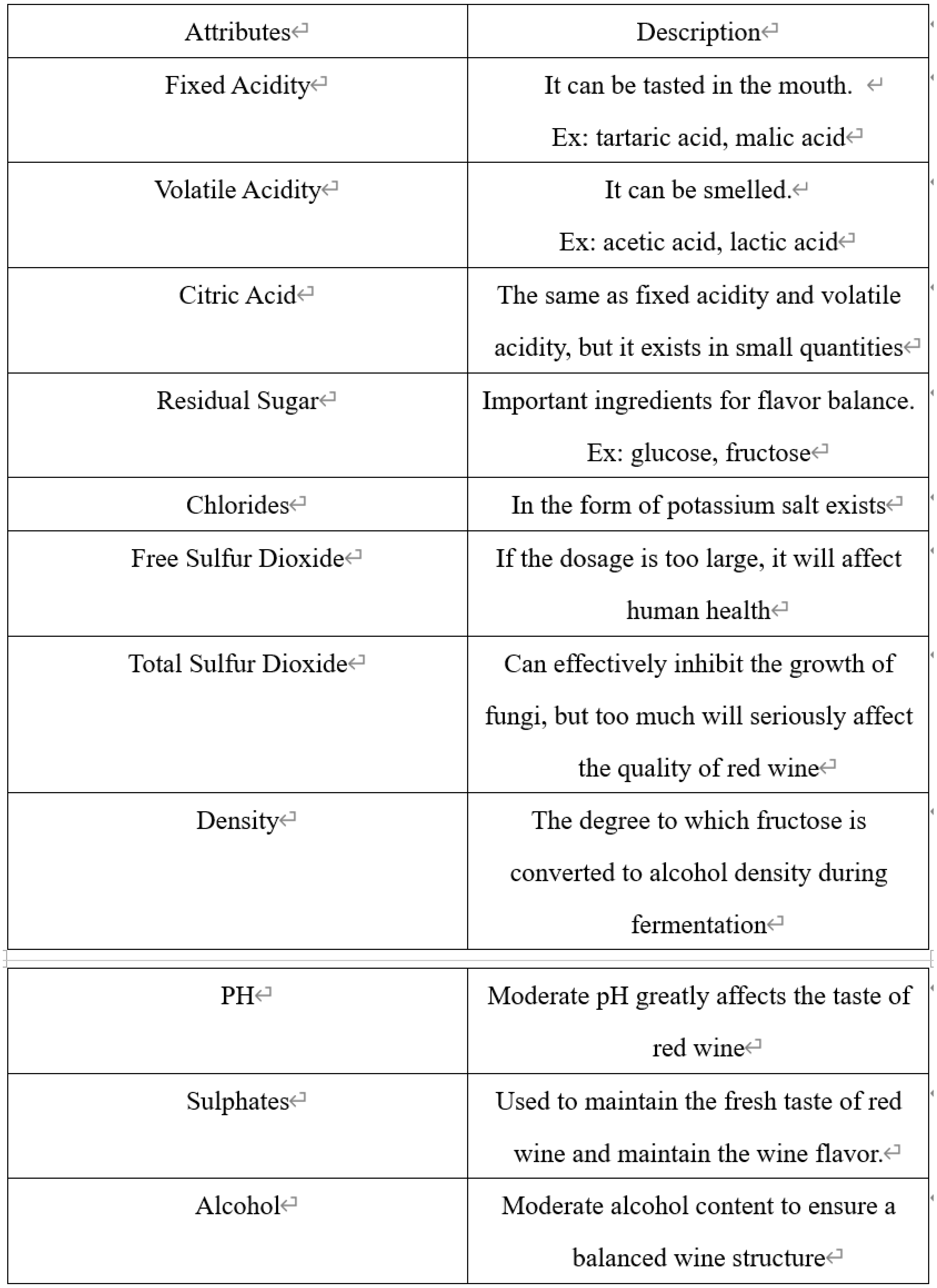
Group_1

David,Andi,Justin

2022-05-22

Motivation: Red wine has received more and more attention in the field of health preservation. Therefore, how to taste fine wine and choose red wine has become a problem that more and more people are concerned about. What physical and chemical properties of red wine can meet people’s needs for health preservation?

Data: Our data set is from Kaggle. The entire data set has 12 attributes. One of the variables “quality” represents the rating of the wine by wine experts (0-10). The table below shows details



Attributes	Description
Fixed Acidity	It can be tasted in the mouth. Ex: tartaric acid, malic acid
Volatile Acidity	It can be smelled. Ex: acetic acid, lactic acid
Citric Acid	The same as fixed acidity and volatile acidity, but it exists in small quantities
Residual Sugar	Important ingredients for flavor balance. Ex: glucose, fructose
Chlorides	In the form of potassium salt exists
Free Sulfur Dioxide	If the dosage is too large, it will affect human health
Total Sulfur Dioxide	Can effectively inhibit the growth of fungi, but too much will seriously affect the quality of red wine
Density	The degree to which fructose is converted to alcohol density during fermentation
PH	Moderate pH greatly affects the taste of red wine
Sulphates	Used to maintain the fresh taste of red wine and maintain the wine flavor.
Alcohol	Moderate alcohol content to ensure a balanced wine structure

From the background information, residual sugar, all acidity, and alcohol combine to create a wine’s balance of flavor: acidity provides depth to the wine’s flavor, while neutralizing the residual sugar to keep the wine fresh; alcohol adds to the palate more burning sensation. So, we can know that the quality of red wine mostly depends on “Fixed Acidity”, “Volatile Acidity”, “Citric Acid”, “Residual Sugar”, “PH”, “Density” and “Alcohol”.

Objective: After knowing some background knowledge of red wine and simple data processing, we asked the following questions for this project: 1. What is the difference between a high-rated red wine and a low-rated red wine? 2. To what extent do the physical and chemical properties of red wine determine the quality of red wine?

Load Packages

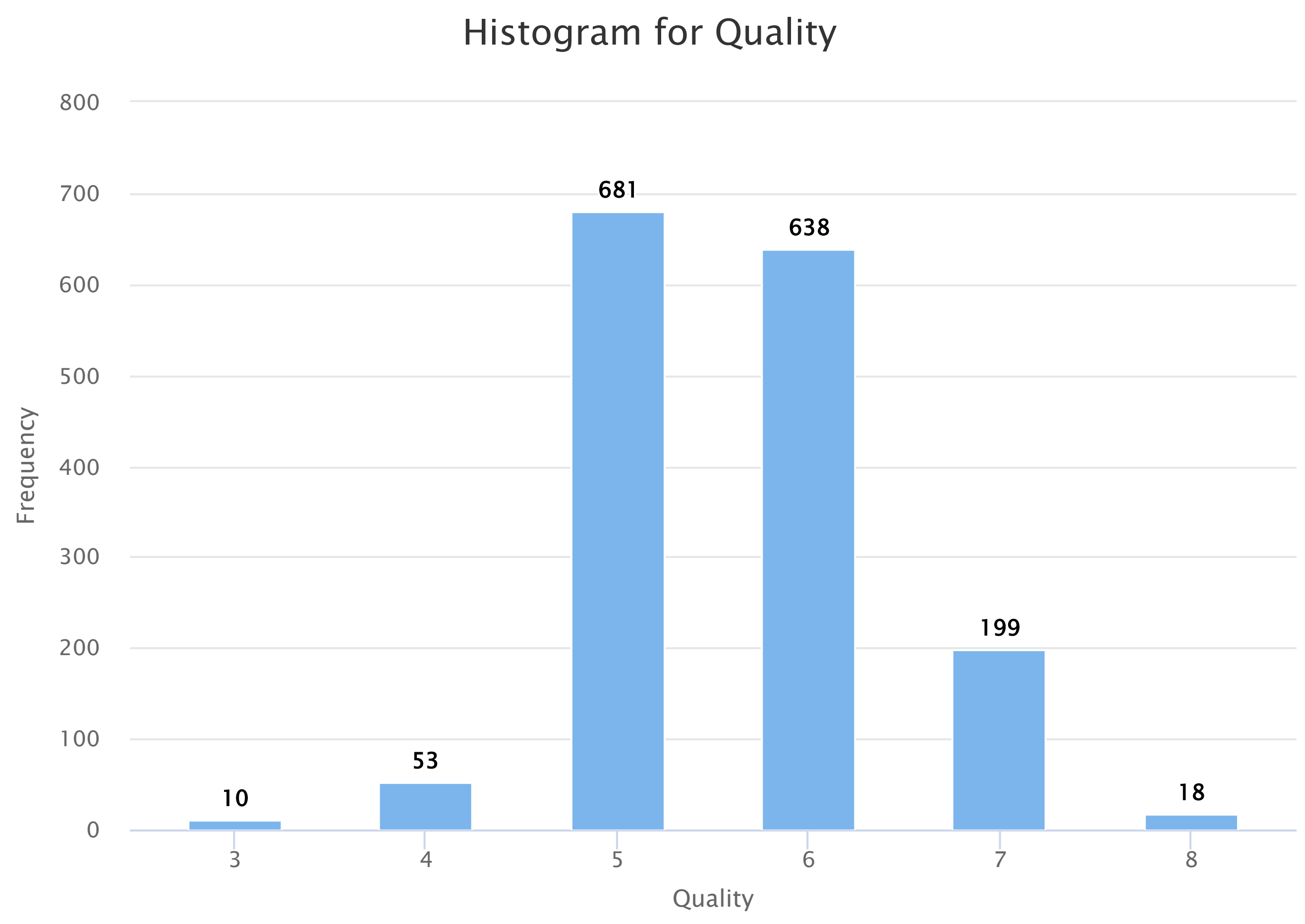
Read CSV

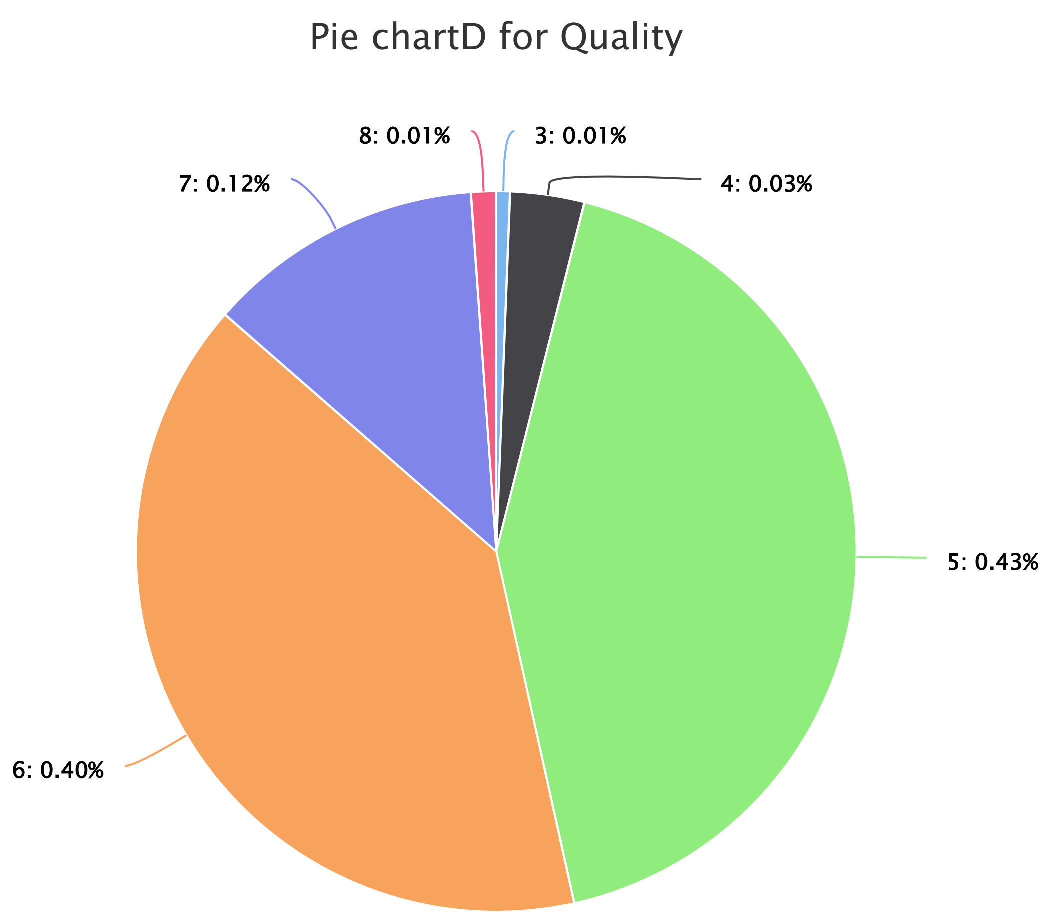
Brief Introduction of Data

```
## 'data.frame':    1599 obs. of  8 variables:
##  $ fixed.acidity   : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity: num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid     : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar  : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ density         : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH              : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ alcohol         : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality         : int  5 5 5 6 5 5 5 7 7 5 ...
```

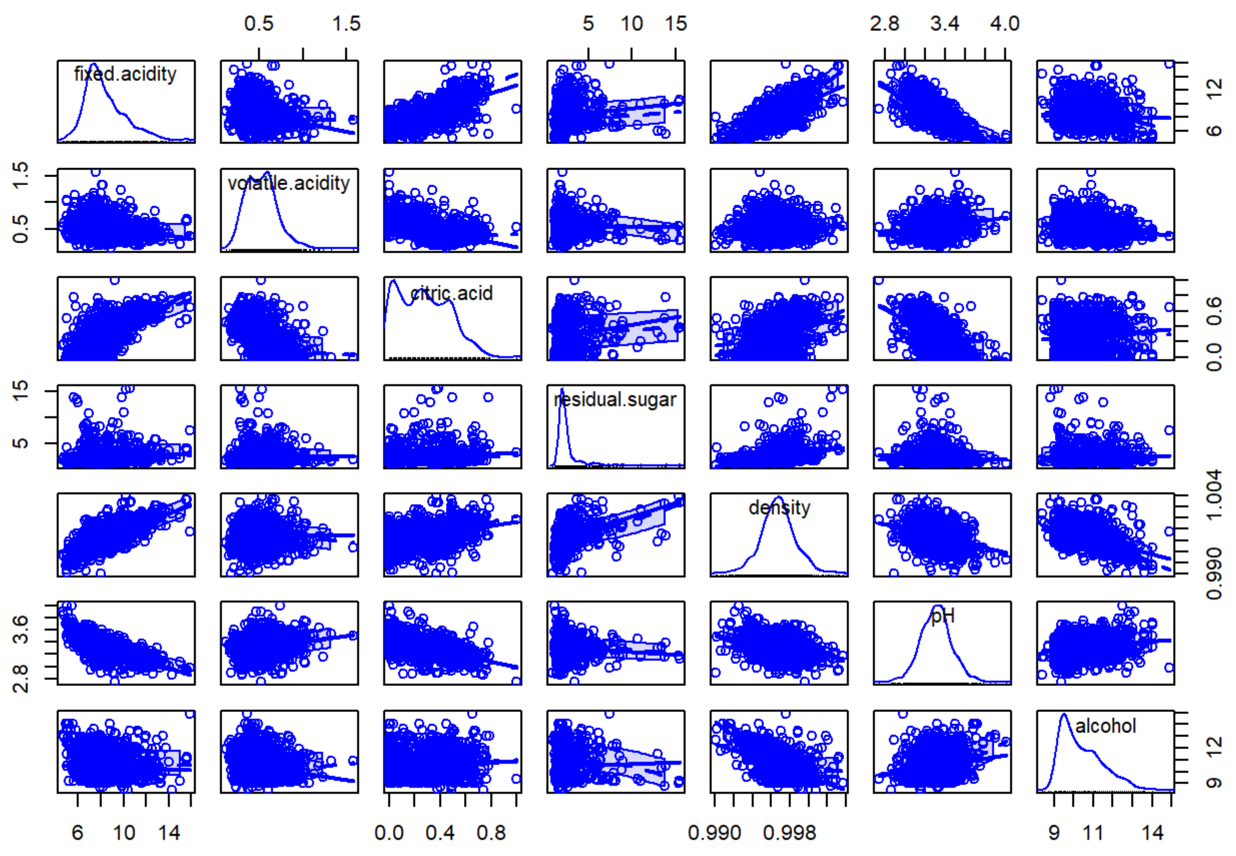
```
##                  vars    n  mean   sd median trimmed  mad  min   max range skew
## fixed.acidity       1 1599  8.32 1.74   7.90    8.15 1.48 4.60 15.90 11.30 0.98
## volatile.acidity    2 1599  0.53 0.18   0.52    0.52 0.18 0.12  1.58  1.46 0.67
## citric.acid         3 1599  0.27 0.19   0.26    0.26 0.25 0.00  1.00  1.00 0.32
## residual.sugar      4 1599  2.54 1.41   2.20    2.26 0.44 0.90 15.50 14.60 4.53
## density             5 1599  1.00 0.00   1.00    1.00 0.00 0.99  1.00  0.01 0.07
## pH                  6 1599  3.31 0.15   3.31    3.31 0.15 2.74  4.01  1.27 0.19
## alcohol             7 1599 10.42 1.07  10.20   10.31 1.04 8.40 14.90  6.50 0.86
## quality             8 1599  5.64 0.81   6.00    5.59 1.48 3.00  8.00  5.00 0.22
##                  kurtosis   se
## fixed.acidity        1.12 0.04
## volatile.acidity     1.21 0.00
## citric.acid         -0.79 0.00
## residual.sugar      28.49 0.04
## density              0.92 0.00
## pH                   0.80 0.00
## alcohol              0.19 0.03
## quality              0.29 0.02
```

Visualization





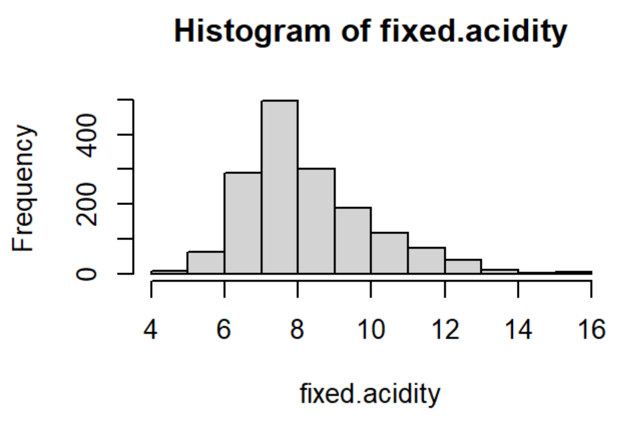
In subsequent research, we determined that a score higher than 7 was a good wine, and a score lower than 7 was a bad wine.

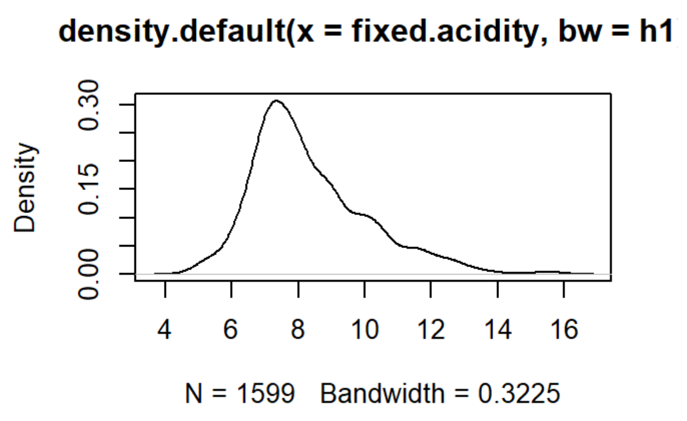


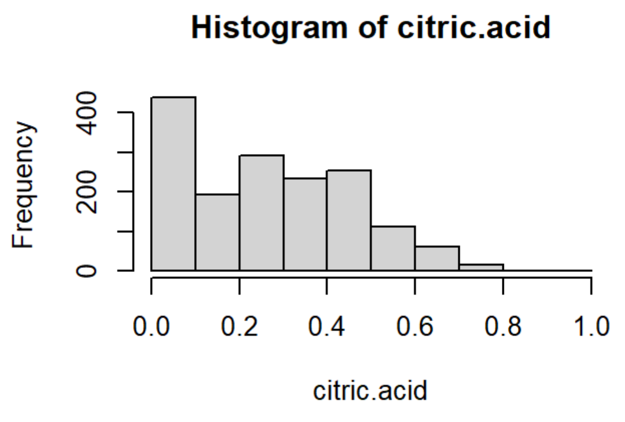
Define Estimate Function

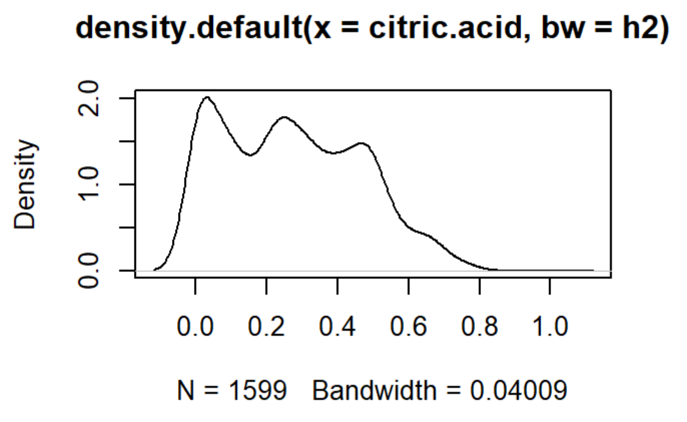
```
# Jack Knife
jackknife<-function(col){
    n<-length(col)
    jackmean<-numeric(n)
    jackvar<-numeric(n)
    for (i in 1:n){
        jackmean[i]<-mean(col[-i])
        jackvar[i]<-var(col[-i])
    }
    return(c(mean(jackmean),mean(jackvar)))
}

# Bootstrap mean and var
bootstrap<-function(seed_num,col,B){
    set.seed(seed_num)
    n<-length(col)
    bootmean<-numeric(n)
    bootvar<-numeric(n)
    for (b in 1:B){
        i<-sample(1:n,size = n,replace = TRUE)
        bootmean[i]<-mean(col[i])
        bootvar[i]<-var(col[i])
    }
    return(c(mean(bootmean[i]),mean(bootvar[i])))
}

# Bootstrap bias and error
bootstrapE<-function(seed_num,col,B){
    set.seed(seed_num)
    n<-length(col)
    bootmean<-numeric(n)
    bootvar<-numeric(n)
    for (b in 1:B){
        i<-sample(1:n,size = n,replace = TRUE)
        bootmean[i]<-mean(col[i])
        bootvar[i]<-var(col[i])
    }
    bias_mean<-mean(bootmean[i])-mean(col)
    bias_var<-mean(bootvar[i])-var(col)
    se_mean<-sd(bootmean)
    se_var<-sd(bootvar)
    return(c(bias_mean,se_mean,bias_var,se_var))
}

## with no optimization
MLEOP<- function(theta,x) {
    mu<-theta[1]
    var<-theta[2]
    return (-(-(n/2)*log(2*pi)-(n/2)*log(var)-(1/(2*var))*sum((x-mu)^2)))
}

# EM algorithm for mixture estimations
gmm <- function(x, mean, sd = NULL)
{
    # initialize
    num <- length(mean)

    epsilon <- 1e-4
    probs <- rep(1/num, num)
    mu_s <- mean
    sigma_s <- sd ^ 2
    n <- length(x)
    while(TRUE)
    {
        # E-step calculate estimated prob
        ps <- matrix(0, ncol = num, nrow = n)
        for(j in seq(num))
        {
            ps[, j] <- probs[j] * dnorm(x, mean = mu_s[j], sd = sqrt(sigma_s[j]))
        }
        ps <- ps / rowSums(ps)

        sigma_s_p <- sigma_s
        # M-step update the mean, sigma and prob
        for(j in seq(num))
        {
            sigma_s[j] <- sum( ps[, j] * (x - mu_s[j])^2) / sum(ps[, j])
            mu_s[j] <- sum(x * ps[, j]) / sum(ps[, j])
            probs[j] <- mean(ps[, j])
        }

        if (max(abs(sigma_s_p - sigma_s)) < epsilon)
        {
            break
        }

    }

    return (list(mu = mu_s, var = sigma_s, prob = probs))

}
```

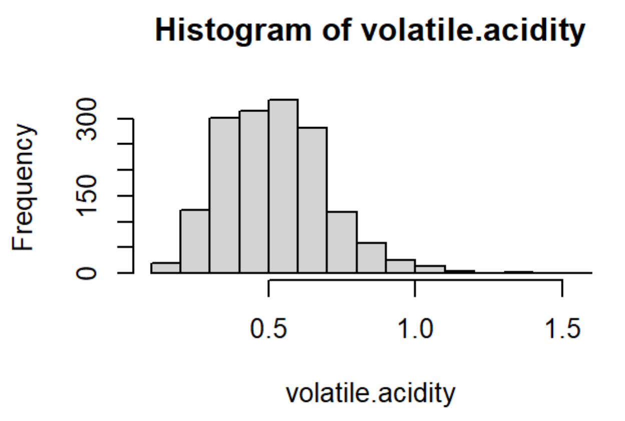
Show the Histogram and Kernel Density Estimates

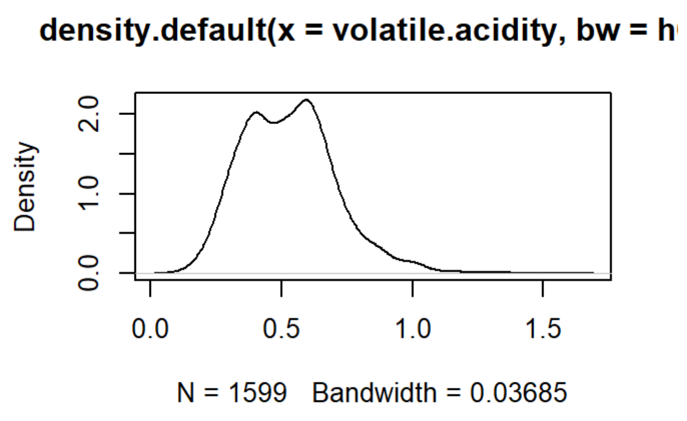


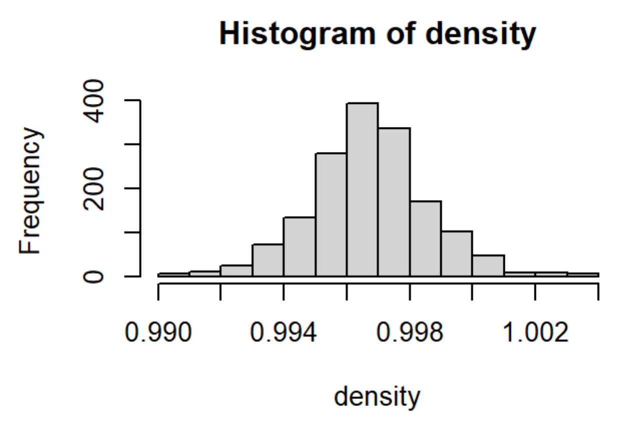


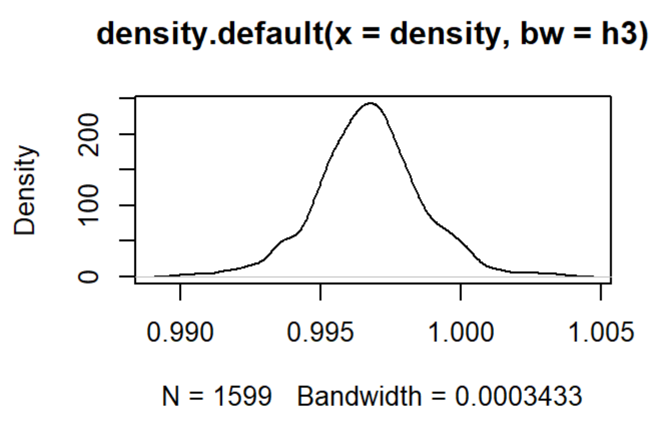


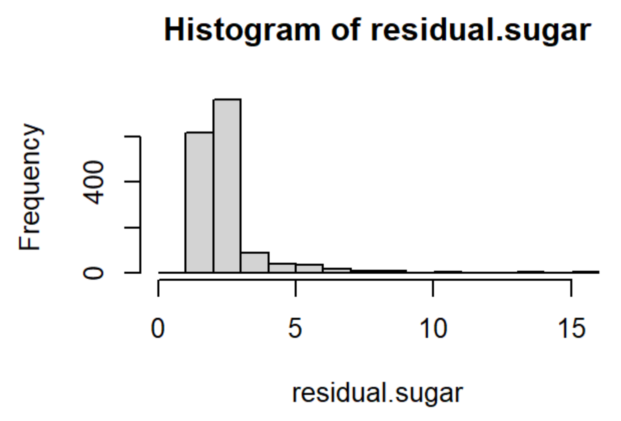


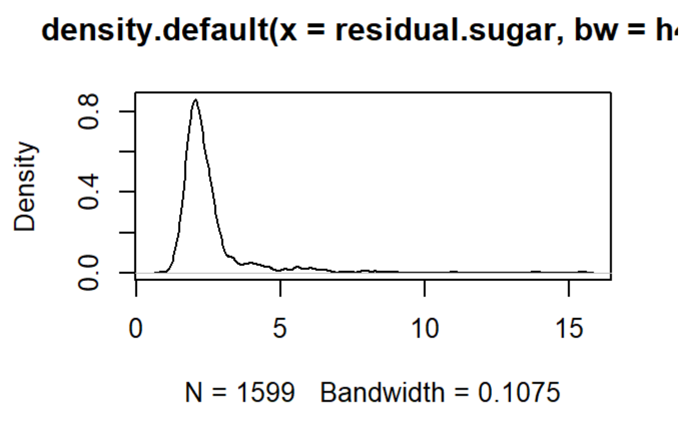


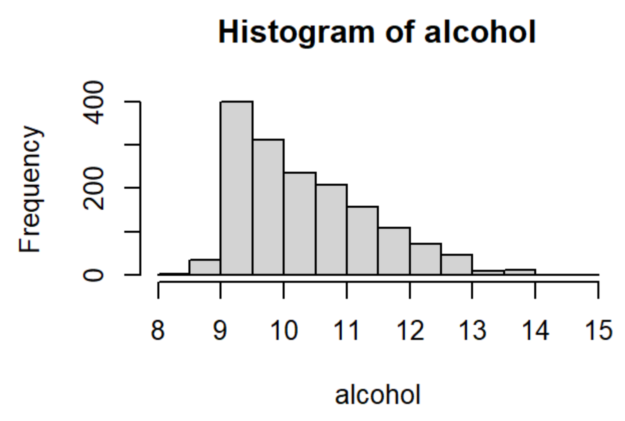


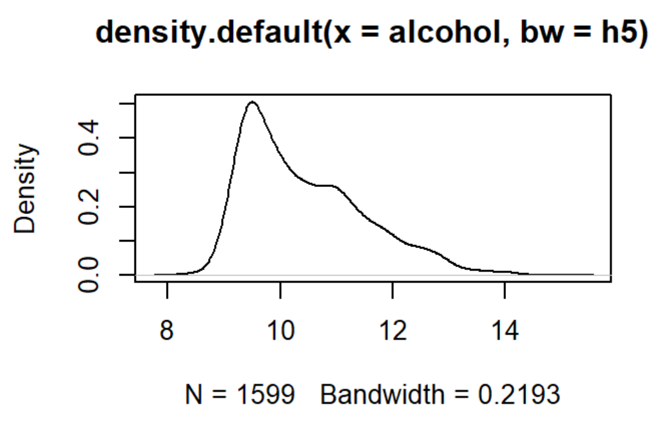


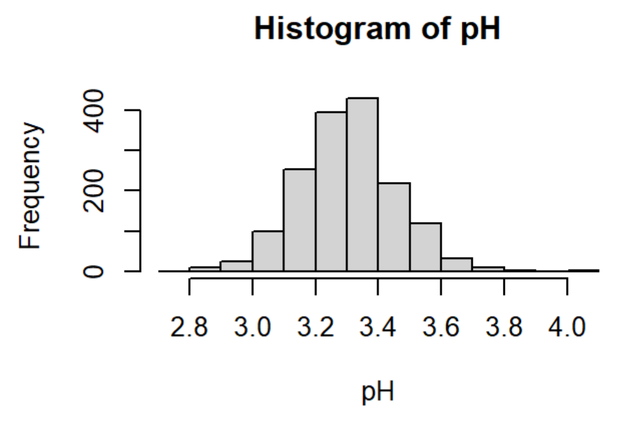


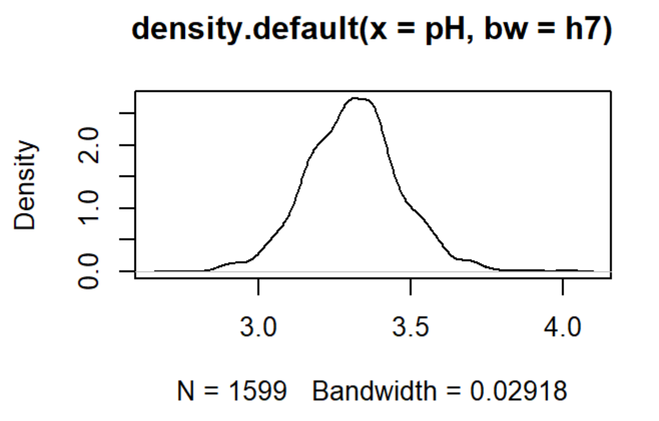






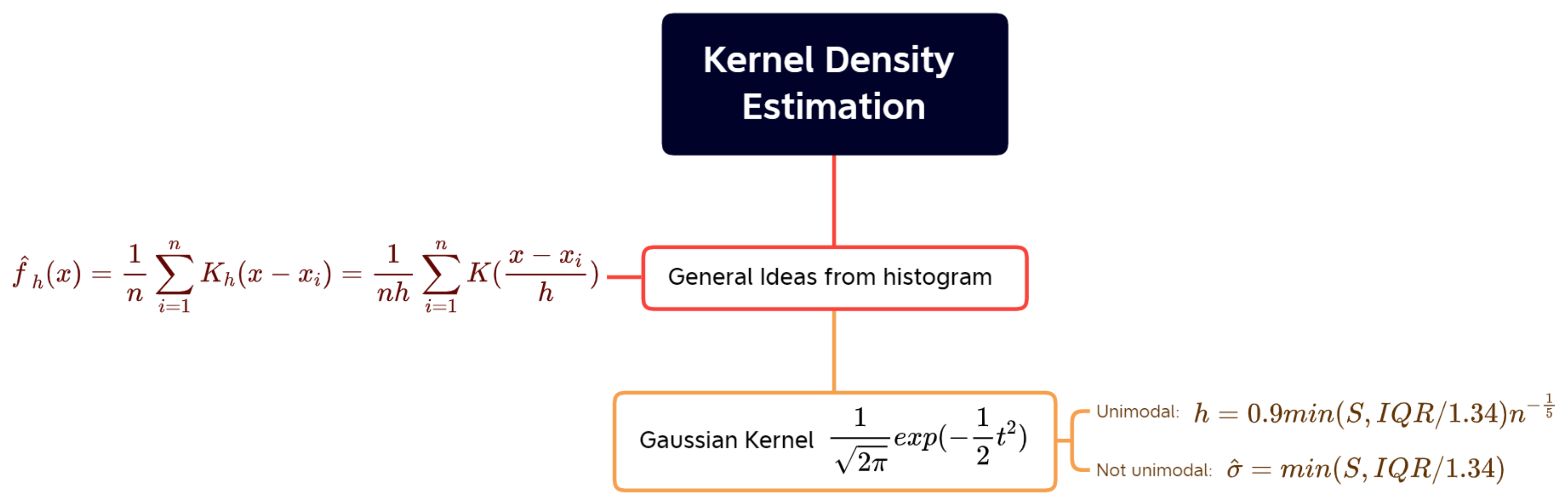






For the Kernel density estimates, all variable we use the Gaussian Kernel, K(t)=1/√2π exp(-1/2 t^2). In addition, for unimodal distribution we set a better estimate for Gaussian \(h=0.9 min(S,\frac{IQR}{1.34})n^{-\frac{1}{5}}\). But for which is not unimodal, we set \(\hat \sigma=min(S,\frac{IQR}{1.34})\).

Explain Keneral Density Estimates


Kernel Density Estimation
$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x-x_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h})$
General Ideas from histogram
Gaussian Kernel $\frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}t^2)$
Unimodal: $h = 0.9min(S, IQR/1.34)n^{-\frac{1}{5}}$
Not unimodal: $\hat{\sigma} = min(S, IQR/1.34)$

Kernel density estimation is used in probability theory to estimate the unknown density function, which belongs to one of the non-parametric test methods. Since the kernel density estimation method does not use prior knowledge about the data distribution, it does not attach any assumptions to the data distribution. The method of studying the characteristics of data distribution from the data sample itself. The ideas of Kernel Density Estimation are generated from histogram. If we draw a histogram, the purpose is to draw a “probability density function”, and a histogram essentially thinks that frequency equals probability. But this assumption is not inevitable. The kernel density function is a means of “smoothing”. For example, let (x1,x2,…,xn) be n sample points that are independent and identically distributed, and its probability density function is f, so our estimate: \(\hat f_h(x)=\frac{1}{nh}=\sum_{i=1}^nK(\frac{x-x_i}{h})\). Here h is the bandwidth. For Gaussian Kernel, if the distribution is unimodal, we need to set \(h=0.9 min(S,\frac{IQR}{1.34})n^{-\frac{1}{5}}\). In addition, if the distribution is not unimodal, we need set \(\hat \sigma=min(S,\frac{IQR}{1.34})\).

Explain MLE

The formula for Gaussian Distribution \[f(x)=\frac{1}{\sqrt{2\pi \sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}\] \[L(f(x))=(\frac{1}{\sqrt{2\pi \sigma^2}})^n e^{\sum_{i=1}^n -\frac{(x_i-\mu)^2}{2\sigma^2}}\] \[ln(L(f(x)))=-\frac{n}{2}(ln{(2\pi)}+ln(\sigma^2)) - {\sum_{i=1}^n \frac{(x_i -\mu)^2}{2\sigma^2}}\] \[\frac{\partial ln(L)}{\partial \mu}=\frac{1}{\sigma^2}\sum_{i=1}^n (x_i -\mu)=0\] \[\frac{\partial ln(L)}{\partial \sigma^2}=-\frac{n}{2\sigma^2}+\frac{1}{2\sigma^4}\sum_{i=1}^n (x_i -\mu)^2=0\]

\[\mu=\frac{1}{n}\sum_{i=1}^n x_i = \overline x\] \[\sigma^2=\frac{1}{n}\sum_{i=1}^n (x_i-\mu)^2\] The Maximum Likelihood Estimation method to estimate the value of the most likely parameter by finding the maximum likelihood. For calculation, you need product the all observed sample. After defining the Likelihood Function, we take the logarithm of likelihood function. Because the monotonicity of the logarithm does not change the monotonicity of the likelihood formula, and both take the maximum value at the same point. Finally, derive the parameters and set the equation to zero to get the estimated value. ## Shows the Estimation

```
## [1] "Fixed Acidity"
```

```
## [1] "Bootstrap Mean: 8.26960600375235"
```

```
## [1] "Bootstrap var: 3.02195422514541"
```

```
## [1] "Jacknife Mean: 8.31963727329581"
```

```
## [1] "Jacknife Var: 3.03141638899782"
```

```
## [1] "Sample Mean: 8.31963727329581"
```

```
## [1] "Sample Var: 3.03141638899782"
```

```
## [1] "MLE Mean: 8.32017966701533"
```

```
## [1] "MLE Var: 3.03254901797268"
```

```
## [1] "Cirtic Acdity"
```

```
## [1] "Bootstrap Mean: 0.273652282676673"
```

```
## [1] "Bootstrap var: 0.0392374660007311"
```

```
## [1] "Jacknife Mean: 0.270975609756098"
```

```
## [1] "Jacknife Var: 0.0379474831344058"
```

```
## [1] "Sample Mean: 0.270975609756098"
```

```
## [1] "Sample Var: 0.0379474831344058"
```

```
## [1] "MLE Mean: 0.270966369940073"
```

```
## [1] "MLE Var: 0.0379291736811019"
```

```
## [1] "Cirtic Acdity"
```

```
## [1] "Bootstrap Mean: 0.273652282676673"
```

```
## [1] "Bootstrap var: 0.0392374660007311"
```

```
## [1] "Jacknife Mean: 0.270975609756098"
```

```
## [1] "Jacknife Var: 0.0379474831344058"
## [1] "Sample Mean: 0.270975609756098"
```

```
## [1] "Sample Var: 0.0379474831344058"
```

```
## [1] "EM outputs"
```

```
## $mu
## [1] 0.02664168 0.19459247 0.42916942
## 
## $var
## [1] 0.0008544612 0.0073184049 0.0187964911
## 
## $prob
## [1] 0.2131746 0.3085773 0.4782481
```

```
## [1] "EM Mean 0.270975609756098"
```

```
## [1] "EM Var 0.0379248076881828"
```

```
## [1] "Density"
```

```
## [1] "Bootstrap Mean: 0.996700781738587"
```

```
## [1] "Bootstrap var: 3.49810007686281e-06"
```

```
## [1] "Jacknife Mean: 0.996746679174484"
```

```
## [1] "Jacknife Var: 3.56202945332698e-06"
```

```
## [1] "Sample Mean: 0.996746679174484"
```

```
## [1] "Sample Var: 3.56202945332698e-06"
```

```
## [1] "MLE Mean: 0.996746673199279"
```

```
## [1] "MLE Var: 3.5606124418882e-06"
```

```
## [1] "Residual Sugar"
```

```
## [1] "Bootstrap Mean: 2.53589743589744"
```

```
## [1] "Bootstrap var: 2.129345736658"
```

```
## [1] "Jacknife Mean: 2.53880550343965"
```

```
## [1] "Jacknife Var: 1.98789713298596"
```

```
## [1] "Sample Mean: 2.53880550343965"
```

```
## [1] "Sample Var: 1.98789713298596"
```

```
## [1] "MLE Mean: 2.53896975615089"
```

```
## [1] "MLE Var: 1.98693607002443"
```

```
## [1] "Alcohol"
```

```
## [1] "Bootstrap Mean: 10.415749426725"
```

```
## [1] "Bootstrap var: 1.08912271536871"
```

```
## [1] "Jacknife Mean: 10.4229831144465"
## [1] "Jacknife Var: 1.13564739500047"
```

```
## [1] "Sample Mean: 10.4229831144465"
```

```
## [1] "Sample Var: 1.13564739500047"
```

```
## [1] "MLE Mean: 10.4230233509144"
```

```
## [1] "MLE Var: 1.13408579497747"
```

```
## [1] "Volatile Acidity"
```

```
## [1] "Bootstrap Mean: 0.522983114446529"
```

```
## [1] "Bootstrap var: 0.032133335759756"
```

```
## [1] "Jacknife Mean: 0.527820512820513"
```

```
## [1] "Jacknife Var: 0.0320623776515516"
```

```
## [1] "Sample Mean: 0.527820512820513"
```

```
## [1] "Sample Var: 0.0320623776515516"
```

```
## [1] "MLE Mean: 0.52781629613742"
```

```
## [1] "MLE Var: 0.0320405111771588"
```

```
## [1] "pH"
```

```
## [1] "Bootstrap Mean: 3.31772357723577"
```

```
## [1] "Bootstrap var: 0.0236424992622892"
```

```
## [1] "Jacknife Mean: 3.31111319574734"
```

```
## [1] "Jacknife Var: 0.0238351805454128"
```

```
## [1] "Sample Mean: 3.31111319574734"
```

```
## [1] "Sample Var: 0.0238351805454128"
```

```
## [1] "MLE Mean: 3.31107909416666"
```

```
## [1] "MLE Var: 0.0238015994138302"
```

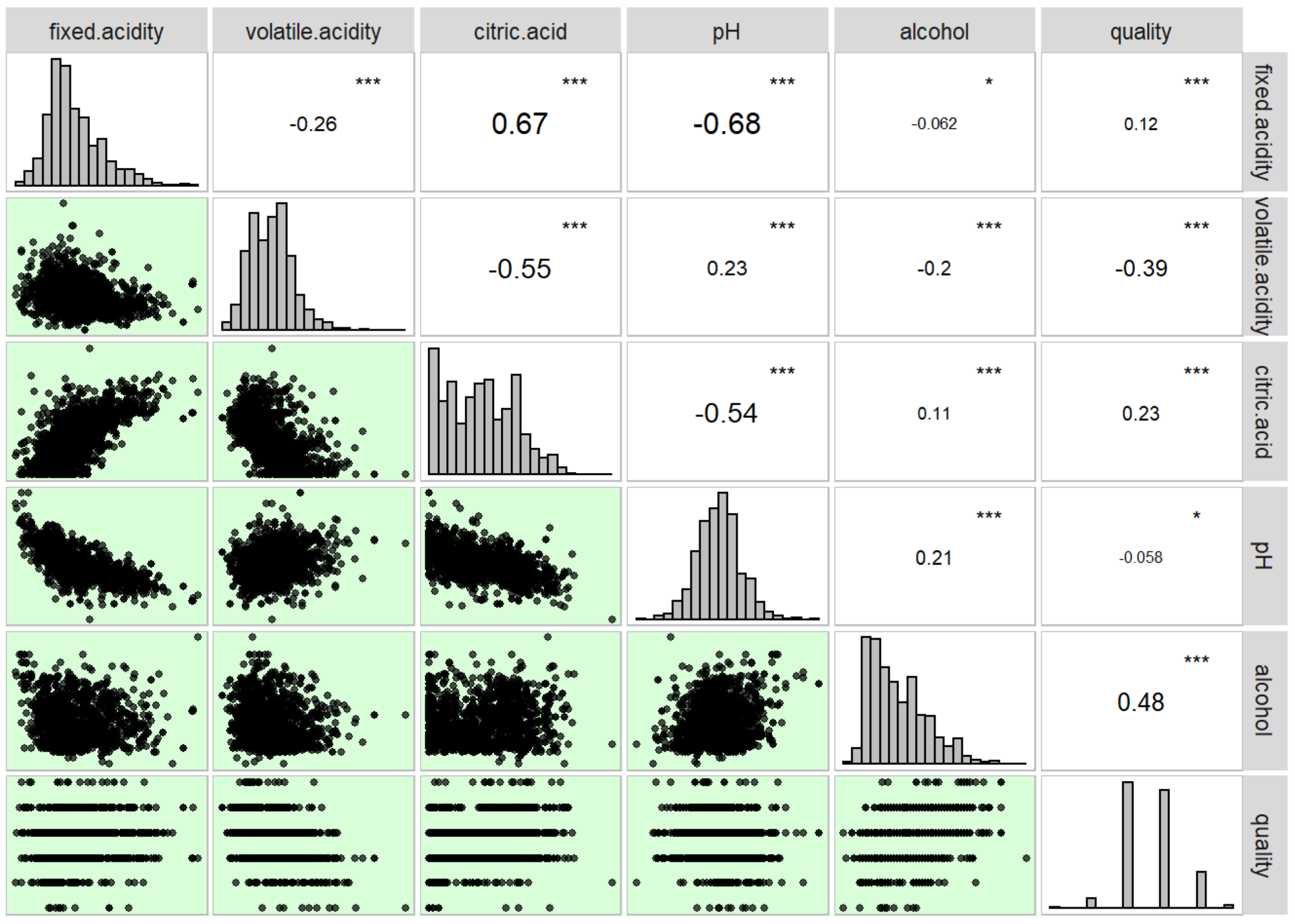
Modeling

Step function for a glance

After we have basic knowledge on these data, we will try to build model base on these data. We firstly conduct the easiest way for linear regression, which is using step function. We totally apply three method for the step function for linear regression: forward step, backward step and both step. The difference between these methods is how the variables are added to the regression model. From these result we can notice that for the quality as y, other variables as x. The more important variables are fixed.acidity, volatile.acidity, citric.acid, pH and the alcohol, which will also be the focus of our analysis later.

Basic analysis for a glance

After we select the variables we interested. We draw the scatter matrix for better viewing the correlation coefficient values and scatter plots between two variables to explore more information between these data.



Besides, we use the powerTransform function to detect whether the variable is suitable for logarithms.

```
## Estimated transformation parameter
## fixed.acidity
##    -0.6686188
```

```
## Estimated transformation parameter
## volatile.acidity
##        0.3996503
```

```
## Estimated transformation parameter
## citric.acid
##    0.444153
```

```
## Estimated transformation parameter
##          pH
## -0.02910134
```

```
## Estimated transformation parameter
##   alcohol
## -3.326066
```

The variable which is close to zero, the log-transformtion is suggested. And the variable which is about 0.5, then the corresponding variable would be replaced by its square root. These result can be very useful when we select the variable for the model by hand.

Select the model by jackknife

Then we will select the model and using the jackknife to find the suitable model. The workflow is to mask one of the data one by one in order. We just select one of the variables each search. For each mask, we use the remaining data for building model. And then use the data being masked to make prediction, and get the error of it.The best variable with the best model has the lowest error square mean.

We totally using two ways to find the variable with corresponding coefficient.

Grid search

We firstly apply most simple but time consuming method to obtain the best model, which is searching one by one with the following form.

1. Linear: \(Y = \beta_0 + \beta_1 X +\varepsilon\)
2. Quadratic: \(Y = \beta_0 + \beta_1 X + \beta_1 X^2 +\varepsilon\)
3. Square root: \(Y = \beta_0 + \beta_1 \sqrt{X}+\varepsilon\)

And the variable which have the smallest error when using the jackknife method to build the corresponding model and make the prediction for the data. After we get the best one, we continuing adding new variable to conduct another operation to find the next variables and the model. The one of step ranking is shown in here, 1 means the linear form, 2 means the quadratic form and 3 means the square root form.

```
##             X_name   E2_mean E_type_list
## 5          alcohol 0.5049280           3
## 2 volatile.acidity 0.5520683           3
## 3      citric.acid 0.6200631           1
## 1    fixed.acidity 0.6434412           1
## 4               pH 0.6513262           1
```

Thus, we can finally get the function which is looks like as follows:

\[Y = \sqrt{alcohol} + volatile.acidity + fixed.acidity + fixed.acidity^2 + citric.acid + pH \]

Select by hand

After we went through the previously obtained information, including the results of powerTransform, correlation coefficient plot, and scatter plot between two two variables, we manually selected the model by experience and finally obtained the following model.

\[Y = \sqrt{alcohol} + volatile.acidity + \sqrt{fixed.acidity} + \sqrt{citric.acid} + log(pH) \]

Final comparison and the view of performance

We make the final comparison for the model we select by hand and the model we select by grid search

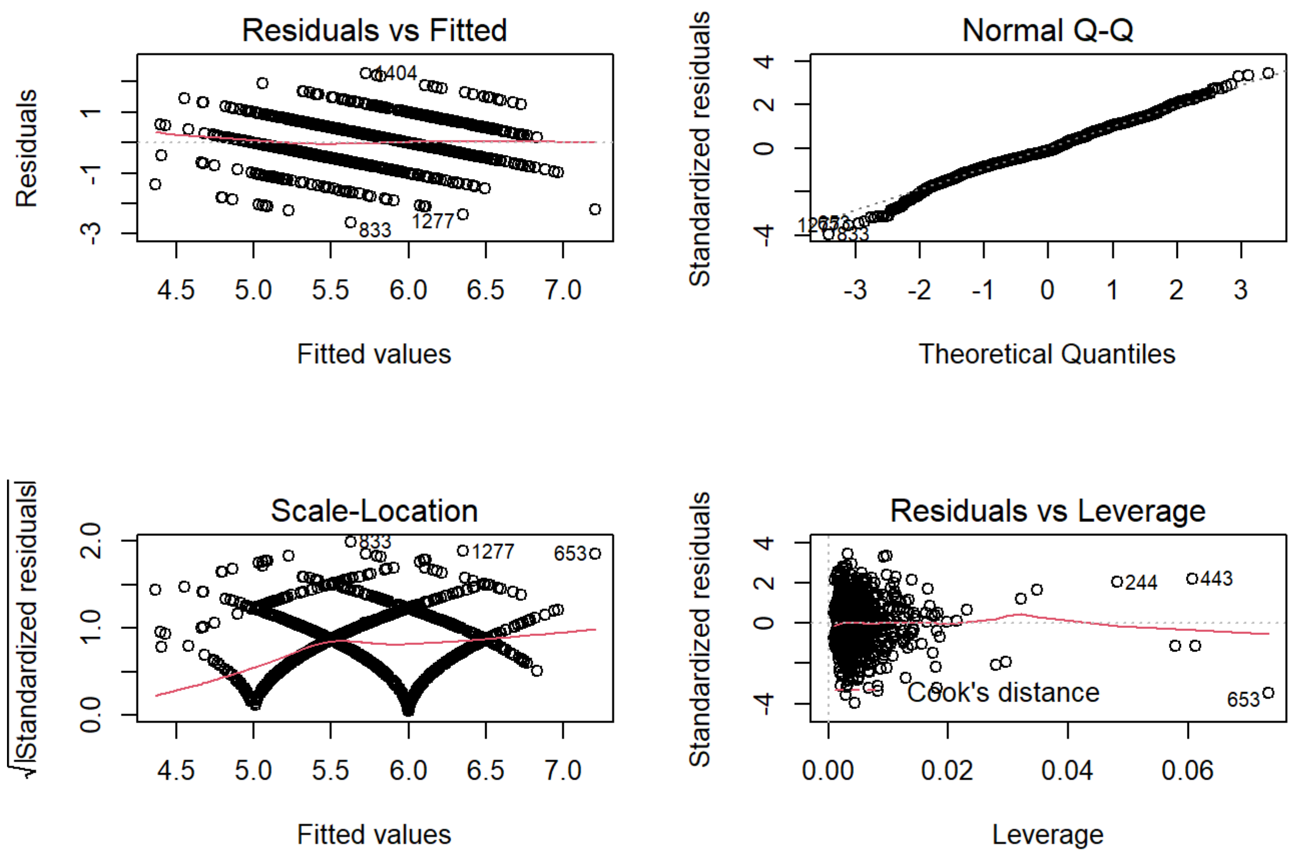
```
## [1] "Model from gird search"
```

```
## [1] 0.4421088
```

```
## [1] "Model select by hand"
```

```
## [1] 0.44253
```

Thus, from the result we can notice that the model gain from the grid search is better than the model we selected by hand. Now we will make analysis for this model.



We then draw the four graph to have a better view of the performance of the model. We can see that the QQ plot is mostly normal distribution, and the line is almost fit all the point. And the residuals are acceptable since the quality is a qualitative variables which use the number to represent it

Get the bias and standard error of the estimator

Then we conduct the bootstrap method to get the bias and standard error of the estimator. We use the sample as the original value of the estimator. And we use the bootstrap to generate a lot of samples to build model by lm function and get new estimator. The result is listing here.

```
## [1] "Bias coefficient mean:"
```

```
##           (Intercept)      sqrt(data$alcohol)   data$volatile.acidity
##          -0.955722538            -0.036399877            -0.006057765
##    data$fixed.acidity I(data$fixed.acidity^2)        data$citric.acid
##           0.036694673            -0.001305644             0.033851395
##               data$pH
##           0.258478893
## [1] "Bias coefficient standard deviation:"
```

```
##             (Intercept)       sqrt(data$alcohol)   data$volatile.acidity 
##             0.718303147              0.117070232              0.132422456 
##      data$fixed.acidity I(data$fixed.acidity^2)        data$citric.acid 
##             0.095187219              0.005288444              0.137693450 
##                 data$pH 
##             0.108792868
```

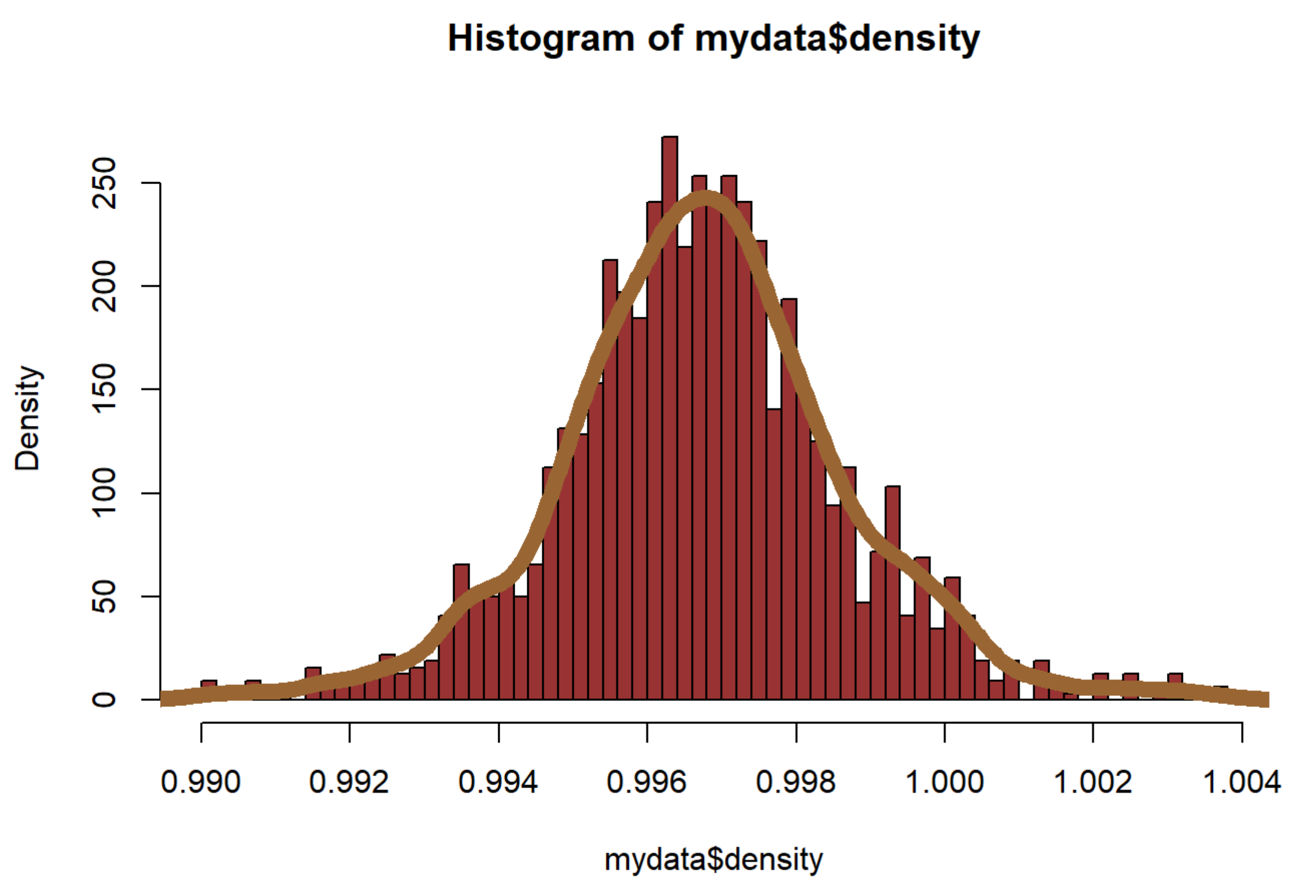
From the results, we can see that the values of some Bias coefficient mean are satisfied, but some of them are still relatively high, so there is still room for improvement of this model. We can optimize this model by trying more combinations, and this is also the direction we can further improve in the future.

Simulation

```
mydata<-read.csv("winequality-red.csv")
```

Because the “density” features is a significant evaluations for the quality of the red wine. So we used it to do the simulation. We drew the histogram of the “density” feature.

```
hist(mydata$density,freq=FALSE,breaks=50,col="#993333",prob=TRUE)
lines(density(mydata$density),col="#996633",lwd=8)
```



we can see that,①The peak is in the middle. ②Both sides gradually decreasing ③The two sides are nearly symmetrical④The ends do not intersect with the x-axis，so it follow the normal distribution. for the ML_function, we can get the mean and variance:

Firstly, we used the optimized method to get the mean and variance: For the big sample:

```
#mean and standard variance
LL <- function(theta,x) {
    mu<-theta[1]
    var<-theta[2]
    return (-(-(n/2)*log(2*pi)-(n/2)*log(var)-(1/(2*var))*sum((x-mu)^2)))
}
optim(c(1,1), LL,x=mydata$density)
```

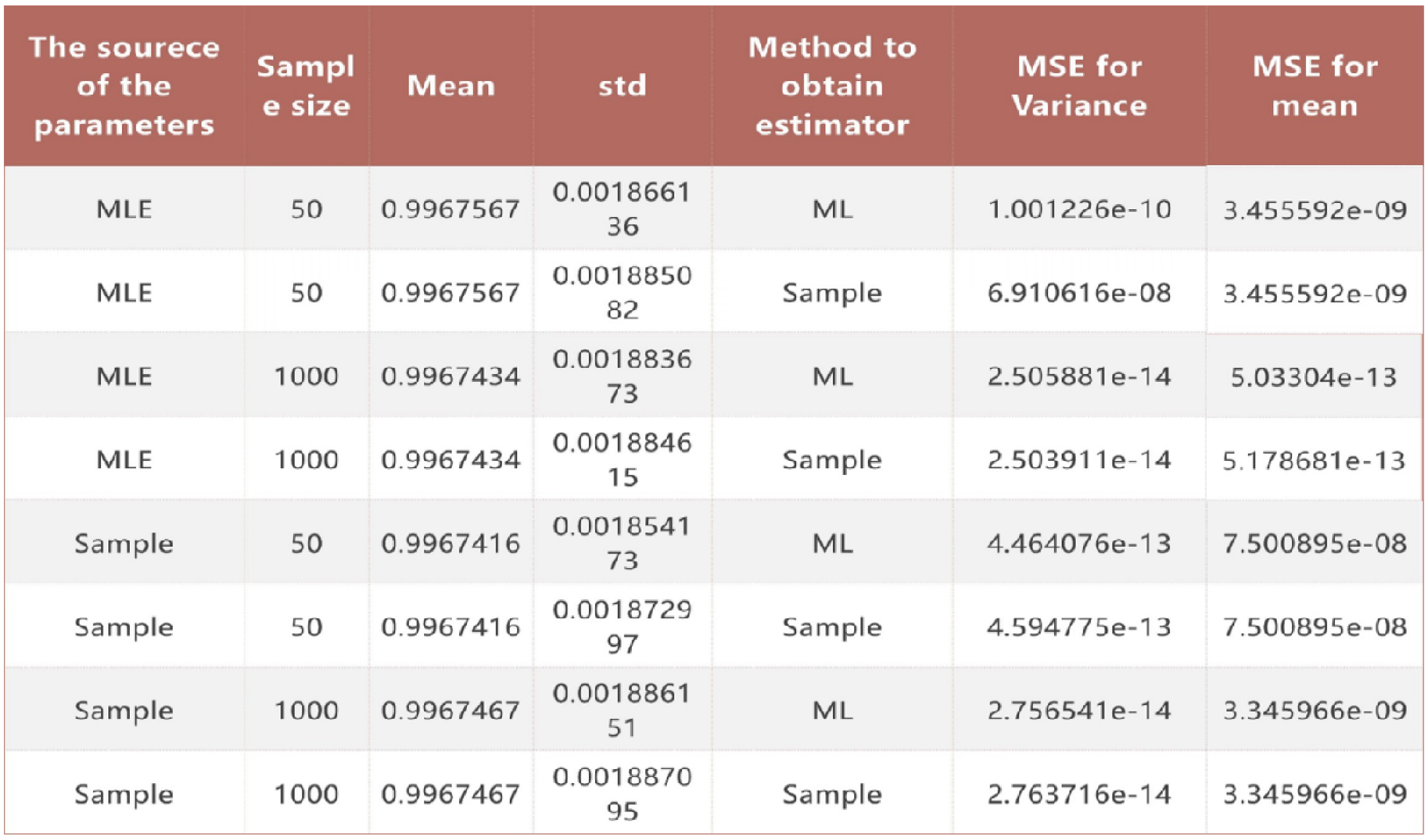
The value methods is used to calculate the the population mean and variance in different method(MLE/Sample):

```
values<-function(sample){
m<-mean(sample)
s_l<-sum(((sample)-mean(sample))^2)/(length(sample))#ml
s_s<-sum(((sample)-mean(sample))^2)/(length(sample)-1)
list(mu = m, s_mle = s_l, s_sample = s_s)
}
values(mydata$density)
```

We write a method to help us calculate different evaluations.The method is used to calculate the Monte Carlo the MEAN/STD/MSE:

```
evaluation<-function(sample,mean,vars){
  #"sample" is the size of the simulation sample
  #"mean" is the the mean of different methods mean value
  #"vars" is the the variance of different methods variance value
df <- matrix(NA,sample,1)
df <- as.data.frame(df)
n=400
for (i in 1:n) {
  df[,i] <- rnorm(sample,mean,sqrt(vars))
}
df2 <- data.frame(mean=NA,ss2=NA)
df3 <- data.frame(mean=NA,ss2=NA)
for (i in 1:n){
  t1 <- mean(df[,i])
  t2 <- sum(((df[,i])-mean(df[,i]))^2)/length(df[,i])#ml method
  df2[i,] <- c(t1,t2)
  t3 <- mean(df[,i])
  t4 <- sum(((df[,i])-mean(df[,i]))^2)/(length(df[,i])-1)#sample method
  df3[i,] <- c(t3,t4)
}
#ml
msef<-data.frame(meanv1=NA,sv1=NA)
n=400#the repeat time
for(u in 1:n){
  mse1<-(df2[u,2]-(mean(df2$ss2)))^2
  mv1<-(df2[u,1]-(mean(df2$mean)))^2
  msef[u,]<-c(mv1,mse1)
}
res_s=mean(msef$sv1)+(mean(df2$ss2)-vars)^2
res_m=mean(msef$meanv1)+(mean(df2$mean)-mean)^2
#sample
msef1<-data.frame(meanv2=NA,sv2=NA)
n=400
for(u in 1:n){
  mse1<-(df3[u,2]-(sum(df3$ss2)/400))^2
  mv1<-(df3[u,1]-(sum(df3$mean)/400))^2
  msef1[u,]<-c(mv1,mse1)
}
c<-sum(msef1$sv2)/400
res_s1=c+(sum(df3$ss2)/400-vars)^2
d<-sum(msef1$meanv2)/400
res_m1=d+(sum(df3$mean)/400-mean)^2
list(mu_mle = mean(df2$mea), std_mle =mean(df2$ss2) ,m_sample= mean(df3$mean),std_sample=mean(df3$ss2),mse_mle_mean = res_m, ms
e_mle_var =res_s,mse_sample_mean = res_m1, mse_sample_var =res_s1)
}
```

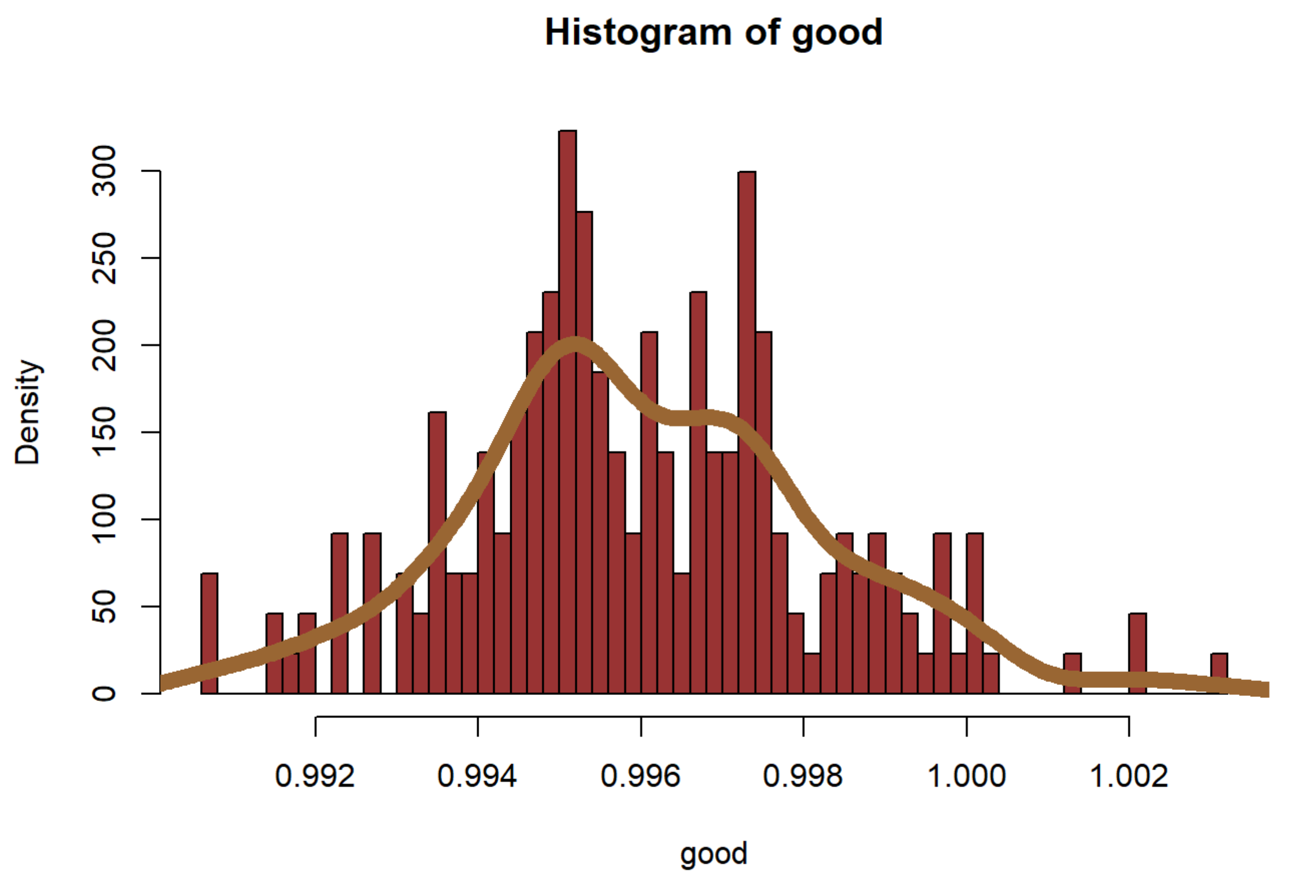
We can use the above function to get the result for the population of “density” of the red wine.



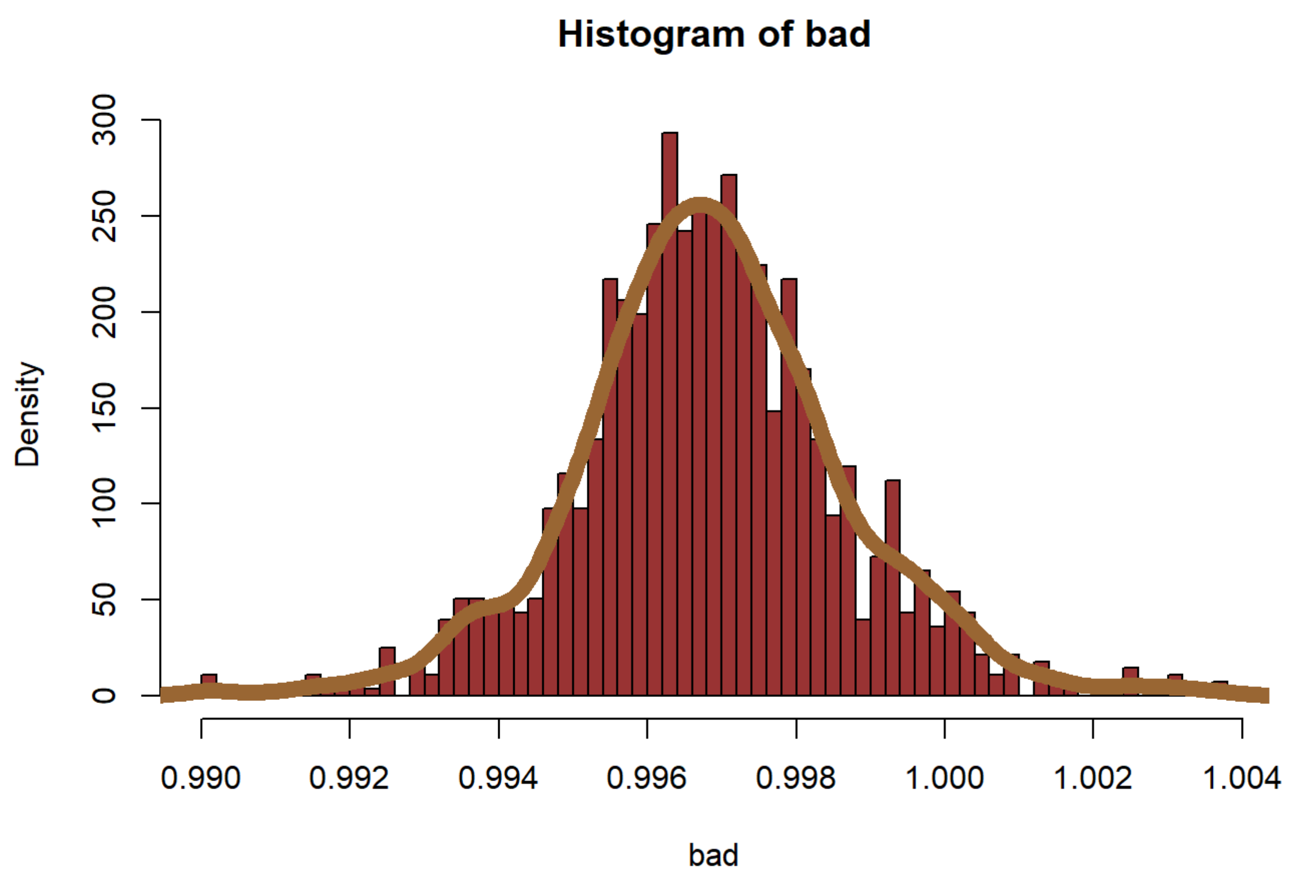
The sourece of the parameters	Sample size	Mean	std	Method to obtain estimator	MSE for Variance	MSE for mean
MLE	50	0.9967567	0.001866136	ML	1.001226e-10	3.455592e-09
MLE	50	0.9967567	0.001885082	Sample	6.910616e-08	3.455592e-09
MLE	1000	0.9967434	0.001883673	ML	2.505881e-14	5.03304e-13
MLE	1000	0.9967434	0.001884615	Sample	2.503911e-14	5.178681e-13
Sample	50	0.9967416	0.001854173	ML	4.464076e-13	7.500895e-08
Sample	50	0.9967416	0.001872997	Sample	4.594775e-13	7.500895e-08
Sample	1000	0.9967467	0.001886151	ML	2.756541e-14	3.345966e-09
Sample	1000	0.9967467	0.001887095	Sample	2.763716e-14	3.345966e-09

Meanwhile, we divided the data set into two part. If the quality is bigger or equal to 7, they can be considered to be good wines. If the quality is smaller than 7, they can be considered to be a bad wines. Also we used this two sample sizes to do the MLE and Sample methods with the Monte Carlo.

```
good<-mydata$density[mydata$quality>=7]
bad<-mydata$density[mydata$quality<7]
#the mixture normal distribution
hist(good,freq=FALSE,breaks=50,col="#993333",prob=TRUE)
lines(density(good),col="#996633",lwd=8)
```



```
#the normal distribution
hist(bad,freq=FALSE,breaks=50,col="#993333",prob=TRUE)
lines(density(bad),col="#996633",lwd=8)
```



```
#for the good wine:
evaluation(50,0.9967467,3.559802e-06)
evaluation(1000,0.9967467,3.559802e-06)
evaluation(50,0.9967467,3.562029e-06)
evaluation(1000,0.9967467,3.562029e-06)
#for the bad wine:
evaluation(50,0.9968592,3.268215e-06)
evaluation(1000,0.9968592,3.268215e-06)
evaluation(50,0.9968592,3.270581e-06)
evaluation(1000,0.9968592,3.270581e-06)
```

The source of parameters	Sample of size	Mean	std	Methods to obtain the estimators	MSE for Variance	MSE for Mean
MLE	50	0.9967451	3.524568e-06	MLE	4.864442e-13	6.192904e-08
MLE	50	0.9967451	3.596498e-06	Sample	5.065557e-13	6.192904e-08
MLE	1000	0.9967473	3.560689e-06	MLE	2.707679e-14	3.180286e-09
MLE	1000	0.9967473	3.564254e-06	Sample	2.715005e-14	3.180286e-09
Sample	50	0.9967552	3.512506e-06	MLE	5.210302e-13	6.922912e-08
Sample	50	0.9967552	3.58419e-06	Sample	5.404511e-13	6.922912e-08
Sample	1000	0.9967488	3.566497e-06	MLE	2.845274e-14	3.668823e-09
Sample	1000	0.9967488	3.570067e-06	Sample	2.855434e-14	3.668823e-09

The source of parameters	Sample of size	Mean	std	Methods to obtain the estimators	MSE for Variance	MSE for Mean
MLE	50	0.9968535	3.160904e-06	MLE	4.480916e-13	6.559702e-08
MLE	50	0.9968535	3.225412e-06	Sample	4.564093e-13	6.559702e-08
MLE	1000	0.9968563	3.253415e-06	MLE	2.06333e-14	3.572689e-09
MLE	1000	0.9968563	3.256671e-06	Sample	2.05884e-14	3.572689e-09
Sample	50	0.9968605	3.216087e-06	MLE	4.363438e-13	6.470986e-08
Sample	50	0.9968605	3.281721e-06	Sample	4.513674e-13	6.470986e-08
Sample	1000	0.9968631	3.269242e-06	MLE	2.093507e-14	3.306573e-09
Sample	1000	0.9968631	3.272514e-06	Sample	2.097894e-14	3.306573e-09

According to the tables, we can see the MSE of the small sample, we can get the MSE of parameters obtained by MLE is small than the MSE obtained by the Sample method. For the mean square error of large samples, the mean square error of parameters obtained by maximum likelihood estimation is larger than that obtained by sample method, which can be obtained by comparing the data in the image. By comparing the mean square error of MLE and sample, we can objectively see that the mean square error of large sample parameters is always less than that of small sample parameters. In other words, the ML method is more reliable than the sample method, and the estimation based on large samples is more reliable than that based on small samples, and the error with the real value is smaller.

5.Conclusion

After doing all the works in there, we can find that “residual_sugar”,“density” and “pH” between the **bad-wine** and **good-wine** just a bit different. But the features of “volatile_acidity” and “citric_acid” are obviously different. So producers can focus on decrease the **volatile_acidity**, because many of the compounds that cause wine faults are already naturally present in wine but at insufficient concentrations to adversely affect it. Also they can increase the **citric_acid** to improve the quality of the red wine.