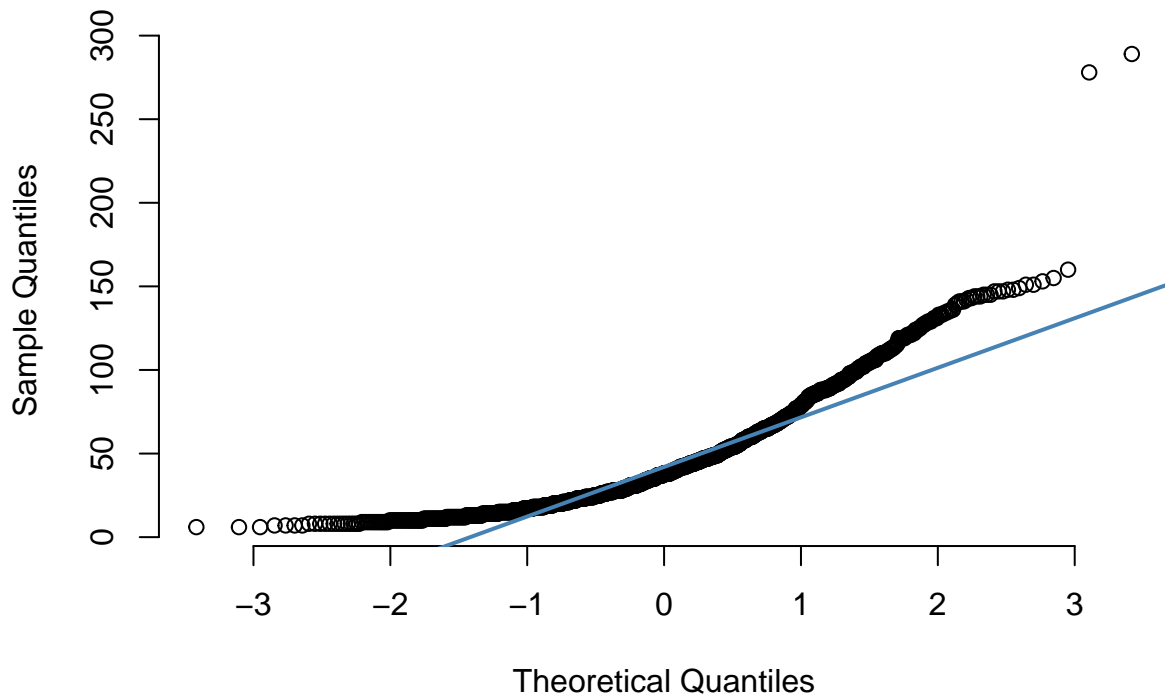
```
qqnorm(wine$FS, pch = 1, frame = FALSE)
qqline(wine$FS, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



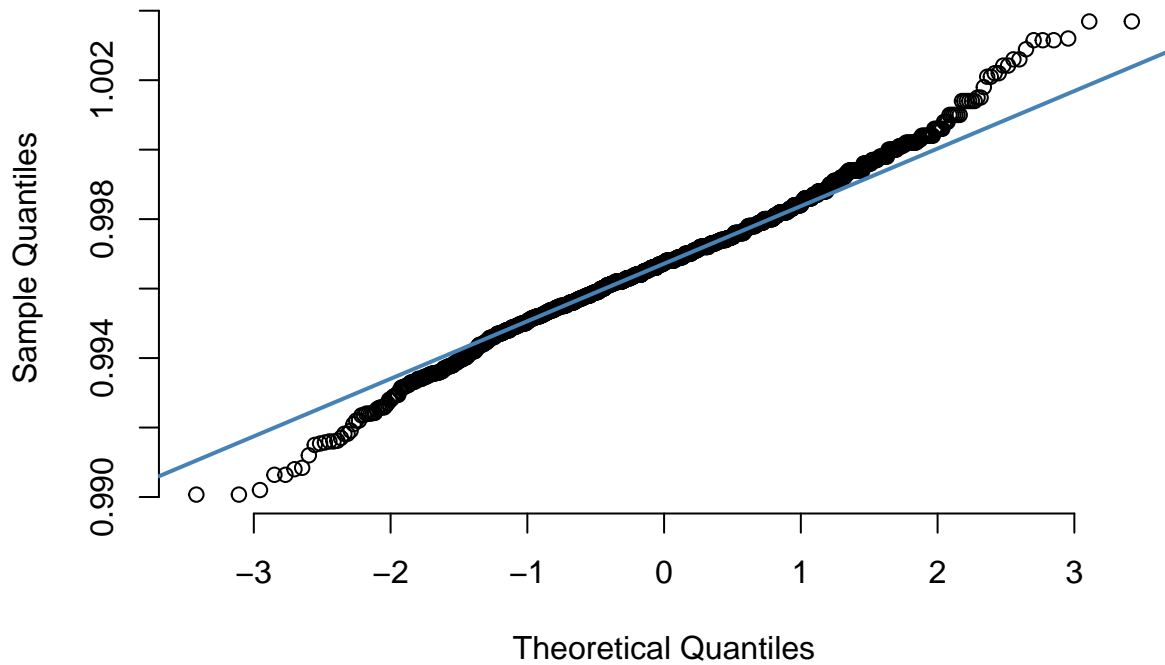
```
qqnorm(wine$SD, pch = 1, frame = FALSE)
qqline(wine$SD, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



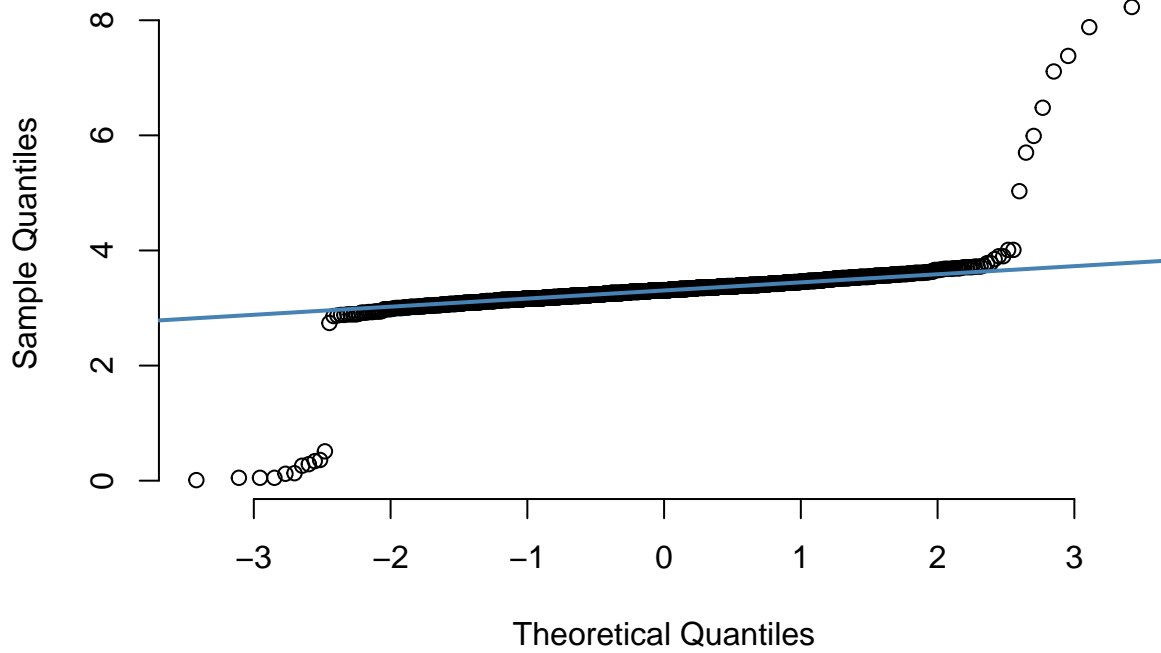
```
qqnorm(wine$DE, pch = 1, frame = FALSE)
qqline(wine$DE, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



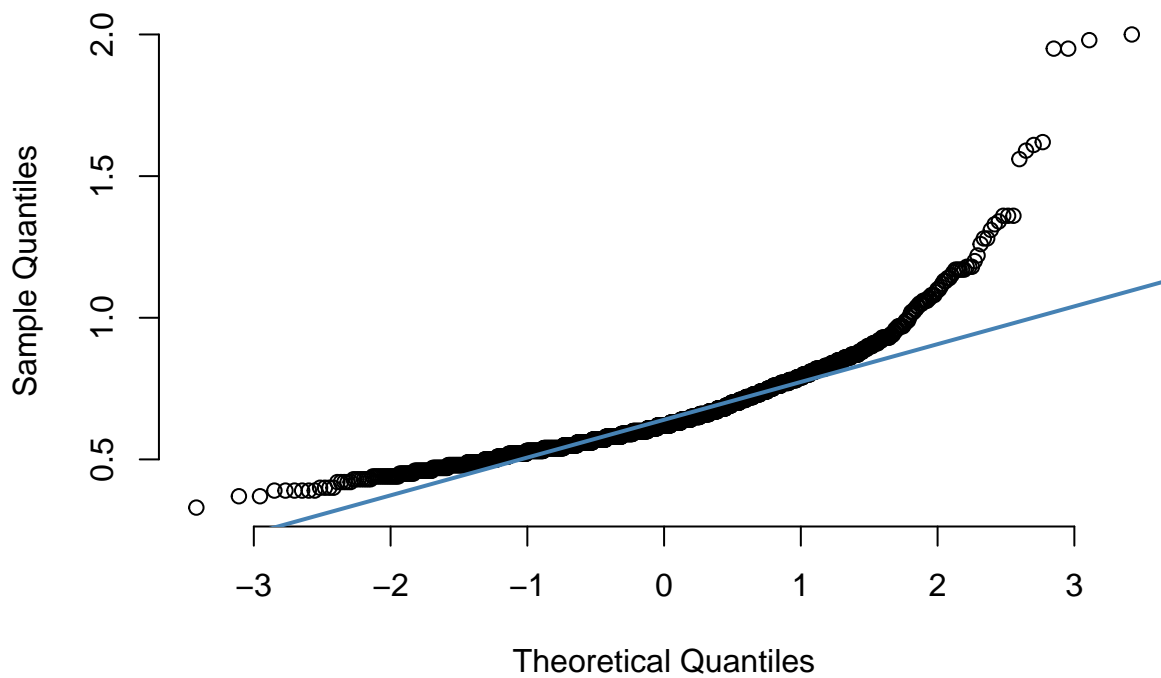
```
qqnorm(wine$PH, pch = 1, frame = FALSE)
qqline(wine$PH, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



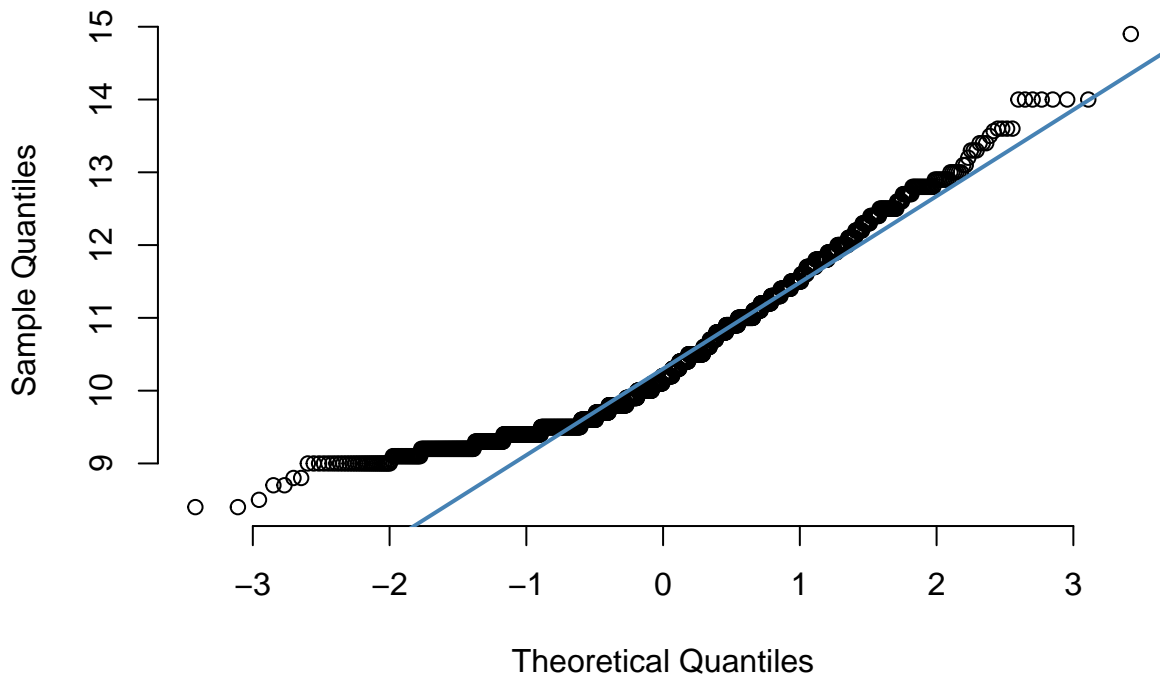
```
qqnorm(wine$SU, pch = 1, frame = FALSE)
qqline(wine$SU, col = "steelblue", lwd = 2)
```

Normal Q-Q Plot



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


Normal Q–Q Plot



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) Use `is.na` to determine which variables have missing values. How many missing values are there in each variable (Hint: use `colSums`)? How many samples have missing values (Hint: use `rowSums`)?

```
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)

Split the dataset into 5-folds. For each fold, use random sampling (from the training set) to fill in the missing values. Train a linear regression model. Compute the Mean Squared Error (MSE)

```
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)]),replace = 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)]),replace = 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(DMwR)

## Loading required package: grid

## 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

```
library(mice)
```

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

## 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='pmm'))
  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] 0.4174154
```

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

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

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

```
## [1] 0.425426
```

2g) Remove NA

```

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. This may be because of the fact that only a small percentage of rows had to be omitted and the model could be trained on the remaining true values of the dataset.