Name - Sarvesh kumar Mishra DataSet -Winedata set. Introduction -

Have you ever tried to reveal the presence of clusters in a high-dimensional dataset? Indeed this task arises very often. For example, let's imagine that our goal is to understand which factors impact on the ranking of calls in the customer support call center using historical reports. The ranks are in a range from 1 and 5, which means that we have 5 classes. The obvious approach is to build the multi-class classification model in order to reach the goal. We may spend a lot of time trying to build an accurate model that distinguishes well between 5 classes in the target variable. What if after wasting a lot of time, we are still unable to achieve good results? May it happen that we deal with a "not well behaved" class structures? Maybe we lack explanatory factors in the dataset? Of course, it is possible to analyze learning curves in order to understand if the model has a low or high variance of the parameter estimates across samples (bias-variance tradeoff) in order to conclude if more samples or more factors should be added to the dataset. However, there are other complementary approaches that may help revealing "not well behaved" class structures at an early stage of the analysis. This is what we are going to investigate in this notebook.

For our analysis we will use the wine dataset. This dataset contains the results of a chemical analysis of wines grown in the same region in Italy but derived from 3 different cultivars. Our goal will be to reveal the presence of clusters in the wine dataset. In other words, we will check if 3 cultivators are distinguishable in the dataset.

Columns Name- Alcohol Malic_Acid Ash Ash_Alcanity Magnesium Total_Phenols Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity Hue OD280 Proline Customer_Segment

1.Load the required library

```
#Load library

suppressWarnings(library(tidyverse)) #data manipulation
```

```
## -- Attaching packages --------------------------------------------------------------------------------------------- tidyve
rse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.5
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
## -- Conflicts ------------------------------------------------------------------------------------------------ tidyverse_co
nflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
suppressWarnings(library(corrplot)) #correlation
```

```
## corrplot 0.84 loaded
```

```
suppressWarnings(library(gridExtra)) #visualization
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
suppressWarnings(library(GGally)) #plot
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
suppressWarnings(library(dplyr)) # Data Manipulation
suppressWarnings(library(naniar)) #missing data
suppressWarnings(library(Amelia)) # Missing Data: Missings Map
```

```
## Loading required package: Rcpp
```

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.7.5, built: 2018-05-07)
## ## Copyright (C) 2005-2019 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
suppressWarnings(library(ggplot2)) # Visualization
suppressWarnings(library(scales)) # Visualization
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
suppressWarnings(library(caTools)) # Prediction: Splitting Data
suppressWarnings(library(car)) # Prediction: Checking Multicollinearity
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

2.Read Wine data set

```
wine_dataset = read.csv("E:/Git/sarveshmishra1/machinelearning/UnsupervisedLearning/Wine.csv")
```

3.Verify the loaded data

```
head(wine_dataset, n =5)
```

```
##   Alcohol Malic_Acid  Ash Ash_Alcanity Magnesium Total_Phenols Flavanoids
## 1   14.23       1.71 2.43         15.6       127          2.80       3.06
## 2   13.20       1.78 2.14         11.2       100          2.65       2.76
## 3   13.16       2.36 2.67         18.6       101          2.80       3.24
## 4   14.37       1.95 2.50         16.8       113          3.85       3.49
## 5   13.24       2.59 2.87         21.0       118          2.80       2.69
##   Nonflavanoid_Phenols Proanthocyanins Color_Intensity  Hue OD280 Proline
## 1                 0.28            2.29            5.64 1.04  3.92    1065
## 2                 0.26            1.28            4.38 1.05  3.40    1050
## 3                 0.30            2.81            5.68 1.03  3.17    1185
## 4                 0.24            2.18            7.80 0.86  3.45    1480
## 5                 0.39            1.82            4.32 1.04  2.93     735
##   Customer_Segment
## 1                1
## 2                1
## 3                1
## 4                1
## 5                1
```

Remove the customer segment id columns

```
wine_dataset$Customer_Segment <- NULL
ncol(wine_dataset)
```

```
## [1] 13
```

4. Exploratory data Analysis

Verify the structure of Data

```
str(wine_dataset)
```

```
## 'data.frame':    178 obs. of  13 variables:
##  $ Alcohol             : num  14.2 13.2 13.2 14.4 13.2 ...
##  $ Malic_Acid          : num  1.71 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 ...
##  $ Ash                 : num  2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 ...
##  $ Ash_Alcanity        : num  15.6 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 ...
##  $ Magnesium           : int  127 100 101 113 118 112 96 121 97 98 ...
##  $ Total_Phenols       : num  2.8 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 ...
##  $ Flavanoids          : num  3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 ...
##  $ Nonflavanoid_Phenols: num  0.28 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 ...
##  $ Proanthocyanins     : num  2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 ...
##  $ Color_Intensity     : num  5.64 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 ...
##  $ Hue                 : num  1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 ...
##  $ OD280               : num  3.92 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 ...
##  $ Proline             : int  1065 1050 1185 1480 735 1450 1290 1295 1045 1045 ...
```

Summarize the data

```
summary(wine_dataset)
```

```
##     Alcohol       Malic_Acid        Ash         Ash_Alcanity
##  Min.   :11.03   Min.   :0.740   Min.   :1.360   Min.   :10.60
##  1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210   1st Qu.:17.20
##  Median :13.05   Median :1.865   Median :2.360   Median :19.50
##  Mean   :13.00   Mean   :2.336   Mean   :2.367   Mean   :19.49
##  3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558   3rd Qu.:21.50
##  Max.   :14.83   Max.   :5.800   Max.   :3.230   Max.   :30.00
##    Magnesium      Total_Phenols     Flavanoids    Nonflavanoid_Phenols
##  Min.   : 70.00   Min.   :0.980   Min.   :0.340   Min.   :0.1300
##  1st Qu.: 88.00   1st Qu.:1.742   1st Qu.:1.205   1st Qu.:0.2700
##  Median : 98.00   Median :2.355   Median :2.135   Median :0.3400
##  Mean   : 99.74   Mean   :2.295   Mean   :2.029   Mean   :0.3619
##  3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.875   3rd Qu.:0.4375
##  Max.   :162.00   Max.   :3.880   Max.   :5.080   Max.   :0.6600
##  Proanthocyanins Color_Intensity       Hue              OD280
##  Min.   :0.410   Min.   : 1.280   Min.   :0.4800   Min.   :1.270
##  1st Qu.:1.250   1st Qu.: 3.220   1st Qu.:0.7825   1st Qu.:1.938
##  Median :1.555   Median : 4.690   Median :0.9650   Median :2.780
##  Mean   :1.591   Mean   : 5.058   Mean   :0.9574   Mean   :2.612
##  3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.1200   3rd Qu.:3.170
##  Max.   :3.580   Max.   :13.000   Max.   :1.7100   Max.   :4.000
##     Proline
##  Min.   : 278.0
##  1st Qu.: 500.5
##  Median : 673.5
##  Mean   : 746.9
##  3rd Qu.: 985.0
##  Max.   :1680.0
```

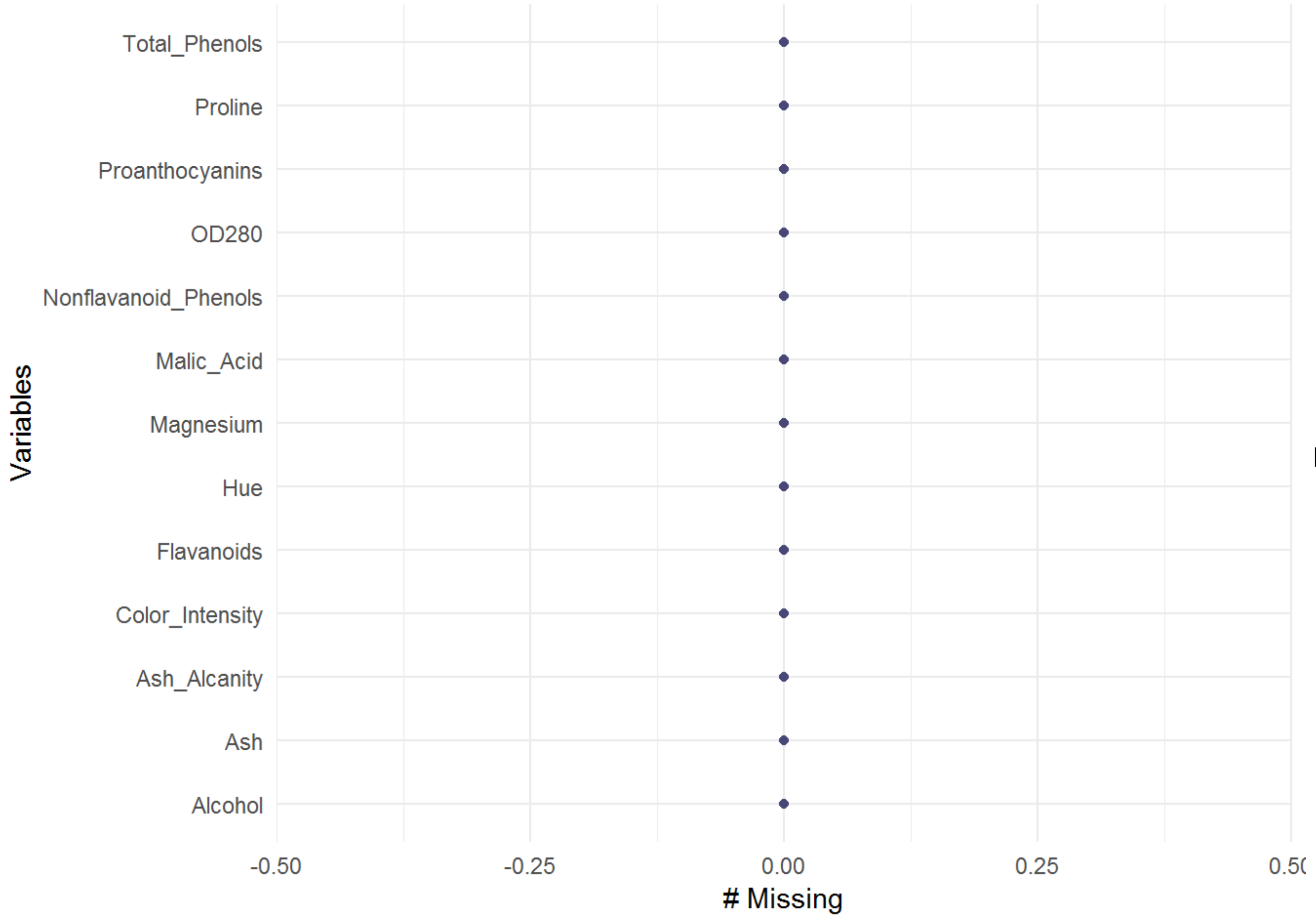
```
glimpse(wine_dataset)
```

```
## Observations: 178
## Variables: 13
## $ Alcohol              <dbl> 14.23, 13.20, 13.16, 14.37, 13.24, 14.20,...
## $ Malic_Acid           <dbl> 1.71, 1.78, 2.36, 1.95, 2.59, 1.76, 1.87,...
## $ Ash                  <dbl> 2.43, 2.14, 2.67, 2.50, 2.87, 2.45, 2.45,...
## $ Ash_Alcanity         <dbl> 15.6, 11.2, 18.6, 16.8, 21.0, 15.2, 14.6,...
## $ Magnesium            <int> 127, 100, 101, 113, 118, 112, 96, 121, 97...
## $ Total_Phenols        <dbl> 2.80, 2.65, 2.80, 3.85, 2.80, 3.27, 2.50,...
## $ Flavanoids           <dbl> 3.06, 2.76, 3.24, 3.49, 2.69, 3.39, 2.52,...
## $ Nonflavanoid_Phenols <dbl> 0.28, 0.26, 0.30, 0.24, 0.39, 0.34, 0.30,...
## $ Proanthocyanins      <dbl> 2.29, 1.28, 2.81, 2.18, 1.82, 1.97, 1.98,...
## $ Color_Intensity      <dbl> 5.64, 4.38, 5.68, 7.80, 4.32, 6.75, 5.25,...
## $ Hue                  <dbl> 1.04, 1.05, 1.03, 0.86, 1.04, 1.05, 1.02,...
## $ OD280                <dbl> 3.92, 3.40, 3.17, 3.45, 2.93, 2.85, 3.58,...
## $ Proline              <int> 1065, 1050, 1185, 1480, 735, 1450, 1290, ...
```

All the column are defined as number or integer.

Check the missing values in column (NA) not the blank values

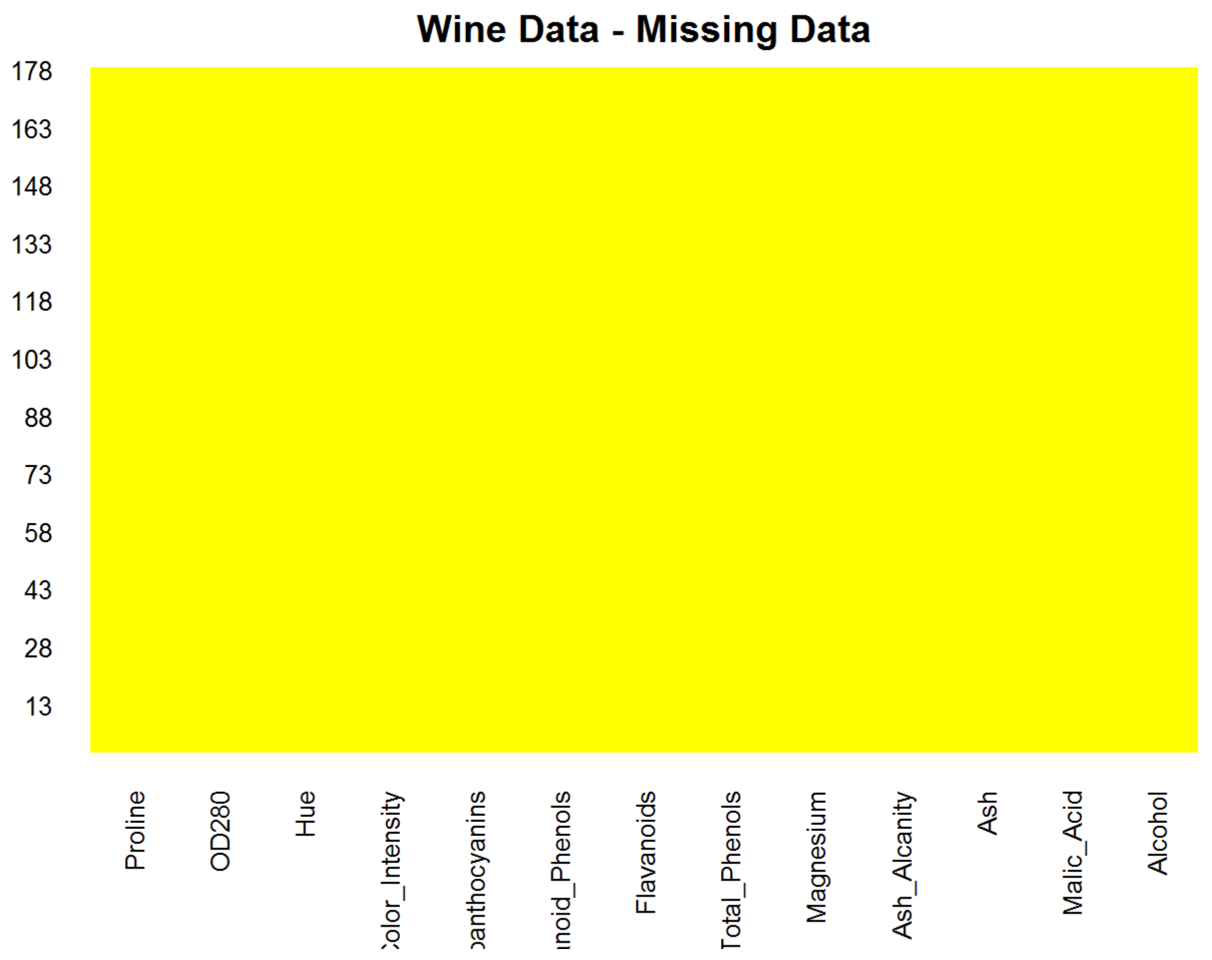
```
gg_miss_var(wine_dataset)
```



No missing values in columns.

missmap allows us to explore how much missing data we have.

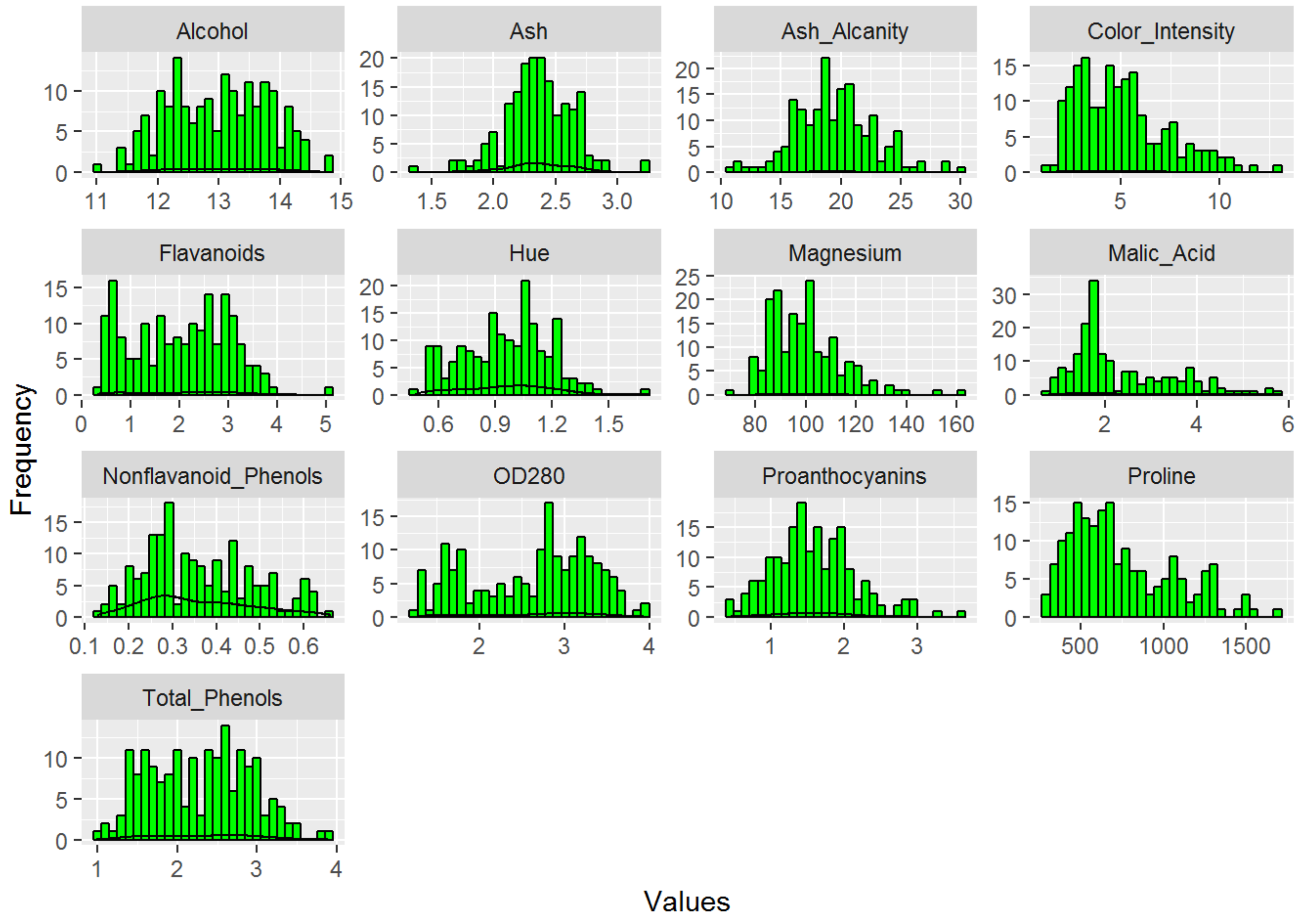
```
missmap(wine_dataset, main = "Wine Data - Missing Data", col = c("Red", "Yellow"), legend=FALSE)
```



Visualization of variables

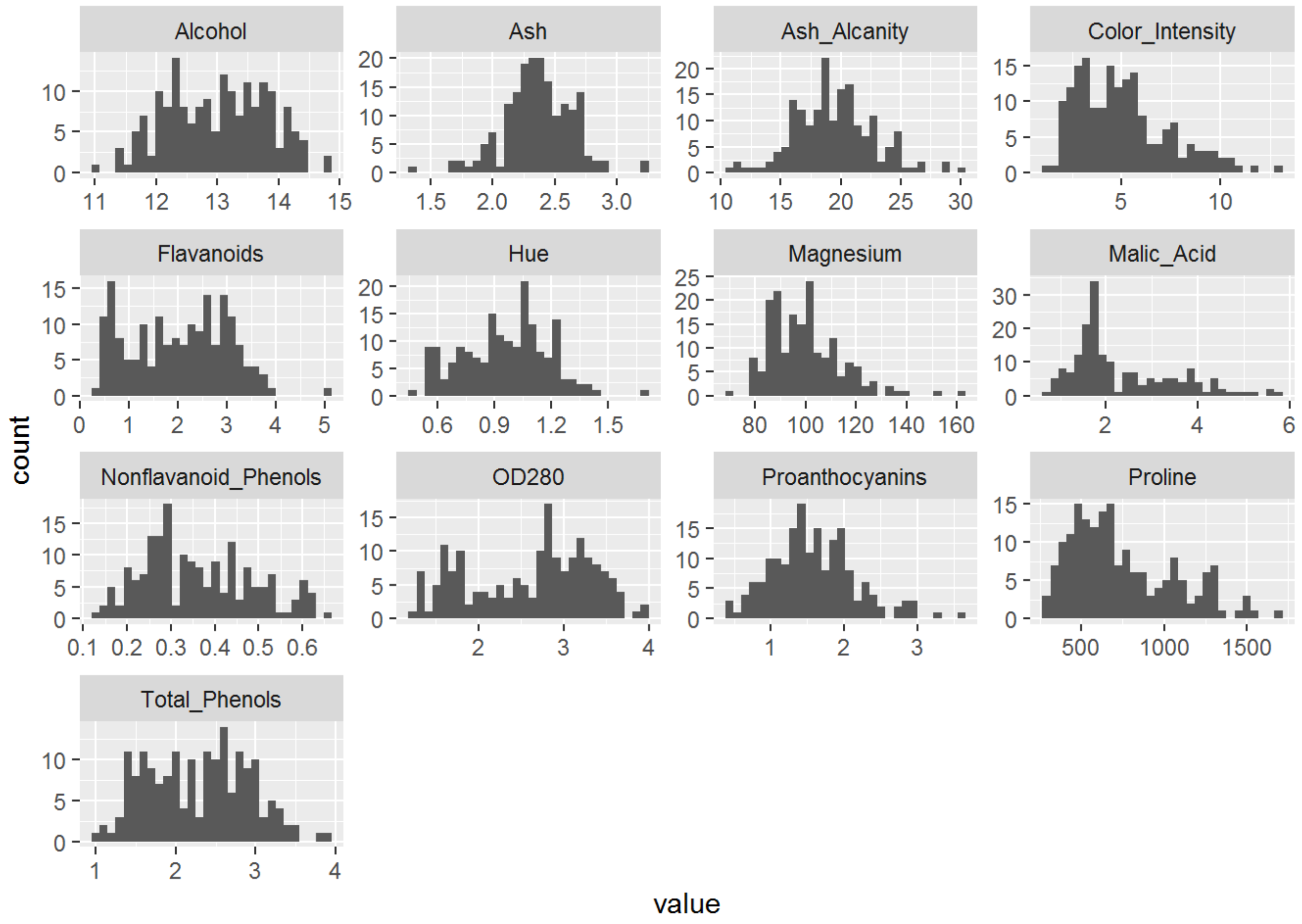
```
wine_dataset %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
  geom_histogram(fill="green", colour="black") +
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density() +
  labs(x="Values", y="Frequency")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
wine_dataset %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
require(reshape2)
```

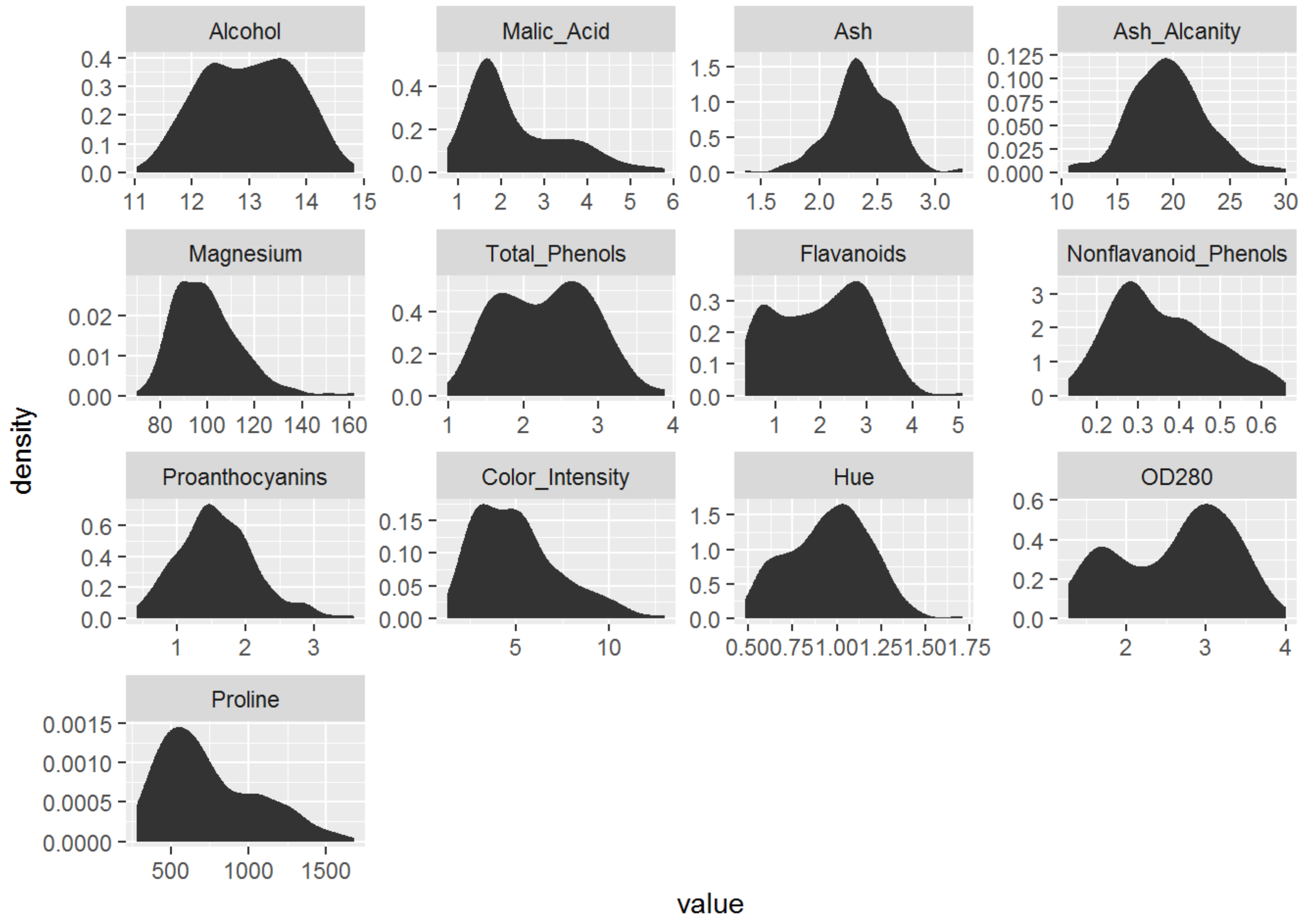
```
## Loading required package: reshape2
```

```
##
## Attaching package: 'reshape2'
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
melt.wine_dataset <- melt(wine_dataset)
```

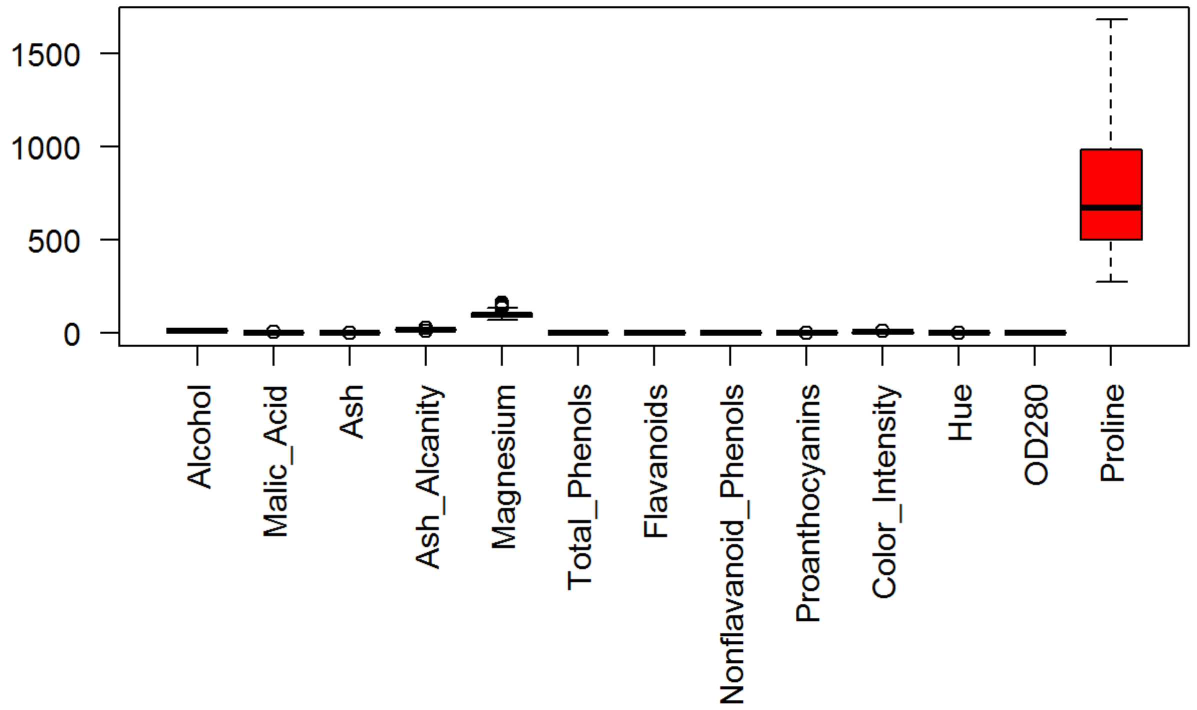
```
## No id variables; using all as measure variables
```

```
ggplot(data = melt.wine_dataset, aes(x = value)) +
stat_density() +
facet_wrap(~variable, scales = "free")
```



From above density we can see, the densities are not normal an presence of outliers. Identify outliers using box plot.

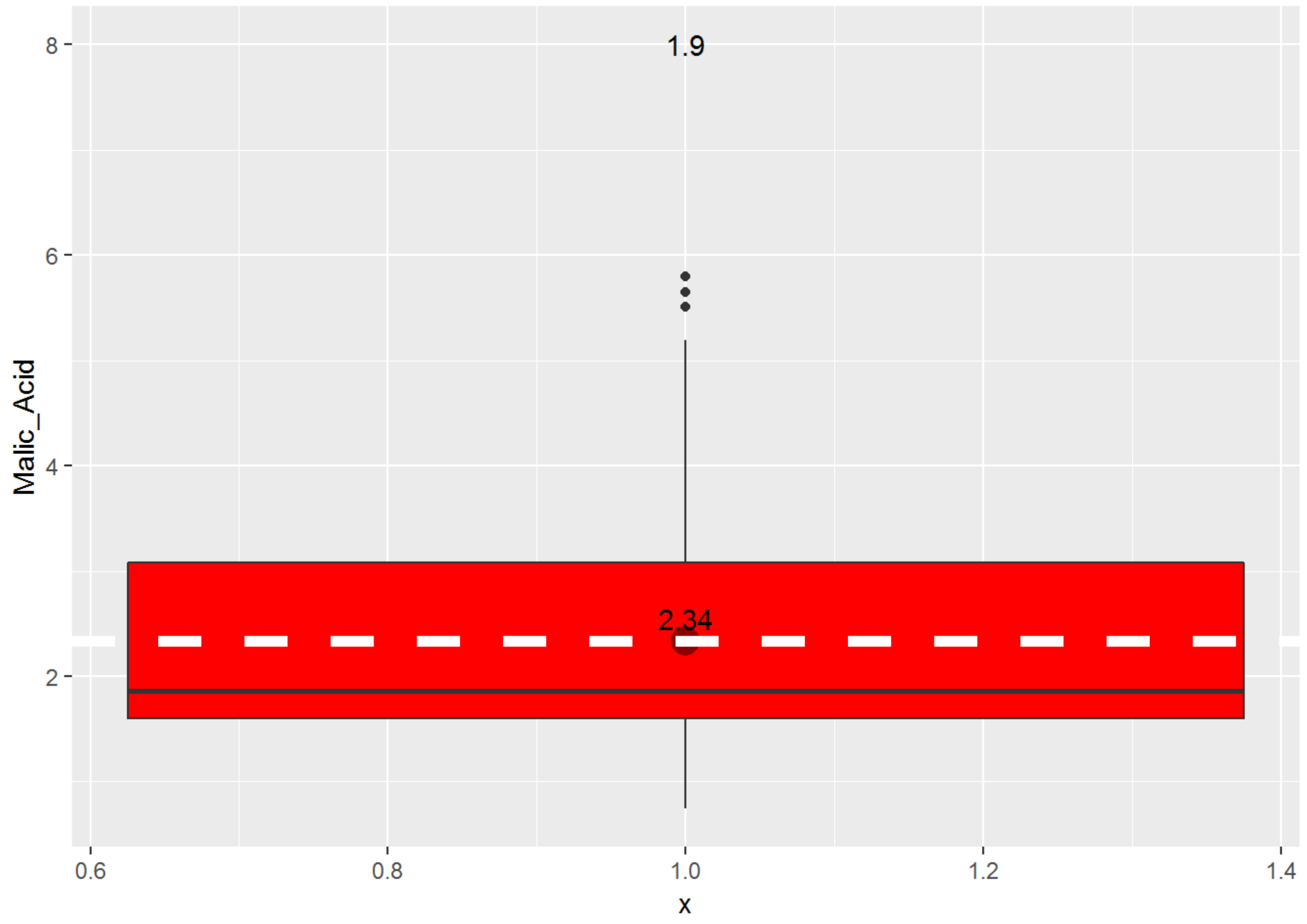
```
boxplot(wine_dataset,
        las = 2,
        col = c("red","sienna","palevioletred1","royalblue2","red","sienna","palevioletred1",
        "royalblue2","red","sienna","palevioletred1","royalblue2"),
        at =c(1,2,3,4, 5,6,7,8,9, 10,11,12,13),
        par(mar = c(12, 5, 4, 2) + 0.1)

        )
```



From above boxplot we can see that features like Malic_Acid, Ash,Ash_Alcanity, Magnesium, Proanthocyanins ,Color_Intensity Hue have outliers.

Lets us create Boxplot, separate for each to understand in details.

```r
calcmed <- function(x) {
   return(c(y = 8, label = round(median(x), 1)))
   # modify 8 to suit your needs
}

fun_mean <- function(x){
  return (round((data.frame(y=mean(x),label=mean(x,na.rm=T))),2))}

#boxplot(wine_dataset[,c(2)], col="red")
ggplot(data=wine_dataset, aes(x=1, y=Malic_Acid)) +
  geom_boxplot(fill="red")+
  #geom_dotplot(binaxis='y', stackdir='center', dotsize=1)+
  stat_summary(fun.data = calcmed, geom = "text") +

  stat_summary(fun.y = mean, geom="point",colour="darkred", size=5) +
  stat_summary(fun.data = fun_mean, geom="text", vjust=-0.5) +
  geom_hline(aes(yintercept=mean(Malic_Acid, na.rm=T)), colour = "white", linetype="dashed", size=2)
```



Find the outliers values for Malic_Acid

```
print("The outlier values for  Malic_Acid are below: ")
```

```
## [1] "The outlier values for  Malic_Acid are below: "
```

```
boxplot(wine_dataset$Malic_Acid, plot=FALSE)$out
```

```
## [1] 5.80 5.51 5.65
```

Outlier treatment - There are 3 ways for outlier treatment 1.Remove the obervation with outlier 2.Imputation - Replace outlier with mean, medain mode. 3.Capping - For missing values that lie outside the 1.5 * IQR limits, we could cap it by replacing those observations outside the lower limit with the value of 5th %ile and those that lie above the upper limit, with the value of 95th %ile.

I have tried both method, Imputation and Capping, Imputation did not removed the outlier, it has added another one. Capping worked perfect, hence we will use capping method for outlier treatment. we will use method 2, that is Impuation to replace outliers.

Outlier treatment for Malic_Acid. Below is the test I performed with mean, but still the outliers

```
wine_dataset_RND <- wine_dataset

#wine_dataset_RND$Malic_Acid[wine_dataset_RND$Malic_Acid  %in% c(5.80, 5.51, 5.65)] <- mean(wine_dataset_RND$Malic_Acid, na.
rm = T)

print("The outlier values for  Malic_Acid are below: ")
```

```
## [1] "The outlier values for  Malic_Acid are below: "
```

```
boxplot(wine_dataset_RND$Malic_Acid, plot=FALSE)$out
```

```
## [1] 5.80 5.51 5.65
```

```
x <- wine_dataset$Malic_Acid
qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
caps <- quantile(x, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]

wine_dataset$Malic_Acid <- x

print("The outlier values for  Malic_Acid are below: ")
```

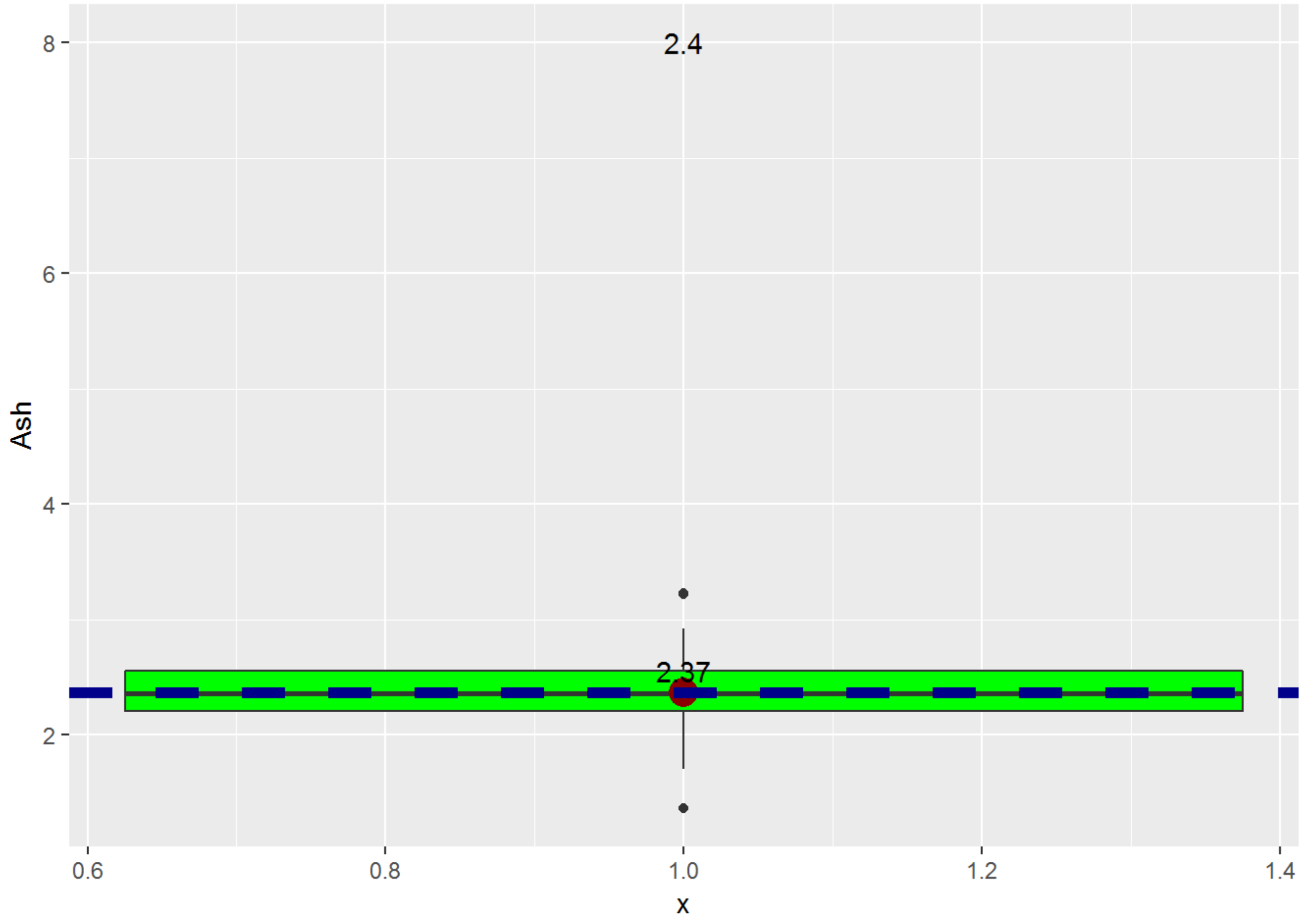
```
## [1] "The outlier values for  Malic_Acid are below: "
```

```
boxplot(wine_dataset$Malic_Acid, plot=FALSE)$out
```

```
## numeric(0)
```

Visualization plot for : Ash

```
#Malic_acid, Ash,Ash_Alcanity, Magnesium, Proanthocyanins ,Color_Intensity
#Hue
calcmed <- function(x) {
   return(c(y = 8, label = round(median(x), 1)))
   # modify 8 to suit your needs
}

fun_mean <- function(x){
  return (round((data.frame(y=mean(x),label=mean(x,na.rm=T))),2))}

ggplot(data=wine_dataset, aes(x=1, y=Ash)) +
  geom_boxplot(fill="green") +
  #geom_dotplot(binaxis='y', stackdir='center', dotsize=1) +
    #geom_dotplot(binaxis='y', stackdir='center', dotsize=1)+
  stat_summary(fun.data = calcmed, geom = "text") +

  stat_summary(fun.y = mean, geom="point",colour="darkred", size=5) +
  stat_summary(fun.data = fun_mean, geom="text", vjust=-0.5) +
  geom_hline(aes(yintercept=mean(Ash, na.rm=T)), colour = "dark blue", linetype="dashed", size=2)
```



Find the outliers values for Ash

```
print("The outlier values for  Ash are below: ")
```

```
## [1] "The outlier values for  Ash are below: "
```

```
boxplot(wine_dataset$Ash, plot=FALSE)$out
```

```
## [1] 3.22 1.36 3.23
```

Outlier treatment for Ash - The mean method did not removed the outliers, hence will use capping method.

```
wine_dataset_RND <- wine_dataset
wine_dataset_RND$Ash[wine_dataset_RND$Ash  %in% c(3.22, 1.36, 3.23)] <- median(wine_dataset_RND$Ash, na.rm = T)

print("The outlier values for  Ash are below: ")
```

```
## [1] "The outlier values for  Ash are below: "
```

```
boxplot(wine_dataset_RND$Ash, plot=FALSE)$out
```

```
## [1] 1.70 1.71 1.70
```

```
x <- wine_dataset$Ash
qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
caps <- quantile(x, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]

wine_dataset$Ash <- x

print("The outlier values for  Ash are below: ")
```

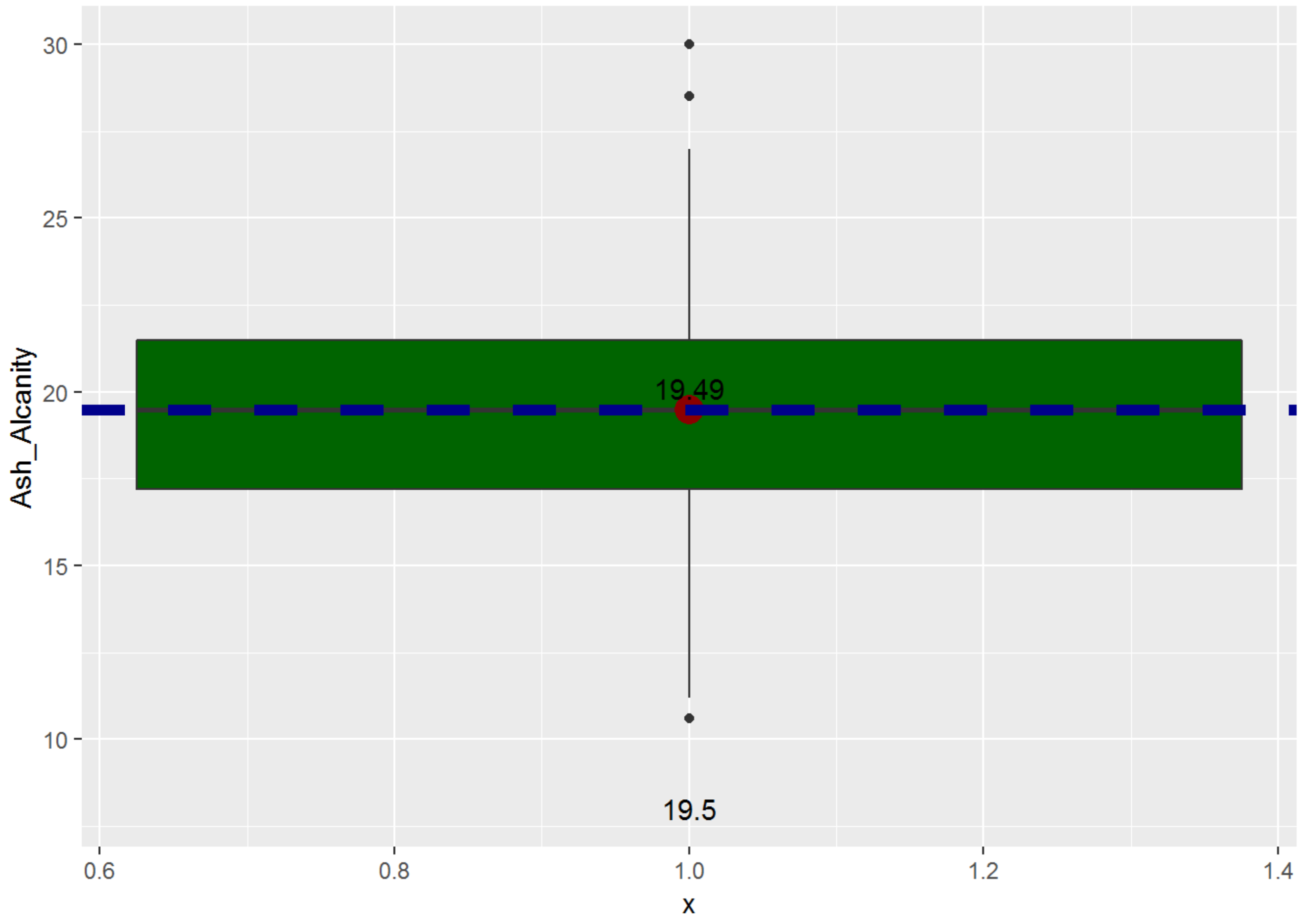
```
## [1] "The outlier values for  Ash are below: "
```

```
boxplot(wine_dataset$Ash, plot=FALSE)$out
```

```
## numeric(0)
```

The visualization plot for Ash_Alcanity.

```
#Malic_acid, Ash,Ash_Alcanity, Magnesium, Proanthocyanins ,Color_Intensity
#Hue

calcmed <- function(x) {
   return(c(y = 8, label = round(median(x), 1)))
   # modify 8 to suit your needs
}

fun_mean <- function(x){
  return (round((data.frame(y=mean(x),label=mean(x,na.rm=T))),2))}

ggplot(data=wine_dataset, aes(x=1, y=Ash_Alcanity)) +
  geom_boxplot(fill="dark green") +
      #geom_dotplot(binaxis='y', stackdir='center', dotsize=1)+
  stat_summary(fun.data = calcmed, geom = "text") +

  stat_summary(fun.y = mean, geom="point",colour="darkred", size=5) +
  stat_summary(fun.data = fun_mean, geom="text", vjust=-0.5) +
  geom_hline(aes(yintercept=mean(Ash_Alcanity, na.rm=T)), colour = "dark blue", linetype="dashed", size=2)
```



Find the outliers values for Ash_Alcanity.

```
print("The outlier values for  Ash_Alcanity. are below: ")
```

```
## [1] "The outlier values for  Ash_Alcanity. are below: "
```

```
boxplot(wine_dataset$Ash_Alcanity, plot=FALSE)$out
```

```
## [1] 10.6 30.0 28.5 28.5
```

Outlier treatment for Ash_Alcanity - For Ash_Alcanity both the method producd the same results and remvoed the outliers, but we will use the capping method as it worked for all the variables.

```
wine_dataset_RND <- wine_dataset

wine_dataset_RND$Ash_Alcanity[wine_dataset_RND$Ash_Alcanity  %in% c(10.6, 30.0, 28.5, 28.5)] <- median(wine_dataset_RND$Ash_
Alcanity, na.rm = T)

print("The outlier values for  Ash_Alcanity are below: ")
```

```
## [1] "The outlier values for  Ash_Alcanity are below: "
```

```
boxplot(wine_dataset_RND$Ash_Alcanity, plot=FALSE)$out
```

```
## numeric(0)
```

```
x <- wine_dataset$Ash_Alcanity
qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
caps <- quantile(x, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]

wine_dataset$Ash_Alcanity <- x

print("The outlier values for  Ash_Alcanity are below: ")
```

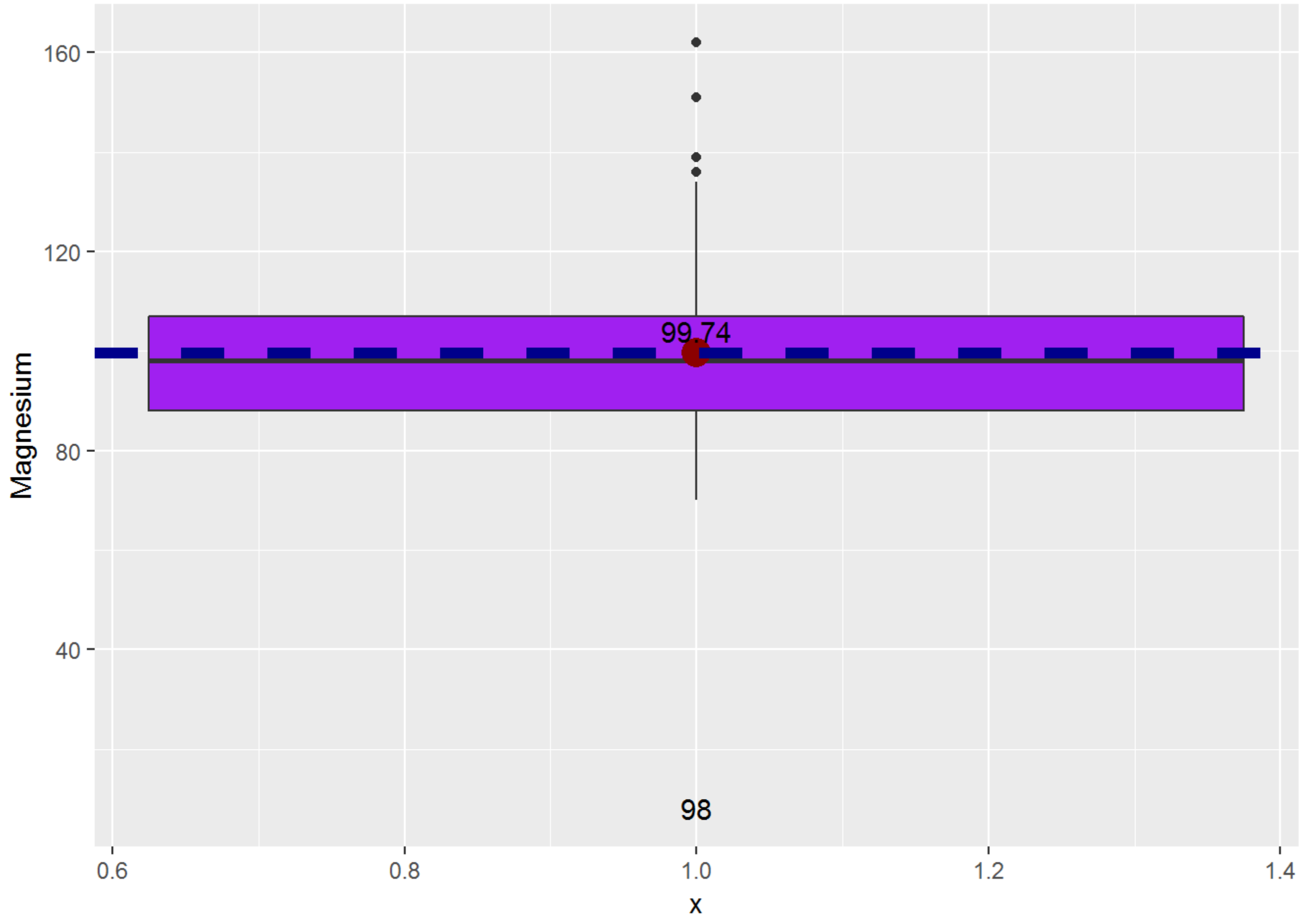
```
## [1] "The outlier values for  Ash_Alcanity are below: "
```

```
boxplot(wine_dataset$Ash_Alcanity, plot=FALSE)$out
```

```
## numeric(0)
```

Visualization plot for Magnesium

```r
#Malic_acid, Ash,Ash_Alcanity, Magnesium, Proanthocyanins ,Color_Intensity
#Hue

calcmed <- function(x) {
   return(c(y = 8, label = round(median(x), 1)))
   # modify 8 to suit your needs
}

fun_mean <- function(x){
  return (round((data.frame(y=mean(x),label=mean(x,na.rm=T))),2))}

ggplot(data=wine_dataset, aes(x=1, y=Magnesium)) +
  geom_boxplot(fill="purple") +
  stat_summary(fun.data = calcmed, geom = "text") +

  stat_summary(fun.y = mean, geom="point",colour="darkred", size=5) +
  stat_summary(fun.data = fun_mean, geom="text", vjust=-0.5) +
  geom_hline(aes(yintercept=mean(Magnesium, na.rm=T)), colour = "dark blue", linetype="dashed", size=2)
```



Find the outliers values for Magnesium

```
print("The outlier values for  Magnesium are below: ")
```

```
## [1] "The outlier values for  Magnesium are below: "
```

```
boxplot(wine_dataset$Magnesium, plot=FALSE)$out
```

```
## [1] 151 139 136 162
```

Outlier treatment for Magnesium - Since the mean and median are almost same, we replaced the outliers with mean and median but that end up addition of new outliers ie 134, we will use capping method here.

```
wine_dataset_RND <- wine_dataset
wine_dataset_RND$Magnesium[wine_dataset_RND$Magnesium  %in% c(151, 139, 136, 162)] <- median(wine_dataset_RND$Magnesium, na.
rm = T)

print("The outlier values for  Magnesium are below: ")
```

```
## [1] "The outlier values for  Magnesium are below: "
```

```
boxplot(wine_dataset_RND$Magnesium, plot=FALSE)$out
```

```
## [1] 134
```

```
x <- wine_dataset$Magnesium
qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
caps <- quantile(x, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]

wine_dataset$Magnesium <- x

print("The outlier values for  Magnesium are below: ")
```

```
## [1] "The outlier values for  Magnesium are below: "
```

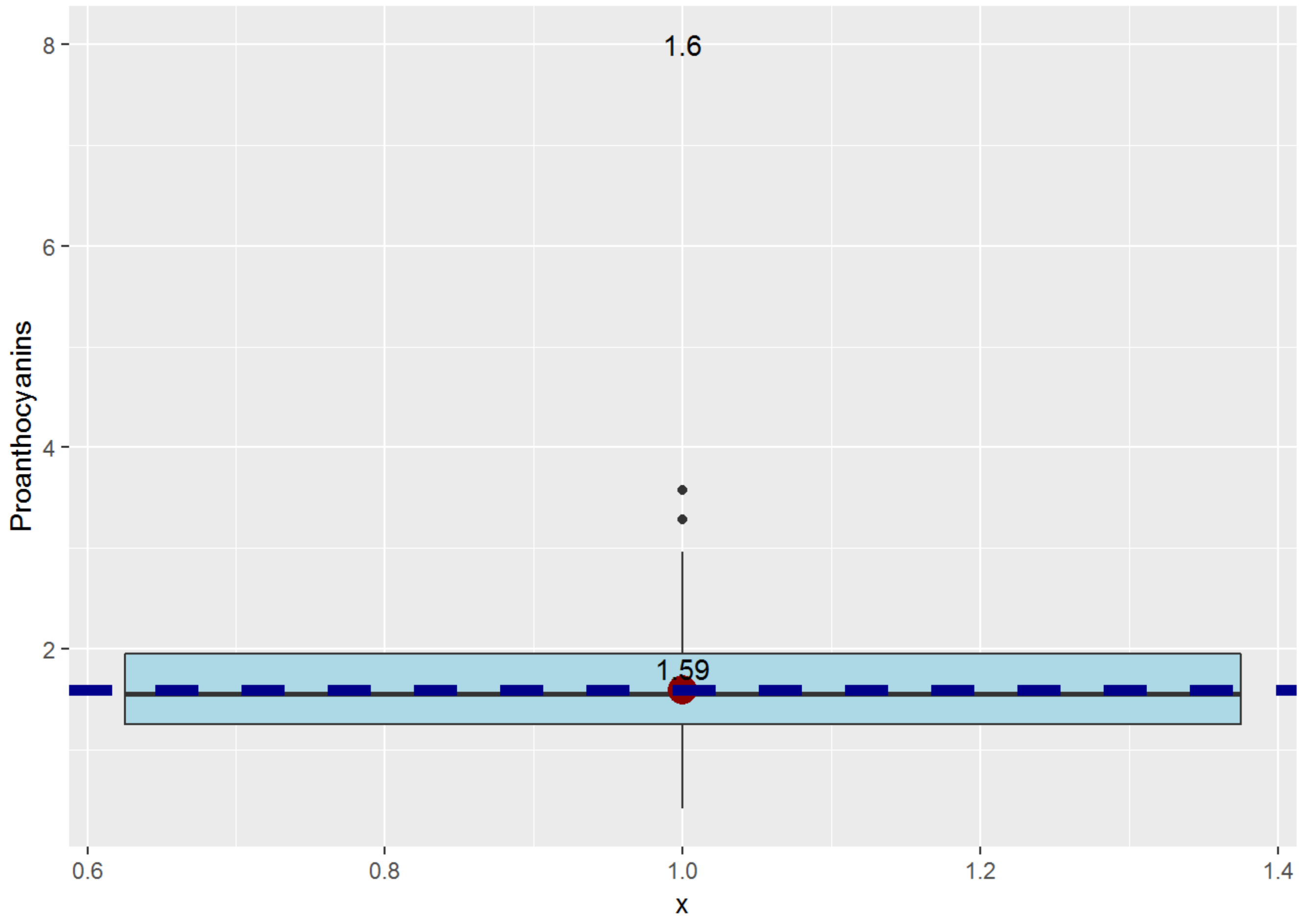
```
boxplot(wine_dataset$Magnesium, plot=FALSE)$out
```

```
## numeric(0)
```

Above we can see caping method able to remove all the outliers and did not added up any new observations.

Visualization for Proanthocyanins

```
#Malic_acid, Ash,Ash_Alcanity, Magnesium, Proanthocyanins ,Color_Intensity
#Hue
calcmed <- function(x) {
   return(c(y = 8, label = round(median(x), 1)))
   # modify 8 to suit your needs
}

fun_mean <- function(x){
  return (round((data.frame(y=mean(x),label=mean(x,na.rm=T))),2))}

ggplot(data=wine_dataset, aes(x=1, y=Proanthocyanins)) +
  geom_boxplot(fill="light blue") +
  stat_summary(fun.data = calcmed, geom = "text") +

  stat_summary(fun.y = mean, geom="point",colour="darkred", size=5) +
  stat_summary(fun.data = fun_mean, geom="text", vjust=-0.5) +
  geom_hline(aes(yintercept=mean(Proanthocyanins, na.rm=T)), colour = "dark blue", linetype="dashed", size=2)
```



Find the outliers values for Proanthocyanins

```
print("The outlier values for  Proanthocyanins are below: ")
```

```
## [1] "The outlier values for  Proanthocyanins are below: "
```

```
boxplot(wine_dataset$Proanthocyanins, plot=FALSE)$out
```

```
## [1] 3.28 3.58
```

Outlier treatment for Proanthocyanins - Since the mean and median are almost same, we can repalce this with either mean or medain. Imputation method able to remove the outliers but we will finally use the capping method.

```
wine_dataset_RND <- wine_dataset
wine_dataset_RND$Proanthocyanins[wine_dataset_RND$Proanthocyanins  %in% c(3.28, 3.58)] <- median(wine_dataset_RND$Proanthocy
anins, na.rm = T)

print("The outlier values for  Proanthocyanins are below: ")
```

```
## [1] "The outlier values for  Proanthocyanins are below: "
```

```
boxplot(wine_dataset_RND$Proanthocyanins, plot=FALSE)$out
```

```
## numeric(0)
```

```
x <- wine_dataset$Proanthocyanins
qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
caps <- quantile(x, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]

wine_dataset$Proanthocyanins <- x

print("The outlier values for  Proanthocyanins are below: ")
```

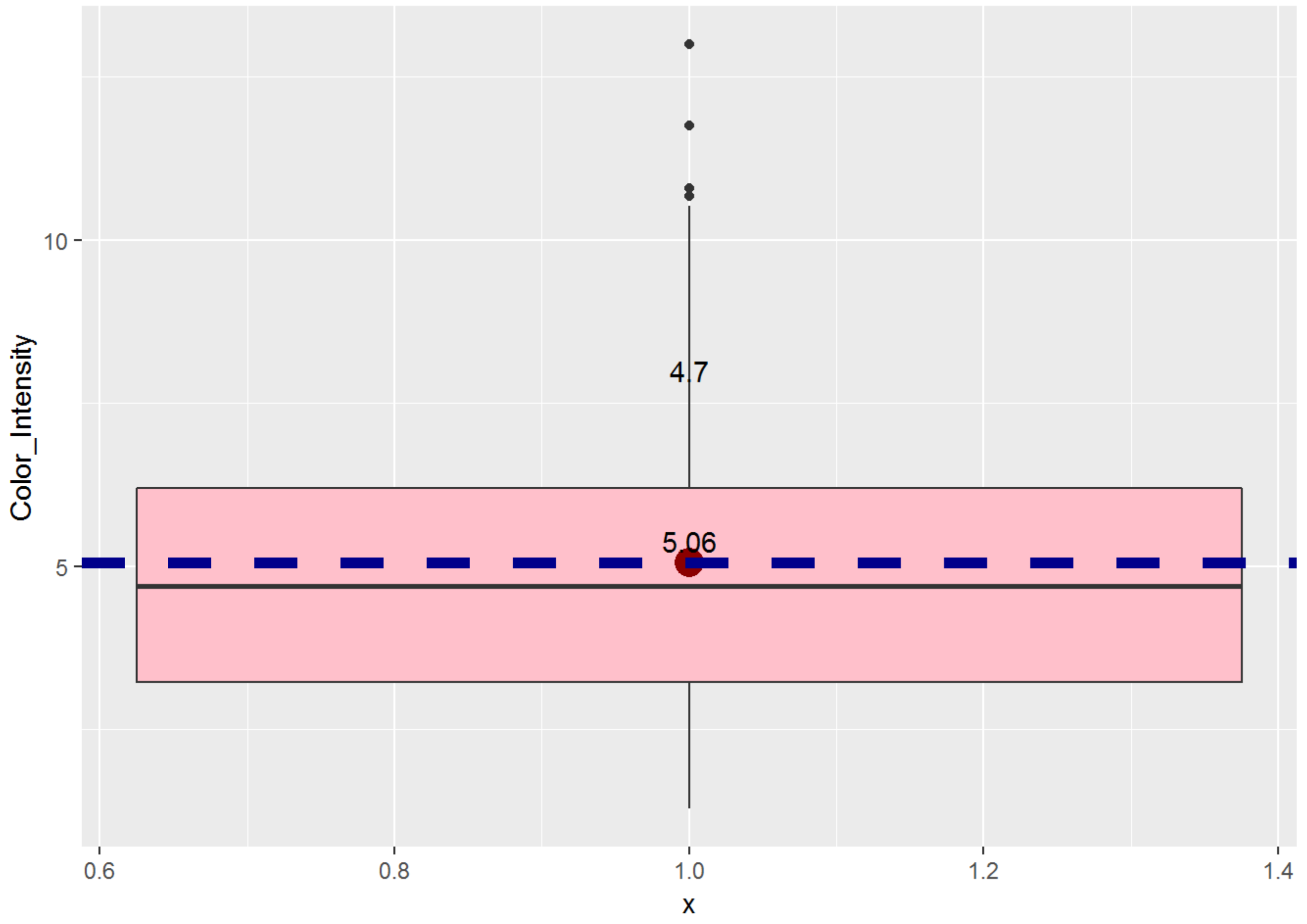
```
## [1] "The outlier values for  Proanthocyanins are below: "
```

```
boxplot(wine_dataset$Proanthocyanins, plot=FALSE)$out
```

```
## numeric(0)
```

Visualization for Color_Intensity

```
#Malic_acid, Ash,Ash_Alcanity, Magnesium, Proanthocyanins ,Color_Intensity
#Hue
calcmed <- function(x) {
   return(c(y = 8, label = round(median(x), 1)))
   # modify 8 to suit your needs
}

fun_mean <- function(x){
  return (round((data.frame(y=mean(x),label=mean(x,na.rm=T))),2))}


ggplot(data=wine_dataset, aes(x=1, y=Color_Intensity)) +
  geom_boxplot(fill="pink") +
    stat_summary(fun.data = calcmed, geom = "text") +

  stat_summary(fun.y = mean, geom="point",colour="darkred", size=5) +
  stat_summary(fun.data = fun_mean, geom="text", vjust=-0.5) +
  geom_hline(aes(yintercept=mean(Color_Intensity, na.rm=T)), colour = "dark blue", linetype="dashed", size=2)
```



Find the outliers values for Color_Intensity

```
print("The outlier values for  Color_Intensity are below: ")
```

```
## [1] "The outlier values for  Color_Intensity are below: "
```

```
boxplot(wine_dataset$Color_Intensity, plot=FALSE)$out
```

```
## [1] 10.80 13.00 11.75
```

Outlier treatment for Color_Intensity - The Imputation method could not able to remove the outliers and added up new one as below. We will continue to go ahead with capping method for outliers treatment.

```
#This section is just to show that mean and median repalcement did not worked.
wine_dataset_RND <- wine_dataset
wine_dataset_RND$Color_Intensity[wine_dataset_RND$Color_Intensity  %in% c(10.80, 13.00, 11.75)] <- median(wine_dataset_RND$C
olor_Intensity, na.rm = T)

print("The outlier values for  Color_Intensity are below: ")
```

```
## [1] "The outlier values for  Color_Intensity are below: "
```

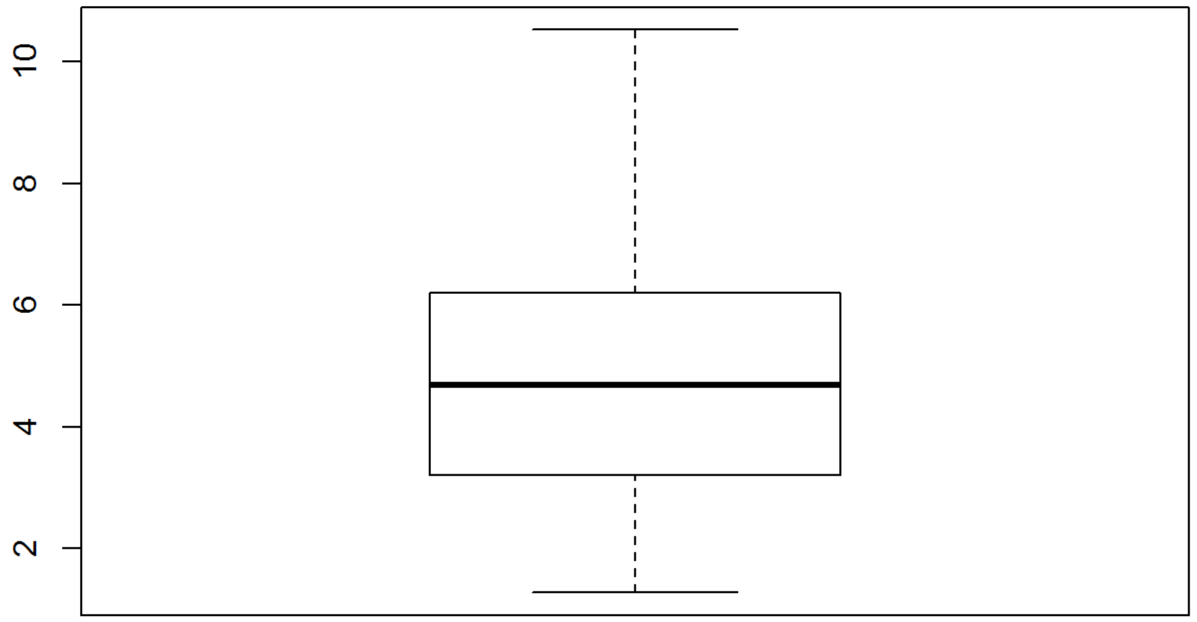
```
boxplot(wine_dataset$Color_Intensity, plot=FALSE)$out
```

```
## [1] 10.80 13.00 11.75
```

```
x <- wine_dataset$Color_Intensity
qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
caps <- quantile(x, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]

wine_dataset$Color_Intensity <- x

print("The outlier values for  Color_Intensity are below: ")
```

```
## [1] "The outlier values for  Color_Intensity are below: "
```

```
boxplot(wine_dataset$Color_Intensity)$out
```

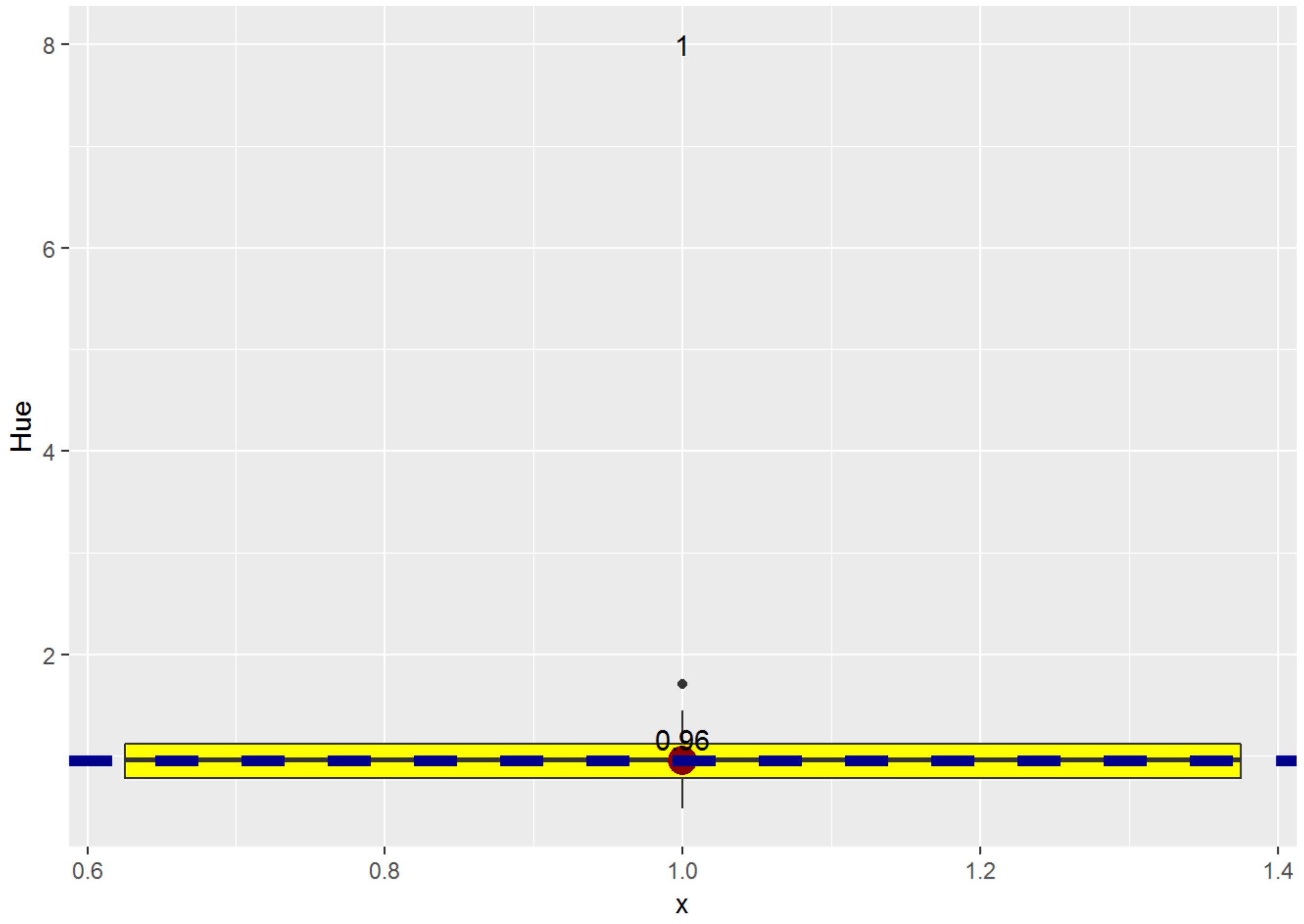


```
## numeric(0)
```

Above we can see capping has removed all outliers from Color_Intensity.

Visualization for Hue.

```r
#Malic_acid, Ash,Ash_Alcanity, Magnesium, Proanthocyanins ,Color_Intensity
#Hue
calcmed <- function(x) {
   return(c(y = 8, label = round(median(x), 1)))
   # modify 8 to suit your needs
}

fun_mean <- function(x){
  return (round((data.frame(y=mean(x),label=mean(x,na.rm=T))),2))}

ggplot(data=wine_dataset, aes(x=1, y=Hue)) +
  geom_boxplot(fill="yellow")+
  #geom_dotplot(binaxis='y', stackdir='center', dotsize=1) +
     stat_summary(fun.data = calcmed, geom = "text") +

  stat_summary(fun.y = mean, geom="point",colour="darkred", size=5) +
  stat_summary(fun.data = fun_mean, geom="text", vjust=-0.5) +
  geom_hline(aes(yintercept=mean(Hue, na.rm=T)), colour = "dark blue", linetype="dashed", size=2)
```



Find the outliers values for Hue

```
print("The outlier values for hue are below: ")
```

```
## [1] "The outlier values for hue are below: "
```

```
boxplot(wine_dataset$Hue, plot=FALSE)$out
```

```
## [1] 1.71
```

Outlier treatment for Hue - The Imputation method able to remove all the outliers, but we will go ahead with capping method for outliers treatment.

```
#This section is just for showcase purpose for outliers with Imputation method.
wine_dataset_RND <- wine_dataset
wine_dataset_RND$Hue[wine_dataset_RND$Hue  %in% c(1.71)] <- median(wine_dataset_RND$Hue, na.rm = T)

print("The outlier values for  Hue are below: ")
```

```
## [1] "The outlier values for  Hue are below: "
```

```
boxplot(wine_dataset_RND$Hue, plot=FALSE)$out
```

```
## numeric(0)
```

```
x <- wine_dataset$Hue
qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
caps <- quantile(x, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]

wine_dataset$Hue <- x

print("The outlier values for  Hue are below: ")
```

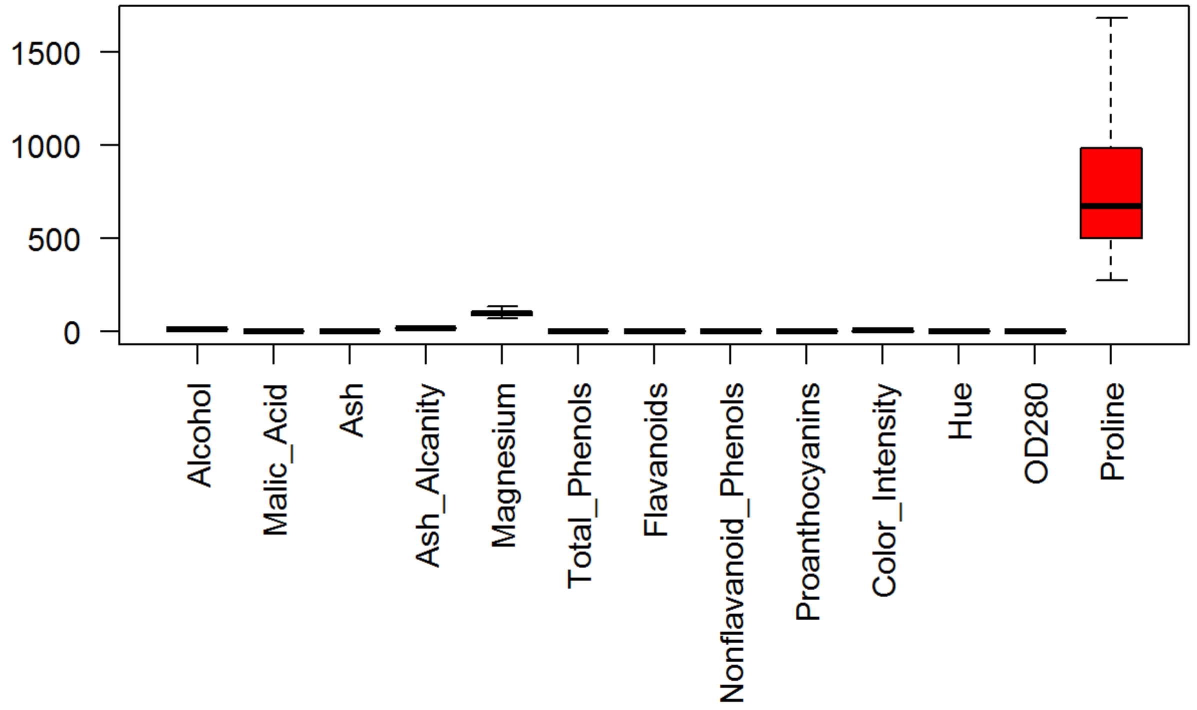
```
## [1] "The outlier values for  Hue are below: "
```

```
boxplot(wine_dataset$Hue, plot=FALSE)$out
```

```
## numeric(0)
```

Let us verify again with combined Box plot

```
boxplot(wine_dataset,
        las = 2,
        col = c("red","sienna","palevioletred1","royalblue2","red","sienna","palevioletred1",
        "royalblue2","red","sienna","palevioletred1","royalblue2"),
        at =c(1,2,3,4, 5,6,7,8,9, 10,11,12,13),
        par(mar = c(12, 5, 4, 2) + 0.1)

        )
```

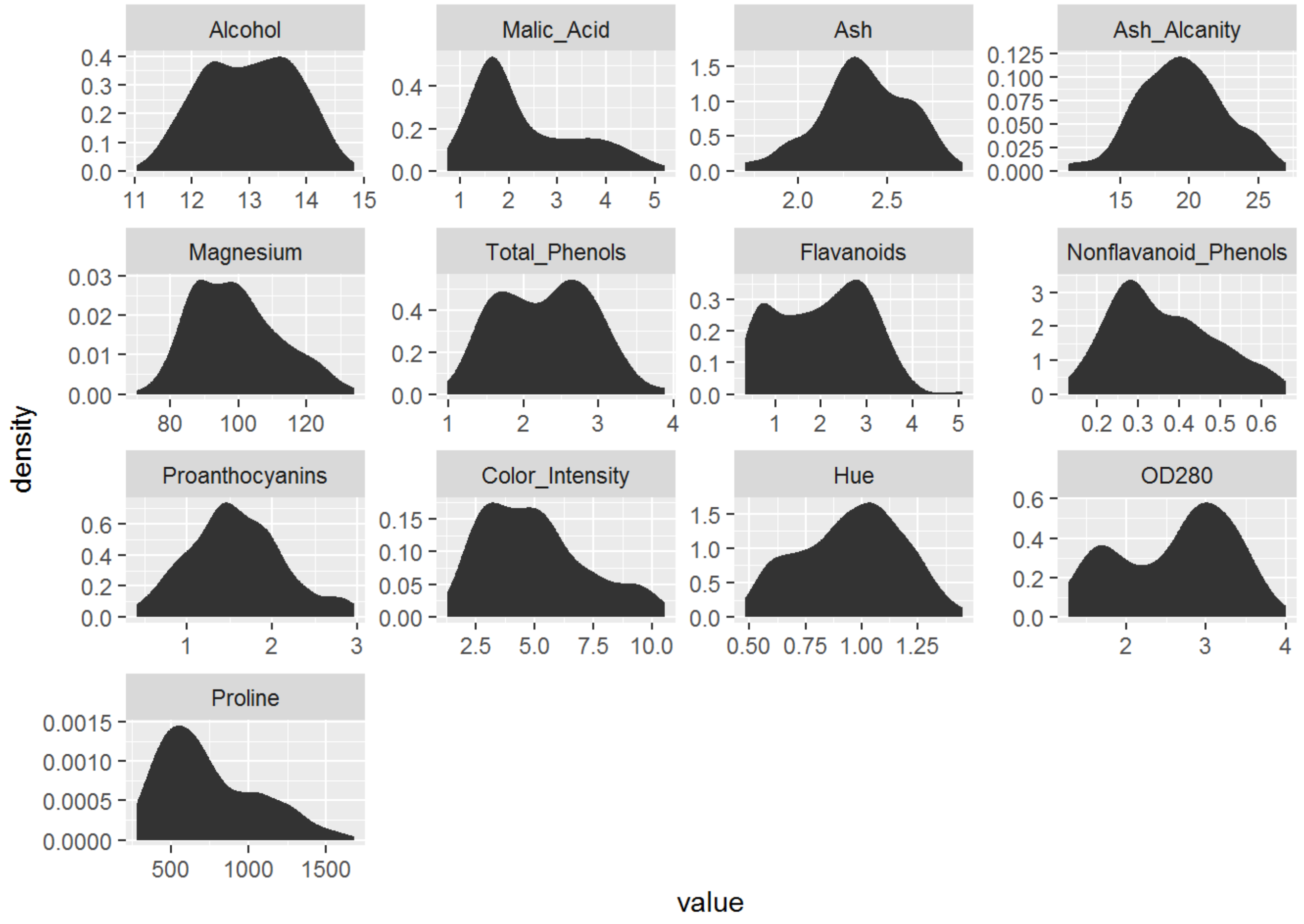


Hurray!, we can see above all the outliers are removed with capping method.

```
require(reshape2)

melt.wine_dataset <- melt(wine_dataset)
```

```
## No id variables; using all as measure variables
```

```
ggplot(data = melt.wine_dataset, aes(x = value)) +
stat_density() +
facet_wrap(~variable, scales = "free")
```



Analysis and visualization of numeric variables.

```
wine_dataset_numericVars <- which(sapply(wine_dataset, is.numeric))
wine_dataset_numericVarNames <- names(wine_dataset_numericVars)

paste("There are ", length(wine_dataset_numericVars) ," numeric variable in datasets")
```

```
## [1] "There are  13  numeric variable in datasets"
```

```
cat("\n") # print one blank new line
```

```
print(wine_dataset_numericVarNames)
```

```
##  [1] "Alcohol"              "Malic_Acid"           "Ash"
##  [4] "Ash_Alcanity"         "Magnesium"            "Total_Phenols"
##  [7] "Flavanoids"           "Nonflavanoid_Phenols" "Proanthocyanins"
## [10] "Color_Intensity"      "Hue"                  "OD280"
## [13] "Proline"
```

Find the corelation between numeric variables.

```
cor(wine_dataset[sapply(wine_dataset, is.numeric)])
```

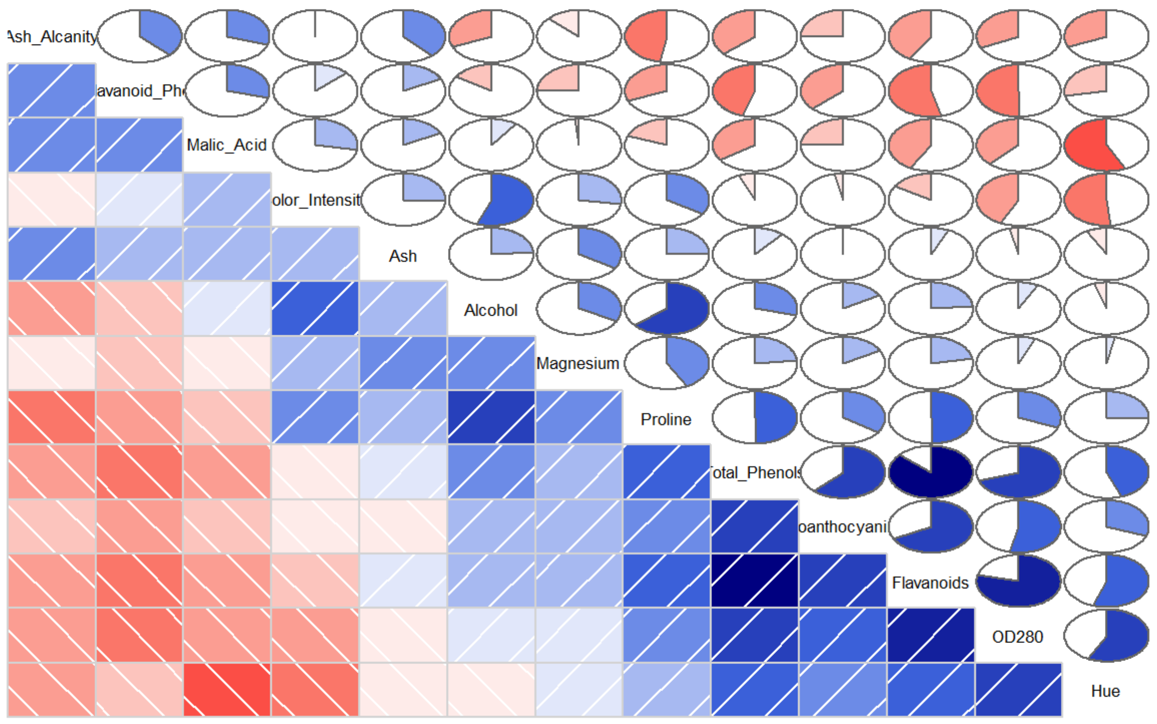
```
##                         Alcohol  Malic_Acid          Ash Ash_Alcanity
## Alcohol              1.00000000  0.09617322  0.238545146 -0.315992932
## Malic_Acid           0.09617322  1.00000000  0.174266248  0.304399498
## Ash                  0.23854515  0.17426625  1.000000000  0.385939331
## Ash_Alcanity        -0.31599293  0.30439950  0.385939331  1.000000000
## Magnesium            0.32822621 -0.01691567  0.340009524 -0.130629882
## Total_Phenols        0.28910112 -0.34411668  0.112498528 -0.368030633
## Flavanoids           0.23681493 -0.41804357  0.066282045 -0.414151857
## Nonflavanoid_Phenols -0.15592947  0.29040684  0.175069632  0.375520378
## Proanthocyanins      0.16288417 -0.24230034 -0.004604632 -0.245259292
## Color_Intensity      0.55594370  0.27862755  0.258573838 -0.005880644
## Hue                 -0.04741295 -0.57537494 -0.077845298 -0.317540557
## OD280                0.07234319 -0.37616727 -0.035148482 -0.322938348
## Proline              0.64372004 -0.18861337  0.240588463 -0.470888417
##                       Magnesium Total_Phenols  Flavanoids
## Alcohol              0.32822621    0.28910112  0.23681493
## Malic_Acid          -0.01691567   -0.34411668 -0.41804357
## Ash                  0.34000952    0.11249853  0.06628204
## Ash_Alcanity        -0.13062988   -0.36803063 -0.41415186
## Magnesium            1.00000000    0.23455224  0.21819958
## Total_Phenols        0.23455224    1.00000000  0.86456350
## Flavanoids           0.21819958    0.86456350  1.00000000
## Nonflavanoid_Phenols -0.25047656   -0.44993530 -0.53789961
## Proanthocyanins      0.17119106    0.62203750  0.67212093
## Color_Intensity      0.27286666   -0.06102695 -0.16465173
## Hue                  0.03051834    0.44205495  0.55601429
## OD280                0.05957201    0.69994936  0.78719390
## Proline              0.42416059    0.49811488  0.49419313
##                     Nonflavanoid_Phenols Proanthocyanins Color_Intensity
## Alcohol                       -0.1559295     0.162884173     0.555943697
## Malic_Acid                     0.2904068    -0.242300338     0.278627554
## Ash                            0.1750696    -0.004604632     0.258573838
## Ash_Alcanity                   0.3755204    -0.245259292    -0.005880644
## Magnesium                     -0.2504766     0.171191058     0.272866663
## Total_Phenols                 -0.4499353     0.622037498    -0.061026950
## Flavanoids                    -0.5378996     0.672120931    -0.164651732
## Nonflavanoid_Phenols           1.0000000    -0.369568200     0.128887729
## Proanthocyanins               -0.3695682     1.000000000    -0.034653973
## Color_Intensity                0.1288877    -0.034653973     1.000000000
## Hue                          -0.2831425     0.309781739    -0.514781650
## OD280                        -0.5032696     0.536912772    -0.427236334
## Proline                      -0.3113852     0.345172081     0.339962173
##                               Hue       OD280    Proline
## Alcohol              -0.04741295  0.07234319  0.6437200
## Malic_Acid           -0.57537494 -0.37616727 -0.1886134
## Ash                  -0.07784530 -0.03514848  0.2405885
## Ash_Alcanity         -0.31754056 -0.32293835 -0.4708884
## Magnesium             0.03051834  0.05957201  0.4241606
## Total_Phenols         0.44205495  0.69994936  0.4981149
## Flavanoids            0.55601429  0.78719390  0.4941931
## Nonflavanoid_Phenols -0.28314245 -0.50326960 -0.3113852
## Proanthocyanins       0.30978174  0.53691277  0.3451721
## Color_Intensity      -0.51478165 -0.42723633  0.3399622
## Hue                   1.00000000  0.57616739  0.2539372
## OD280                 0.57616739  1.00000000  0.3127611
## Proline               0.25393715  0.31276108  1.0000000
```

```r
library(corrgram)
```

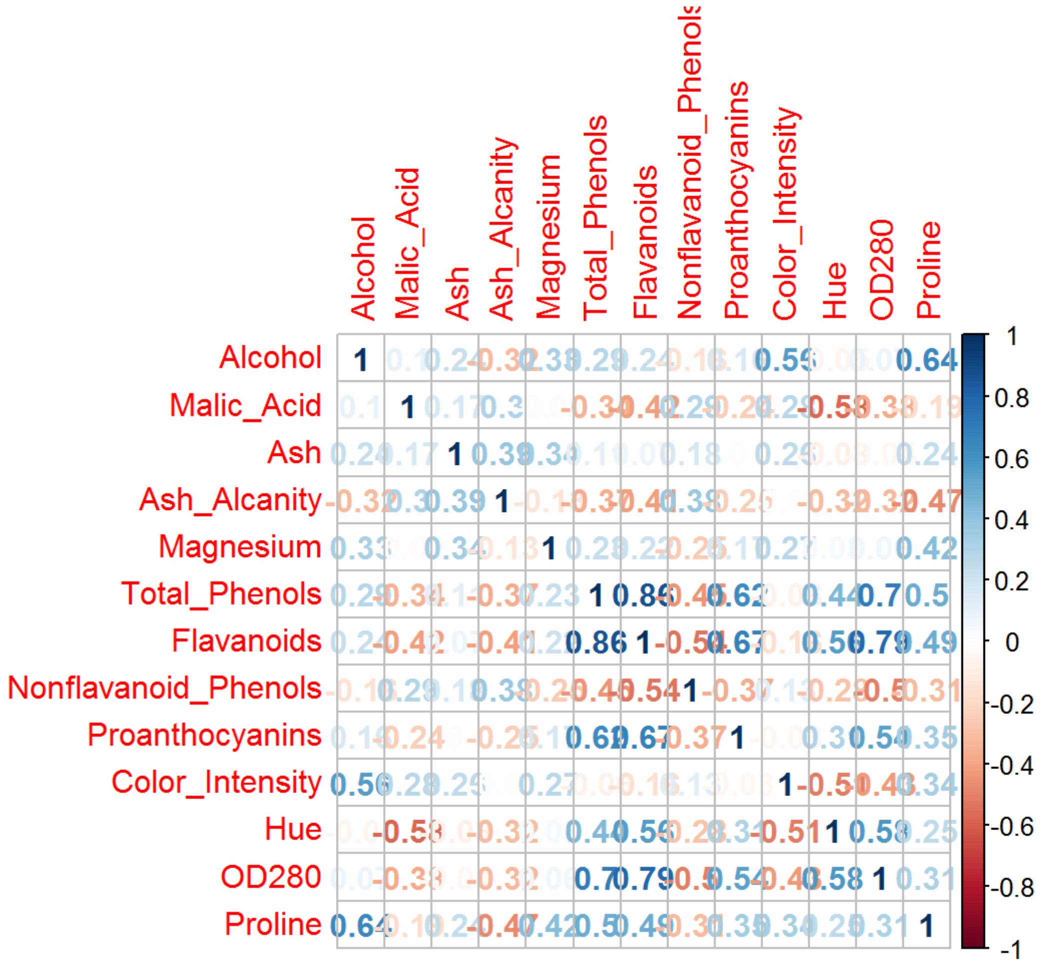
```
## Warning: package 'corrgram' was built under R version 3.5.2
```

```r
corrgram(wine_dataset, order=TRUE, lower.panel=panel.shade,
  upper.panel=panel.pie, text.panel=panel.txt,
  main="Wine dataSet")
```

Wine dataSet



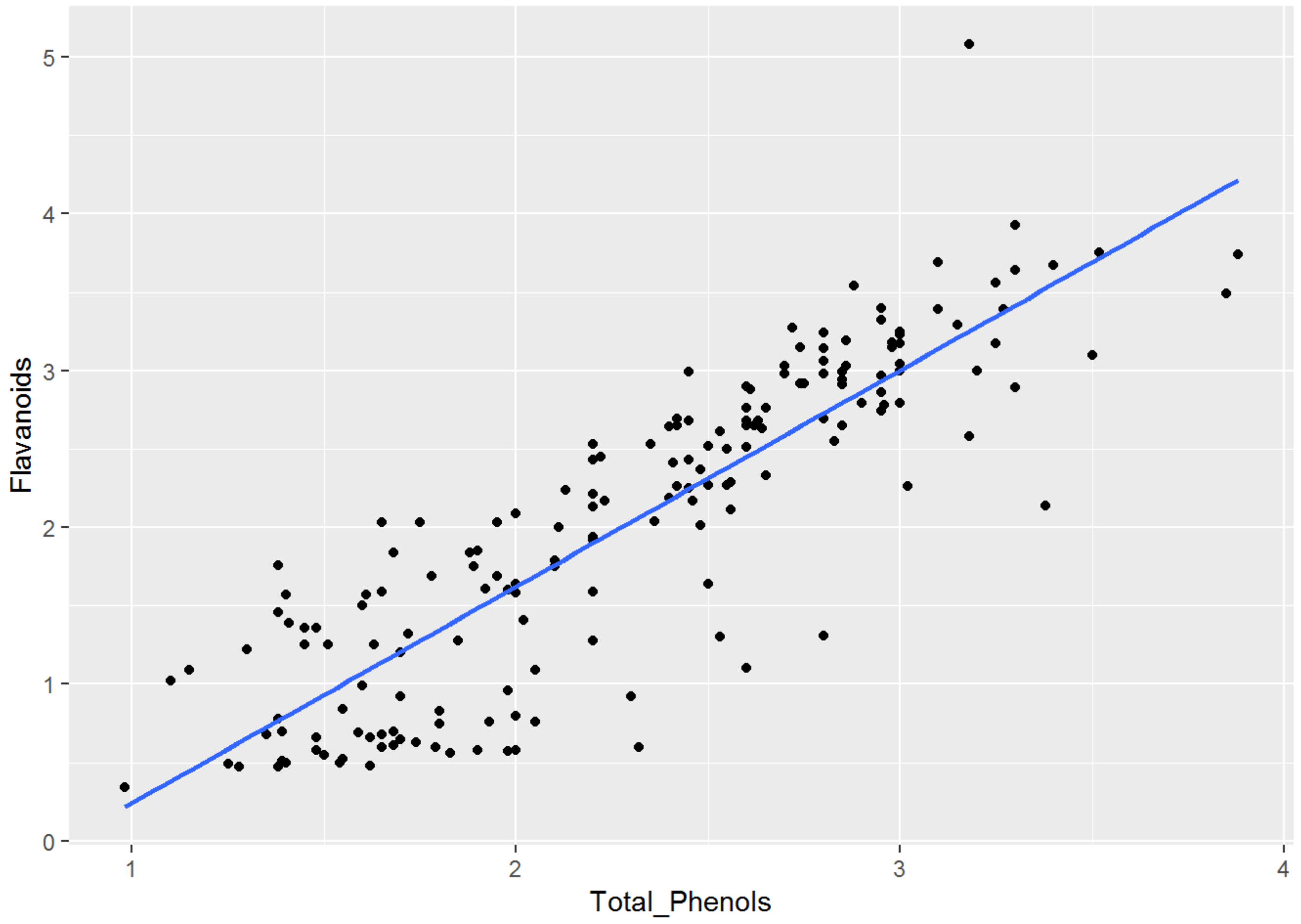
```
M<-cor(wine_dataset)
corrplot(M, method="number")
```



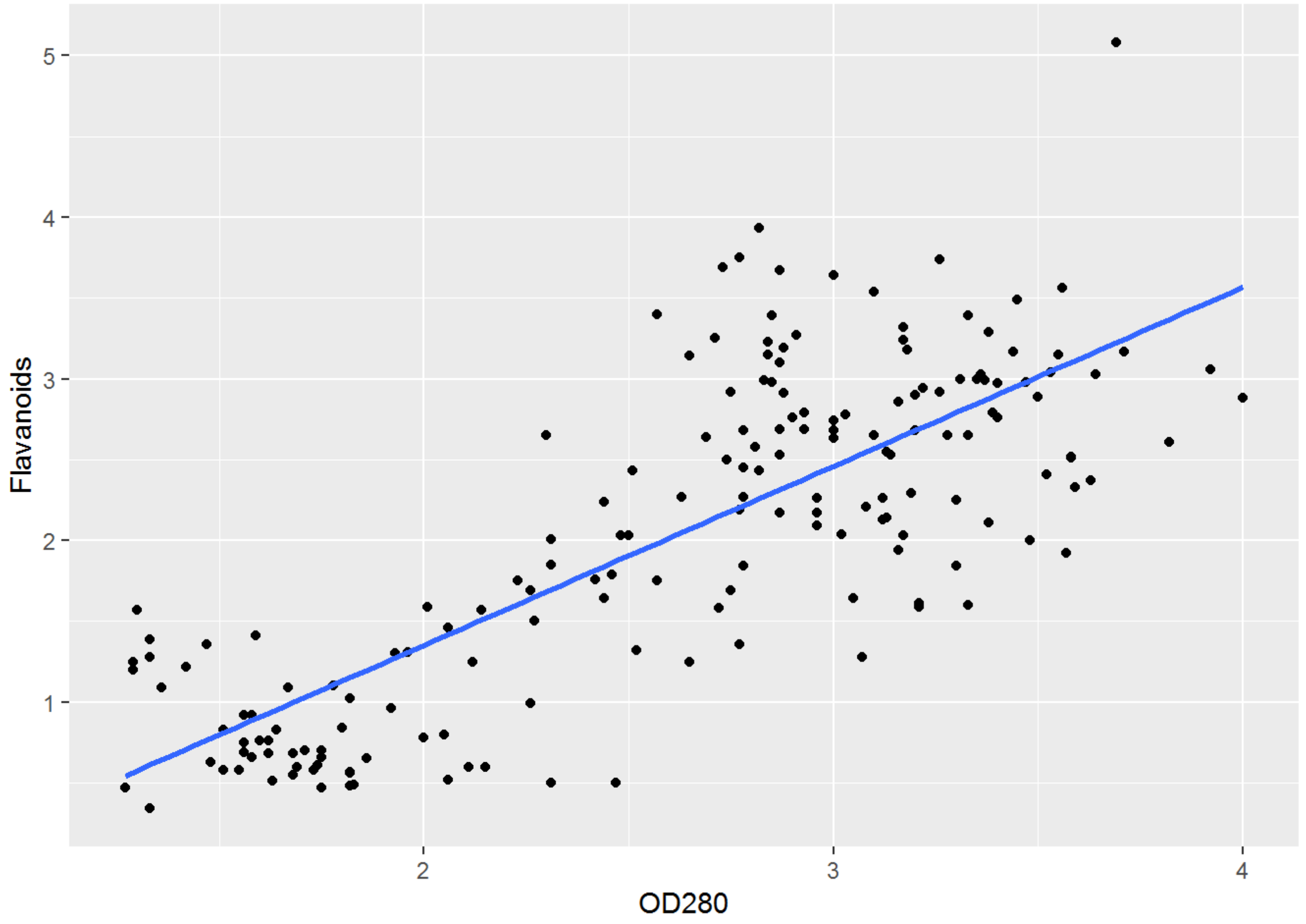
The features which are highly correlated Flavanoids and Total_Phenols Flavanoids and OD280

Let us draw scatter plot for above.

```
ggplot(data=wine_dataset, aes(x=Total_Phenols, y=Flavanoids))+
  geom_point() + geom_smooth(method="lm", se=FALSE)
```



```
ggplot(data=wine_dataset, aes(x=OD280, y=Flavanoids))+
  geom_point() + geom_smooth(method="lm", se=FALSE)
```



The Flavanoids and OD280 are less correlated as compared to Total_Phenols.

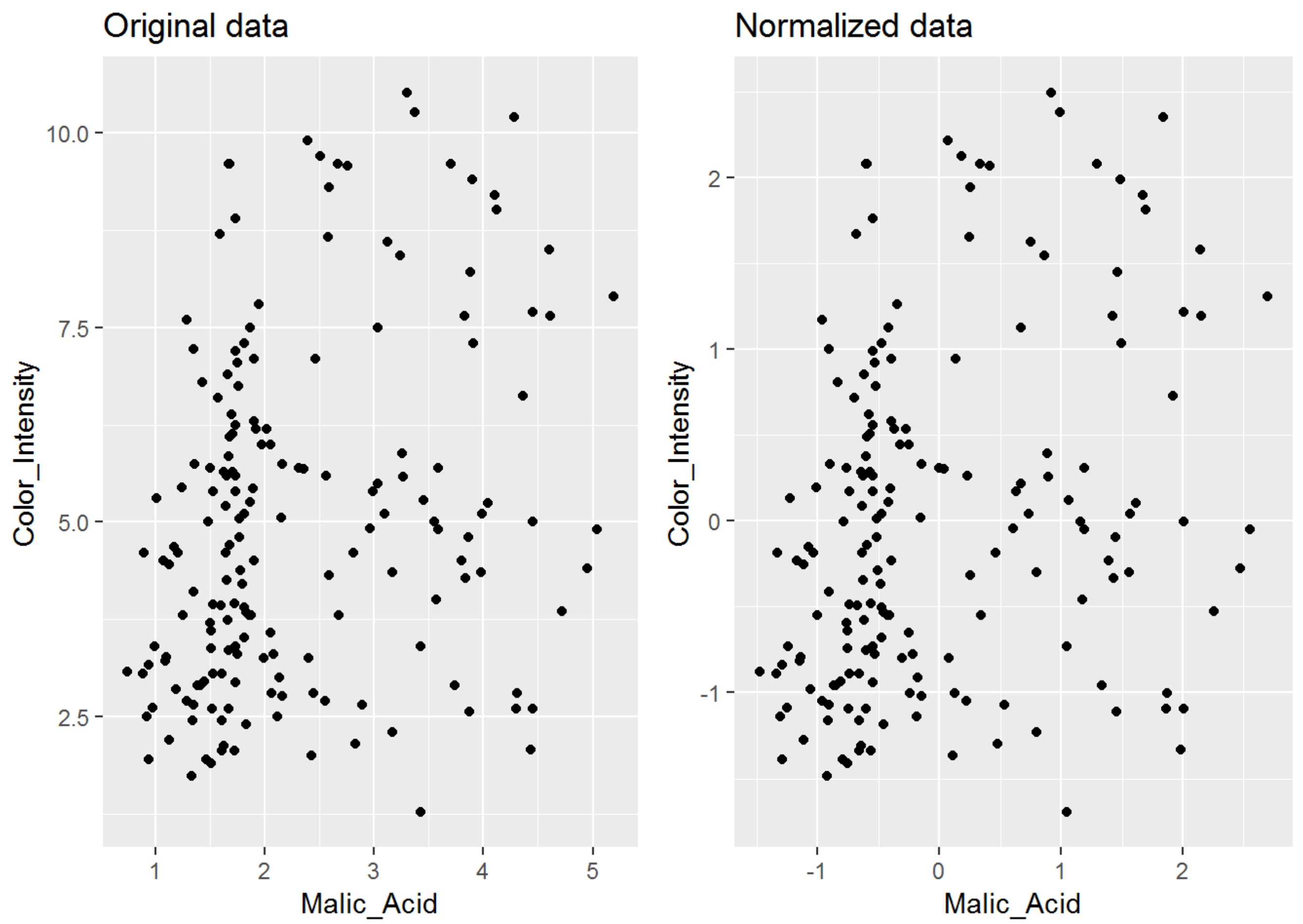
6. data preprocess

Clustering algorithm need data to be in same scale and required normalization.

```
wine_datasetScaled <- as.data.frame(scale(wine_dataset))
```

Visualization of scaled and unscaled

```
# Original data
unscaled <- ggplot(wine_dataset, aes(x=Malic_Acid, y=Color_Intensity)) +
  geom_point() +
  labs(title="Original data")

# Normalized data
scaled <- ggplot(wine_datasetScaled, aes(x=Malic_Acid, y=Color_Intensity)) +
  geom_point() +
  labs(title="Normalized data")

# Subplot
grid.arrange(unscaled, scaled, ncol=2)
```



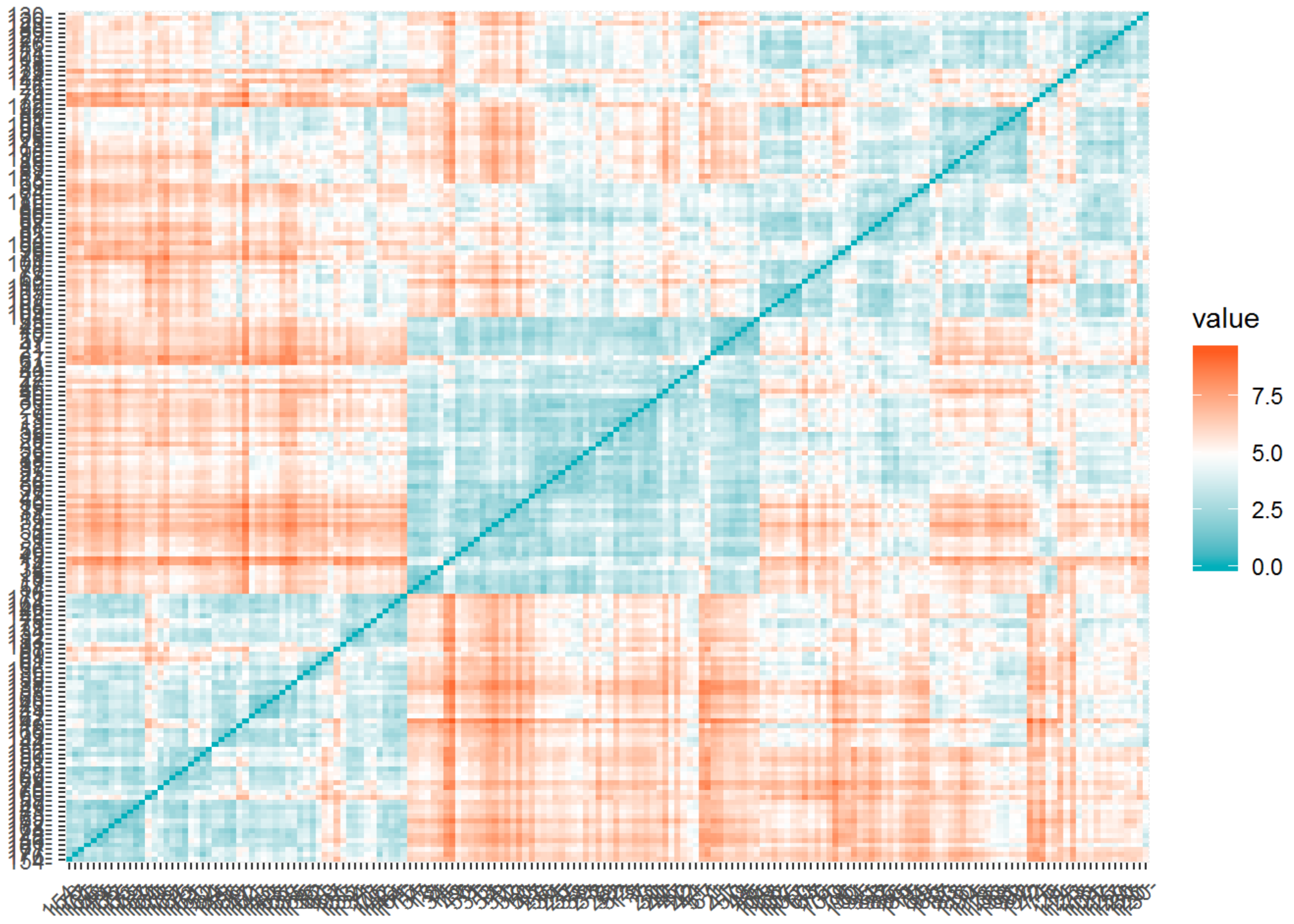
See the scale of X axis has changed.

```
library(tidyverse)  # data manipulation
library(cluster)    # clustering algorithms
library(factoextra) # clustering algorithms & visualization
```

```
## Warning: package 'factoextra' was built under R version 3.5.2
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
distance <- get_dist(wine_datasetScaled)
fviz_dist(distance, gradient = list(low = "#00AFBB", mid = "white", high = "#FC4E07"))
```



7. Model - K-mean clustering

K-means algorithm can be summarized as follows:

Specify the number of clusters (K) to be created (by the analyst) Select randomly k objects from the data set as the initial cluster centers or means Assigns each observation to their closest centroid, based on the Euclidean distance between the object and the centroid For each of the k clusters update the cluster centroid by calculating the new mean values of all the data points in the cluster. The centroid of a Kth cluster is a vector of length p containing the means of all variables for the observations in the kth cluster; p is the number of variables. Iteratively minimize the total within sum of square. That is, iterate steps 3 and 4 until the cluster assignments stop changing or the maximum number of iterations is reached. By default, the R software uses 10 as the default value for the maximum number of iteration.

K-Means Clustering K-means clustering is the most commonly used unsupervised machine learning algorithm for partitioning a given data set into a set of k groups (i.e. k clusters), where k represents the number of groups pre-specified by the analyst. It classifies objects in multiple groups (i.e., clusters), such that objects within the same cluster are as similar as possible (i.e., high intra-class similarity), whereas objects from different clusters are as dissimilar as possible (i.e., low inter-class similarity). In k-means clustering, each cluster is represented by its center (i.e, centroid) which corresponds to the mean of points assigned to the cluster.

The total within-cluster sum of square measures the compactness (i.e goodness) of the clustering and we want it to be as small as possible.

Computing k-means clustering in R We can compute k-means in R with the kmeans function. Here will group the data into two clusters (centers = 2). The kmeans function also has an nstart option that attempts multiple initial configurations and reports on the best one. For example, adding nstart = 25 will generate 25 initial configurations. This approach is often recommended.

```
# Execution of k-means with k=2
set.seed(1234)
wines_km.out <- kmeans(wine_datasetScaled, centers=2, nstart = 25)
```

Show the summary and elements of Model.

```
print(wines_km.out)
```

```
## K-means clustering with 2 clusters of sizes 86, 92
##
## Cluster means:
##      Alcohol Malic_Acid         Ash Ash_Alcanity  Magnesium Total_Phenols
## 1  0.3568876 -0.3461184  0.02939701   -0.5536444  0.3321057     0.7886998
## 2 -0.3336124  0.3235455 -0.02747981    0.5175372 -0.3104466    -0.7372628
##   Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity
## 1  0.8314467           -0.6137948       0.6461509     -0.07232987
## 2 -0.7772219            0.5737647      -0.6040106      0.06761271
##          Hue      OD280    Proline
## 1  0.5560567  0.7187299  0.6247749
## 2 -0.5197921 -0.6718562 -0.5840287
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 1 2 2 1
##  [71] 2 1 2 1 1 2 1 2 2 1 1 1 1 2 2 1 1 2 2 2 2 2 2 2 1 1 1 2 1 1 1 1 2 2 2 1
## [106] 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [141] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [176] 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 719.3588 917.1076
##  (between_SS / total_SS =  28.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Summarize the above output 1. There are 2 cluster, cluster size 1: 86, Cluster size2 = 92 2. Cluster 1 has maximum elements of Flavanoids, Total_Phenols 3. Cluster 2 has maximum elements of Nonflavanoid_Phenols, Ash_Alcanity

kmeans returns an object of class which has a print and a fitted method. It is a list with at least the following components:

cluster A vector of integers (from 1:k) indicating the cluster to which each point is allocated. centers A matrix of cluster centres. totss The total sum of squares. withinss Vector of within-cluster sum of squares, one component per cluster. tot.withinss Total within-cluster sum of squares, i.e. sum(withinss). betweenss The between-cluster sum of squares, i.e. totss-tot.withinss. size The number of points in each cluster. iter The number of (outer) iterations. ifault integer: indicator of a possible algorithm problem - for experts

Print the cluster elements

```
print(wines_km.out$cluster)
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 1 2 2 1
##  [71] 2 1 2 1 1 2 1 2 1 1 1 1 2 2 1 1 2 2 2 2 2 2 2 1 1 1 2 1 1 1 1 2 2 2 1
## [106] 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [141] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [176] 2 2 2
```

Print the cluster centers

```
print(wines_km.out$centers)
```

```
##      Alcohol Malic_Acid         Ash Ash_Alcanity  Magnesium Total_Phenols
## 1  0.3568876 -0.3461184  0.02939701   -0.5536444  0.3321057     0.7886998
## 2 -0.3336124  0.3235455 -0.02747981    0.5175372 -0.3104466    -0.7372628
##   Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity
## 1  0.8314467           -0.6137948       0.6461509     -0.07232987
## 2 -0.7772219            0.5737647      -0.6040106      0.06761271
##          Hue      OD280    Proline
## 1  0.5560567  0.7187299  0.6247749
## 2 -0.5197921 -0.6718562 -0.5840287
```

1. Cluster 1 has maximum elements of Flavanoids, Total_Phenols
2. Cluster 2 has maximum elements of Nonflavanoid_Phenols, Ash_Alcanity

The below paramter provide information about cluster charectistics and size.

betweenss. The between-cluster sum of squares. In an optimal segmentation, one expects this ratio to be as higher as possible, since we would like to have heterogeneous clusters.

withinss. Vector of within-cluster sum of squares, one component per cluster. In an optimal segmentation, one expects this ratio to be as lower as possible for each cluster, since we would like to have homogeneity within the clusters.

tot.withinss. Total within-cluster sum of squares.

totss. The total sum of squares

```
paste("Total within-cluster sum of squares: ", round(wines_km.out$tot.withinss,2))
```

```
## [1] "Total within-cluster sum of squares:  1636.47"
```

```
paste("Total sum of squares: ",round(wines_km.out$totss,2))
```

```
## [1] "Total sum of squares:  2301"
```

```
paste("Within-cluster sum of squares: ",round(wines_km.out$withinss,2))
```

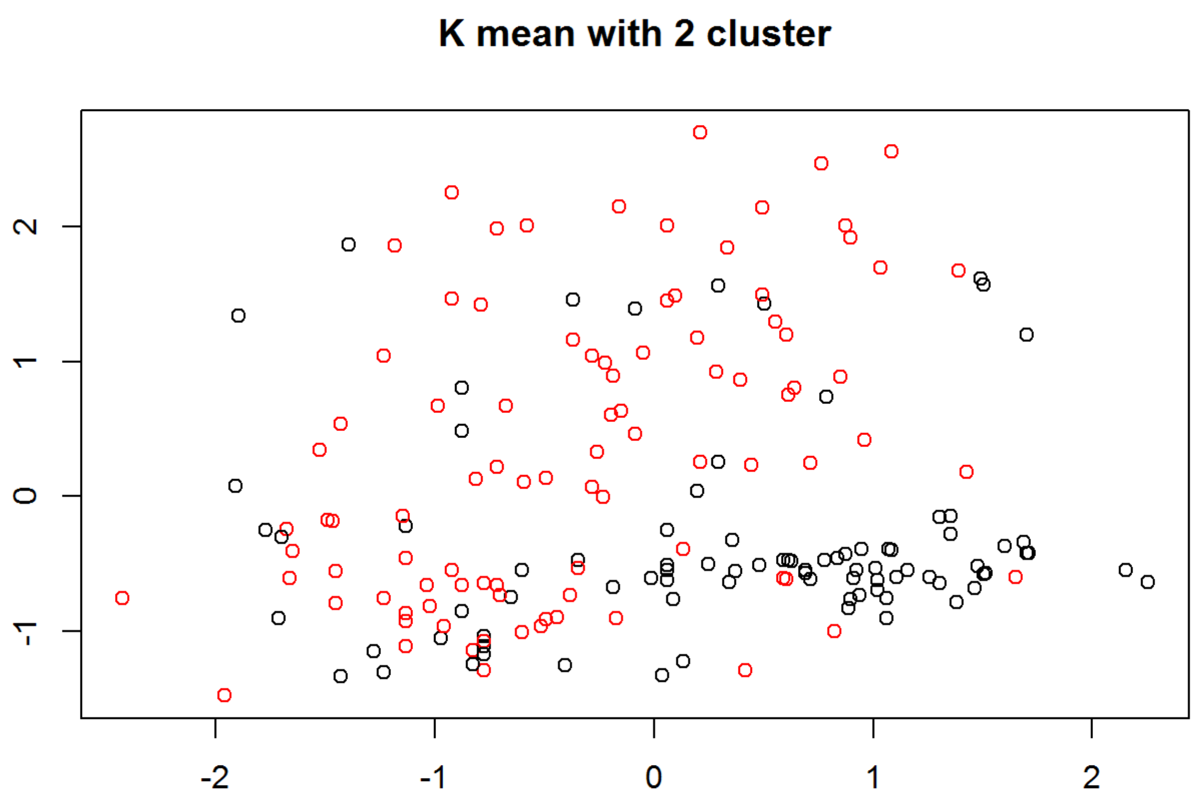
```
## [1] "Within-cluster sum of squares:  719.36"
## [2] "Within-cluster sum of squares:  917.11"
```

```
paste("Between-cluster sum of squares: ",round(wines_km.out$betweenss,2))
```

```
## [1] "Between-cluster sum of squares:  664.53"
```

Visualization and interpretation of results.

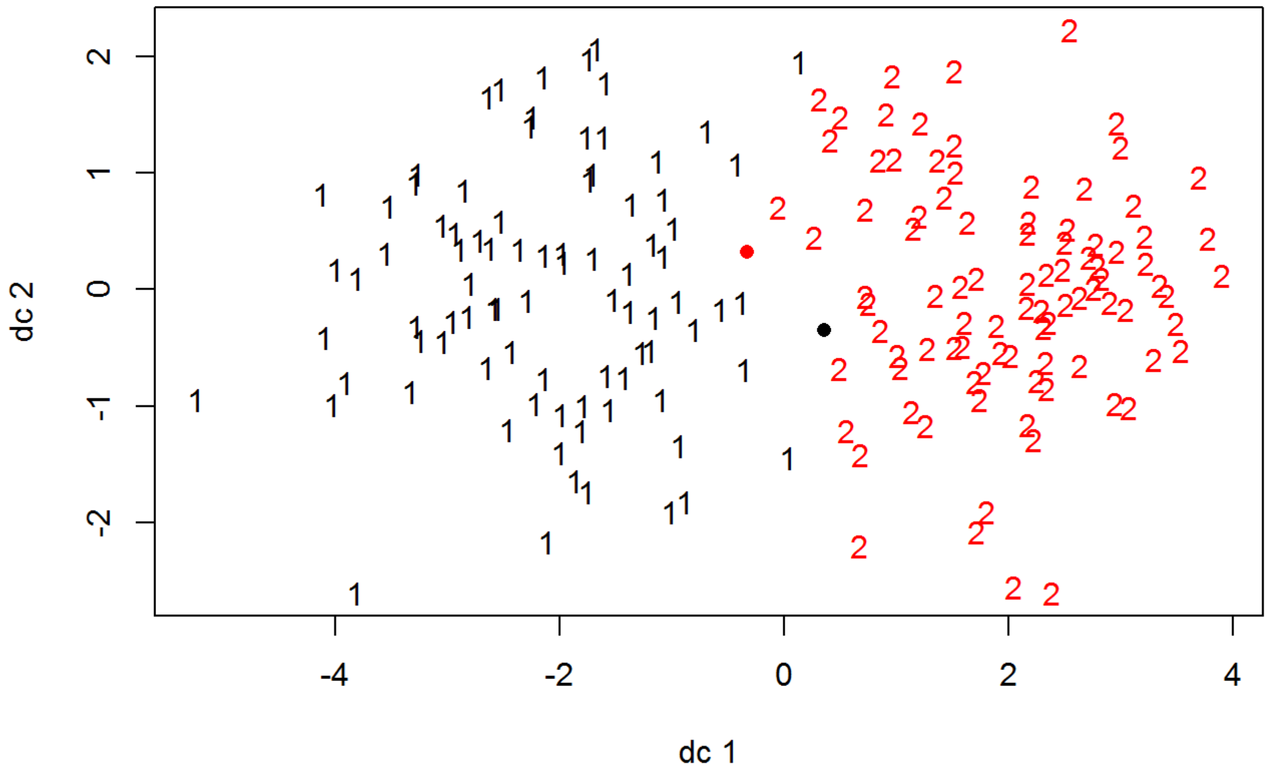
```
plot(as.matrix(wine_datasetScaled), col=wines_km.out$cluster, main = "K mean with 2 cluster",
     xlab="", ylab="")
```



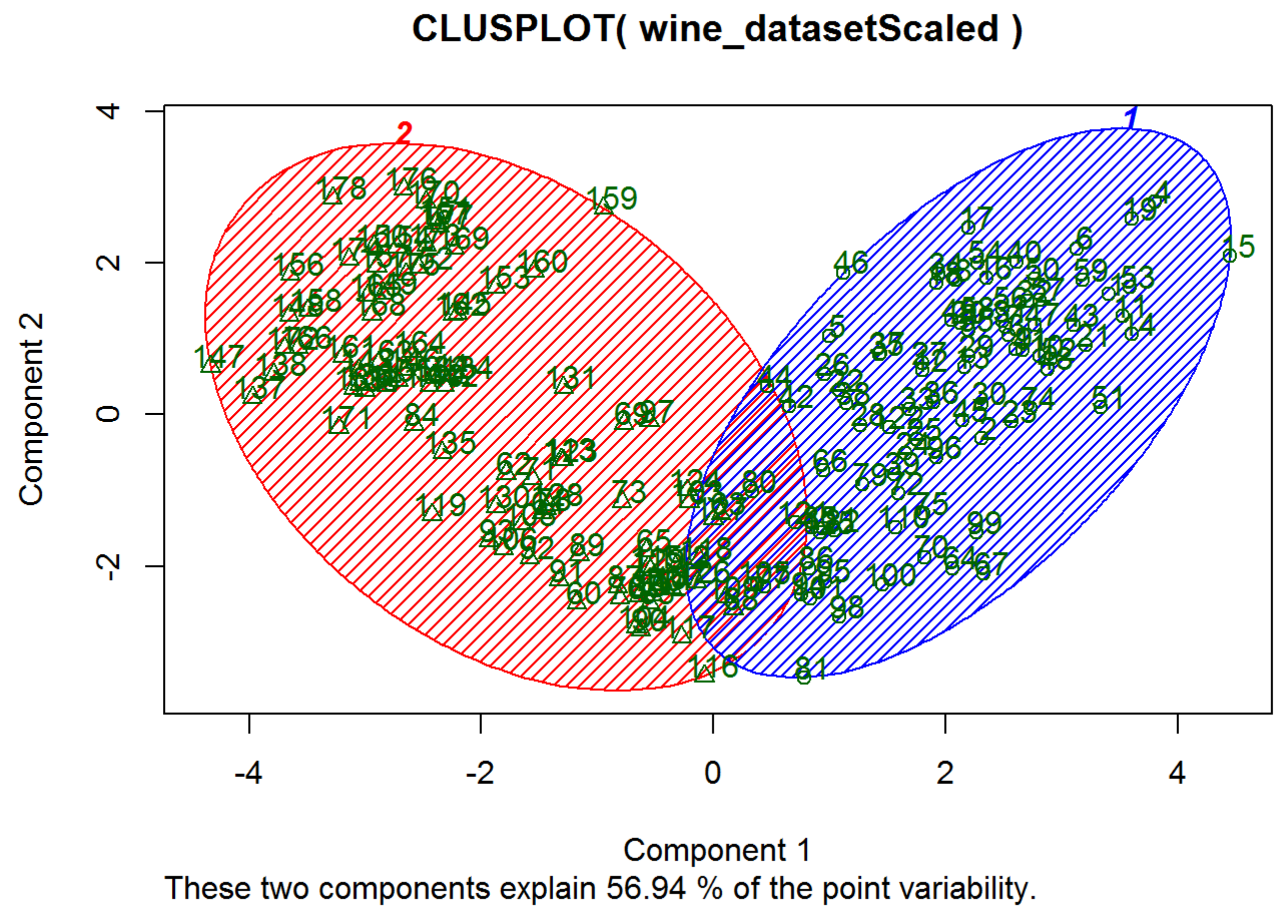
```
library(cluster)
 library(fpc)
```

```
## Warning: package 'fpc' was built under R version 3.5.2
```

```
 plotcluster(wine_datasetScaled,wines_km.out$cluster)
points(wines_km.out$centers,col=1:8,pch=16)
```

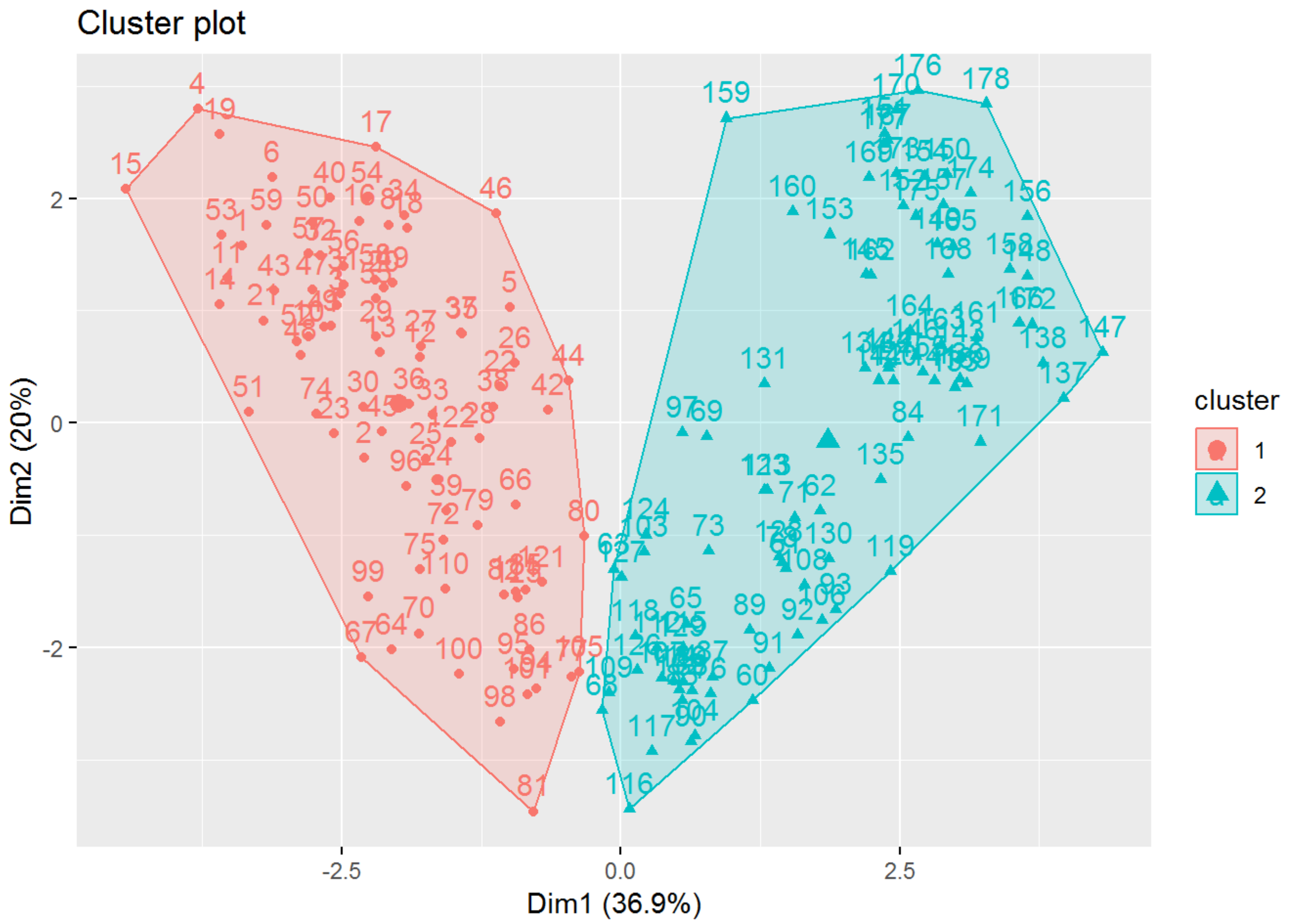


```
clusplot(wine_datasetScaled, wines_km.out$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



We can also view our results by using fviz_cluster. This provides a nice illustration of the clusters. If there are more than two dimensions (variables) fviz_cluster will perform principal component analysis (PCA) and plot the data points according to the first two principal components that explain the majority of the variance.

```
fviz_cluster(wines_km.out, data = wine_datasetScaled)
```



checking mean for each object in each cluster

Usually, as the result of a k-means clustering analysis, we would examine the means for each cluster on each dimension to assess how distinct our k clusters are

```
wines_km.out$centers
```

```
##      Alcohol Malic_Acid         Ash Ash_Alcanity  Magnesium Total_Phenols
## 1  0.3568876 -0.3461184  0.02939701   -0.5536444  0.3321057     0.7886998
## 2 -0.3336124  0.3235455 -0.02747981    0.5175372 -0.3104466    -0.7372628
##   Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity
## 1  0.8314467           -0.6137948       0.6461509     -0.07232987
## 2 -0.7772219            0.5737647      -0.6040106      0.06761271
##          Hue      OD280    Proline
## 1  0.5560567  0.7187299  0.6247749
## 2 -0.5197921 -0.6718562 -0.5840287
```

Determining Optimal Clusters As you may recall the analyst specifies the number of clusters to use; preferably the analyst would like to use the optimal number of clusters. To aid the analyst, the following explains the three most popular methods for determining the optimal clusters, which includes:

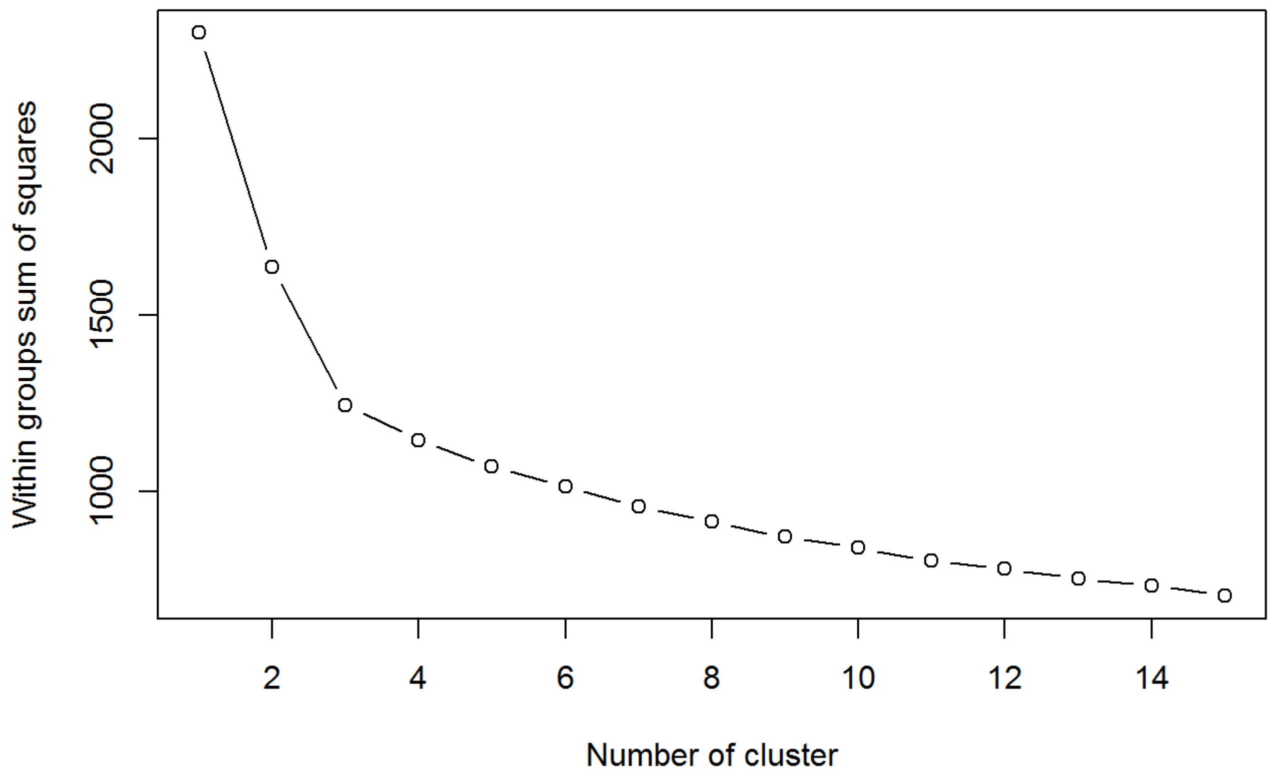
Elbow method Silhouette method Gap statistic

Elbow Method -The basic idea behind cluster partitioning methods, such as k-means clustering, is to define clusters such that the total intra-cluster variation (known as total within-cluster variation or total within-cluster sum of square) is minimized:

The total within-cluster sum of square (wss) measures the compactness of the clustering and we want it to be as small as possible. Thus, we can use the following algorithm to define the optimal clusters:

Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters For each k, calculate the total within-cluster sum of square (wss) Plot the curve of wss according to the number of clusters k. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

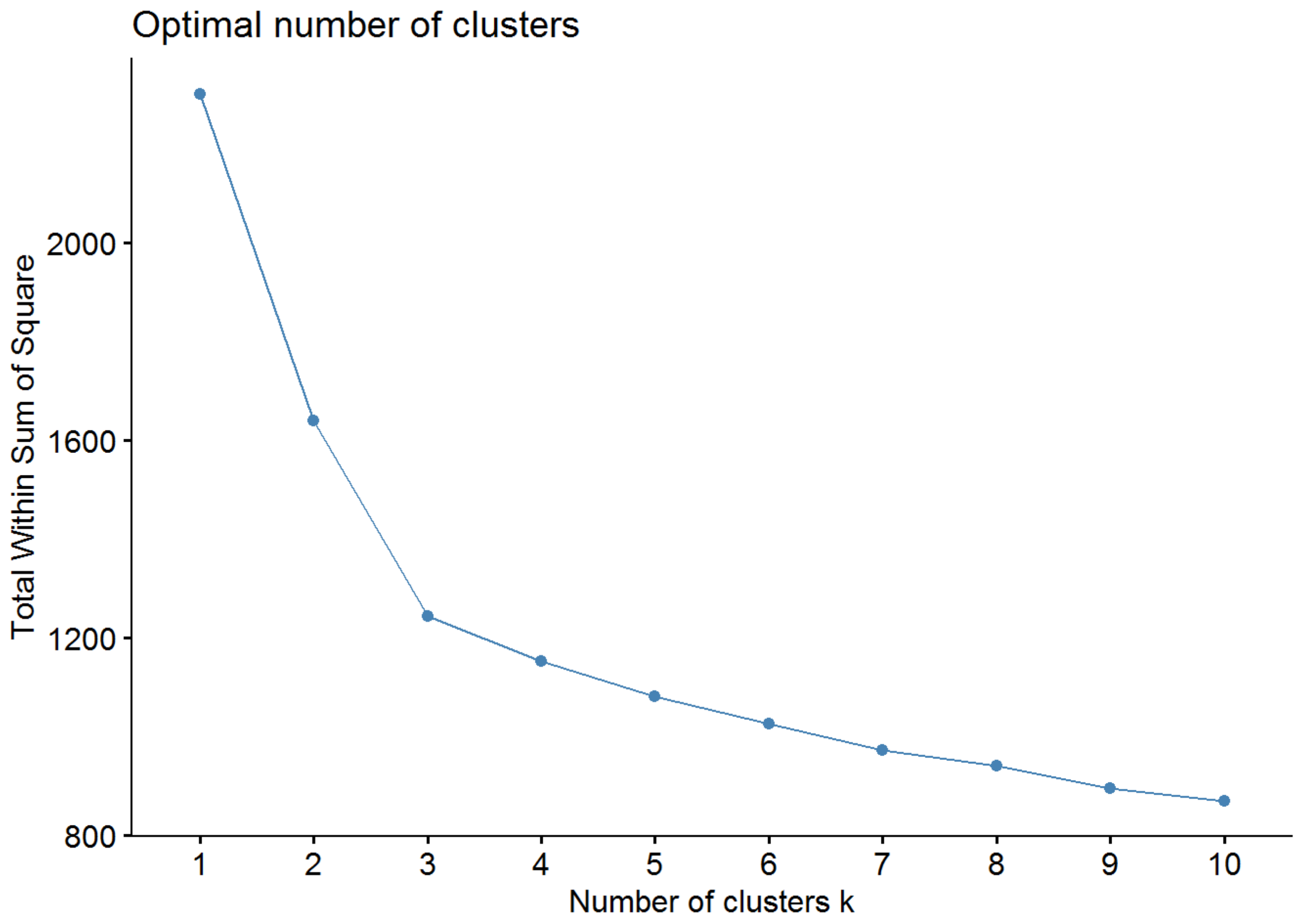
```r
#Initialize the total within sum of square error: wss
set.seed(1234)
wss<- 0

#For 1 to 15 cluster centers
for ( i in 1:15) {

  km.out <- kmeans(wine_datasetScaled, centers = i, nstart = 25)
  wss[i] <- km.out$tot.withinss

}


plot(1:15, wss, type="b",
     xlab="Number of cluster", ylab="Within groups sum of squares")
```



Elbow method shows k =3

Fortunately, this process to compute the “Elbow method” has been wrapped up in a single function (fviz_nbclust):

```
set.seed(1234)

fviz_nbclust(wine_datasetScaled, kmeans, method = "wss")
```

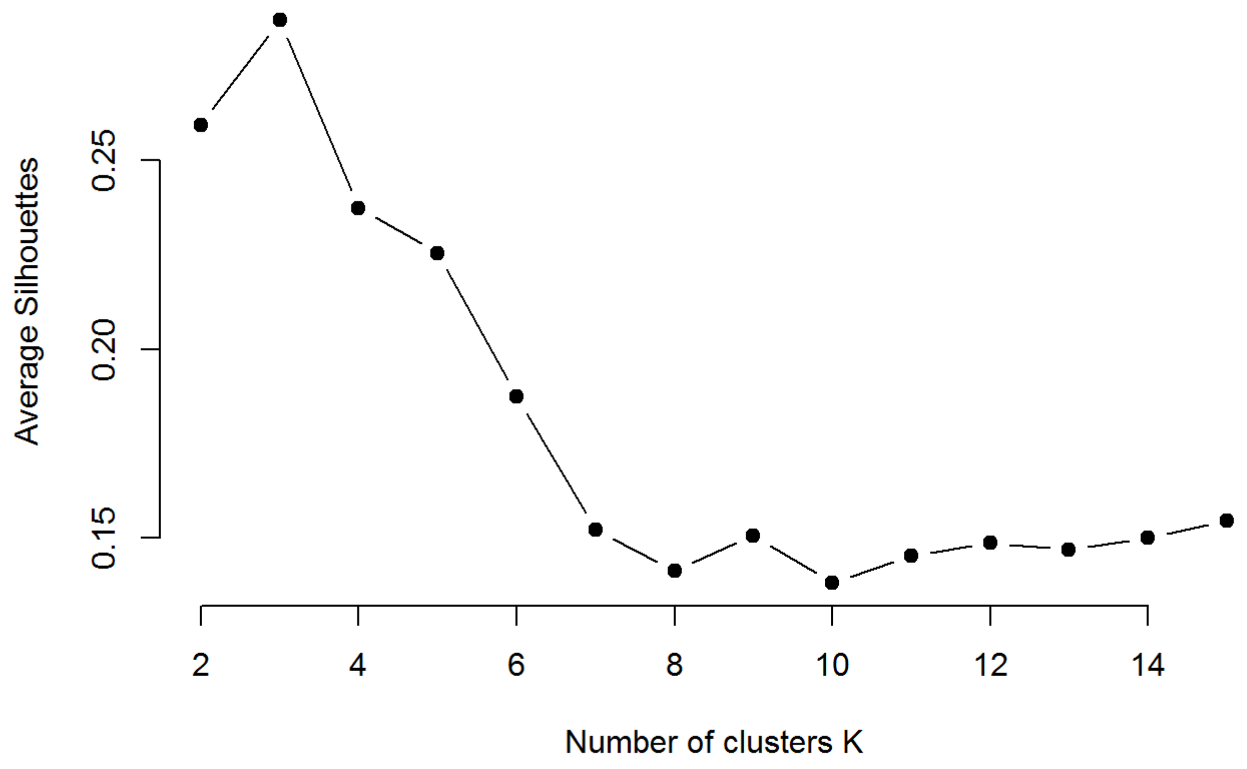


The Elbow methos shows k =3

2.Average Silhouette Method In short, the average silhouette approach measures the quality of a clustering. That is, it determines how well each object lies within its cluster. A high average silhouette width indicates a good clustering. The average silhouette method computes the average silhouette of observations for different values of k.

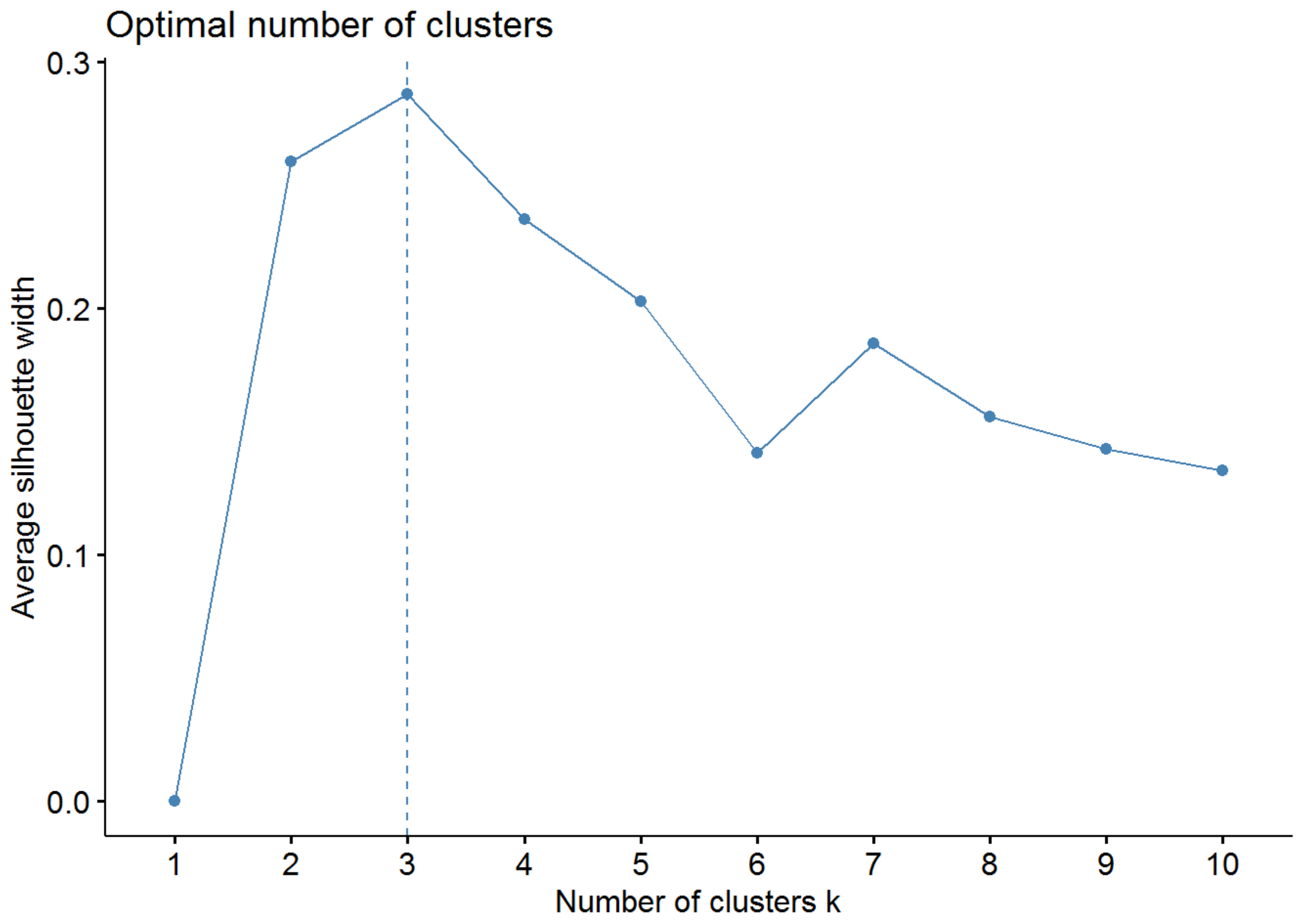
We can use the silhouette function in the cluster package to compuate the average silhouette width. The following code computes this approach for 1-15 clusters. The results show that 2 clusters maximize the average silhouette values with 4 clusters coming in as second optimal number of clusters.

```r
# function to compute average silhouette for k clusters
avg_sil <- function(k) {
  km.res <- kmeans(wine_datasetScaled, centers = k, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(wine_datasetScaled))
  mean(ss[, 3])
}

# Compute and plot wss for k = 2 to k = 15
k.values <- 2:15

# extract avg silhouette for 2-15 clusters
avg_sil_values <- map_dbl(k.values, avg_sil)

plot(k.values, avg_sil_values,
       type = "b", pch = 19, frame = FALSE,
       xlab = "Number of clusters K",
       ylab = "Average Silhouettes")
```



Similar to the elbow method, this process to compute the “average silhoutte method” has been wrapped up in a single function (fviz_nbclust):

```
fviz_nbclust(wine_datasetScaled, kmeans, method = "silhouette")
```



The number of cluster required for Silhouettes is 3.

3. Gap Statistic Method The gap statistic has been published by R. Tibshirani, G. Walther, and T. Hastie (Standford University, 2001). The approach can be applied to any clustering method (i.e. K-means clustering, hierarchical clustering). The gap statistic compares the total intracluster variation for different values of k with their expected values under null reference distribution of the data (i.e. a distribution with no obvious clustering).

To compute the gap statistic method we can use the clusGap function which provides the gap statistic and standard error for an output

```
# compute gap statistic
set.seed(1234)
gap_stat <- clusGap(wine_datasetScaled, FUN = kmeans, nstart = 25,
                    K.max = 10, B = 50)
```

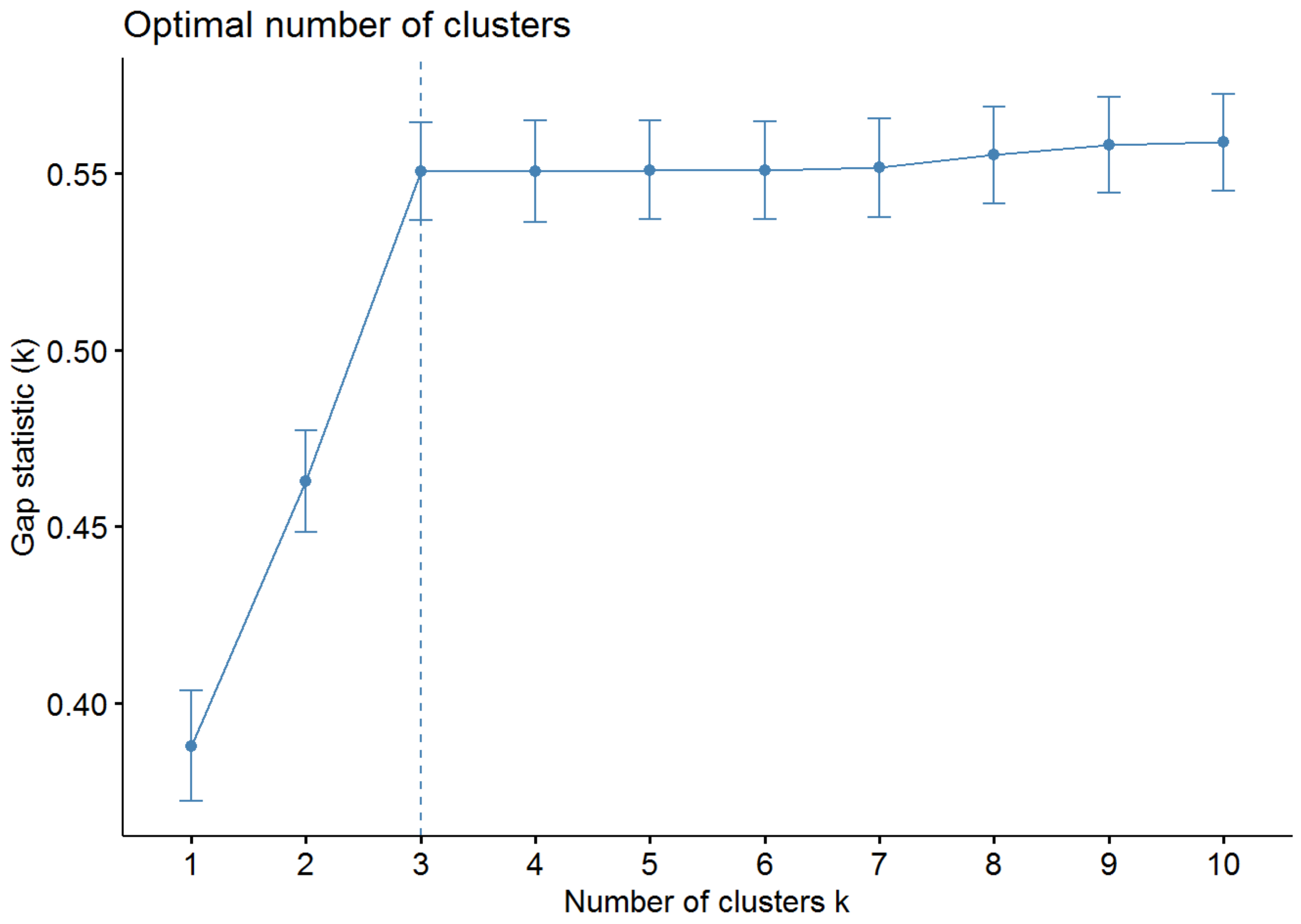
```
## Warning: did not converge in 10 iterations
```

```
print(gap_stat, method = "firstmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = wine_datasetScaled, FUNcluster = kmeans, K.max = 10,     B = 50, nstart = 25)
## B=50 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'firstmax'): 3
##           logW   E.logW       gap     SE.sim
##  [1,] 5.380962 5.768781 0.3878193 0.01563194
##  [2,] 5.205627 5.668533 0.4629060 0.01442375
##  [3,] 5.066175 5.616844 0.5506691 0.01389233
##  [4,] 5.024341 5.574979 0.5506371 0.01430535
##  [5,] 4.989076 5.540101 0.5510253 0.01406824
##  [6,] 4.958879 5.509678 0.5507985 0.01385527
##  [7,] 4.930399 5.482032 0.5516329 0.01396819
##  [8,] 4.901526 5.456756 0.5552298 0.01371514
##  [9,] 4.875430 5.433527 0.5580972 0.01364949
## [10,] 4.853293 5.412144 0.5588506 0.01381019
```

We can visualize the results with fviz_gap_stat which suggests four clusters as the optimal number of clusters.

```
fviz_gap_stat(gap_stat)
```



The optimal number of cluster by

Gap statistics are - 3, K=3

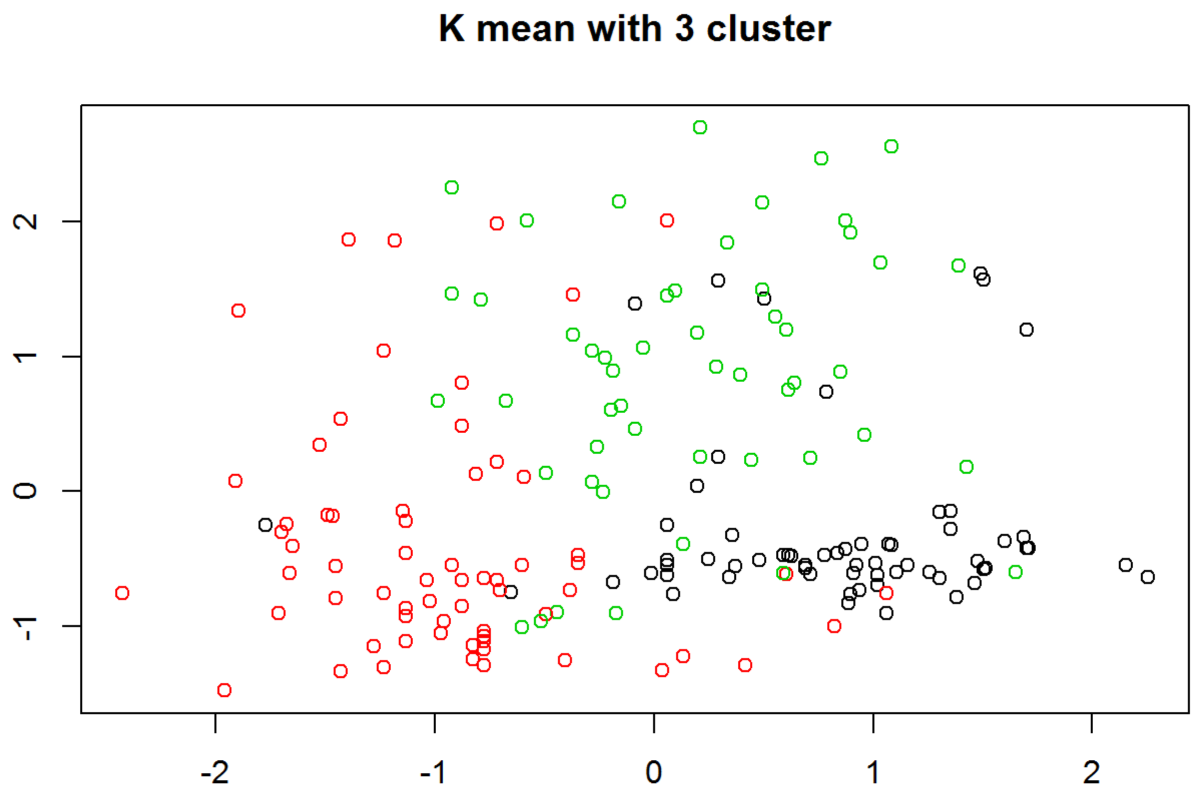
Since k=3 is the value estimated by all method, we will use k=3 and remodel again.

```
# Execution of k-means with k=3
set.seed(1234)
wines_km.out <- kmeans(wine_datasetScaled, centers=3, nstart = 25,iter.max = 50)

print(wines_km.out)
```

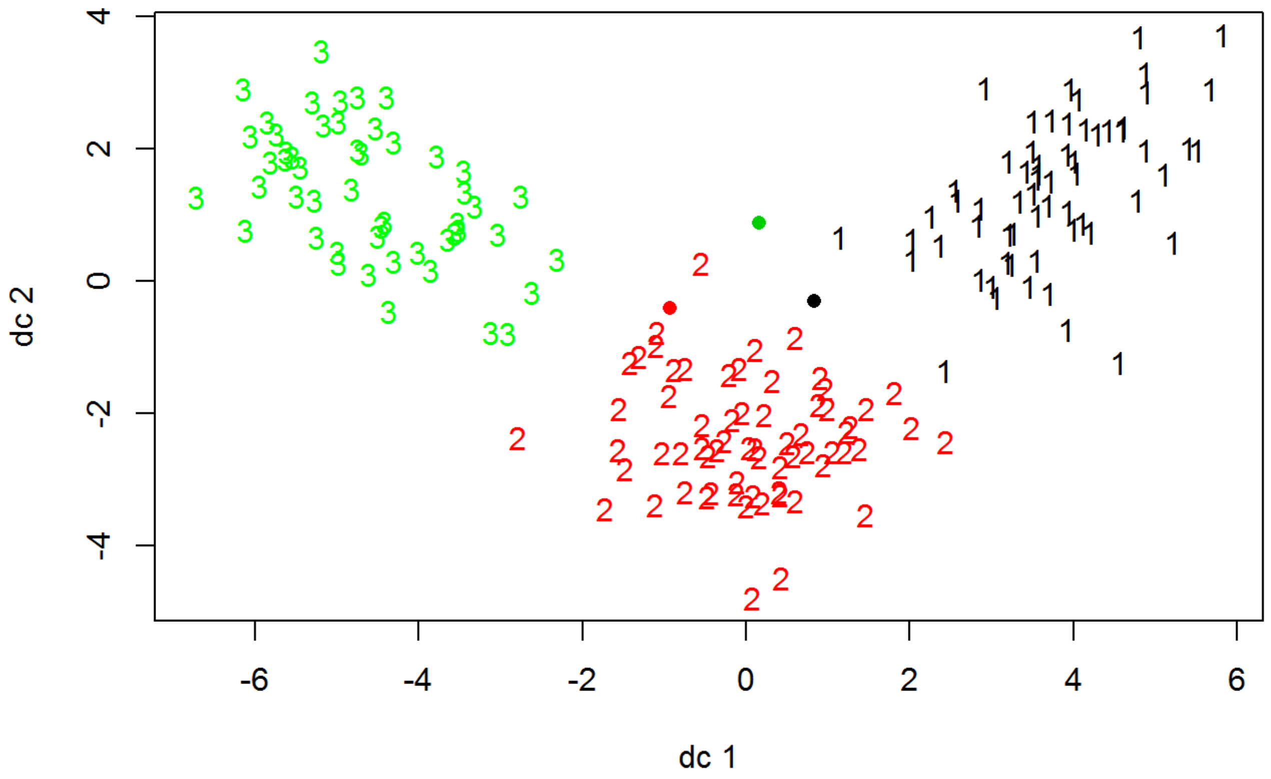
```
## K-means clustering with 3 clusters of sizes 62, 65, 51
##
## Cluster means:
##      Alcohol Malic_Acid        Ash Ash_Alcanity   Magnesium Total_Phenols
## 1  0.8328826 -0.2984676  0.3419812   -0.6787238  0.61529461    0.88274724
## 2 -0.9234669 -0.4121344 -0.4919671    0.1987897 -0.55234295   -0.07576891
## 3  0.1644436  0.8881123  0.2112750    0.5717557 -0.04403871   -0.97657548
##    Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity
## 1  0.97506900          -0.56050853      0.59948662       0.1992738
## 2  0.02075402          -0.03343924      0.05077202      -0.9254173
## 3 -1.21182921           0.72402116     -0.79349710       0.9371990
##          Hue      OD280    Proline
## 1  0.4956031  0.7770551  1.1220202
## 2  0.4537631  0.2700025 -0.7517257
## 3 -1.1808233 -1.2887761 -0.4059428
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2
##  [71] 2 2 2 1 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
## [106] 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [176] 3 3 3
##
## Within cluster sum of squares by cluster:
## [1] 365.4054 541.6852 336.0763
##  (between_SS / total_SS =  46.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Visualization and interpretation of results.

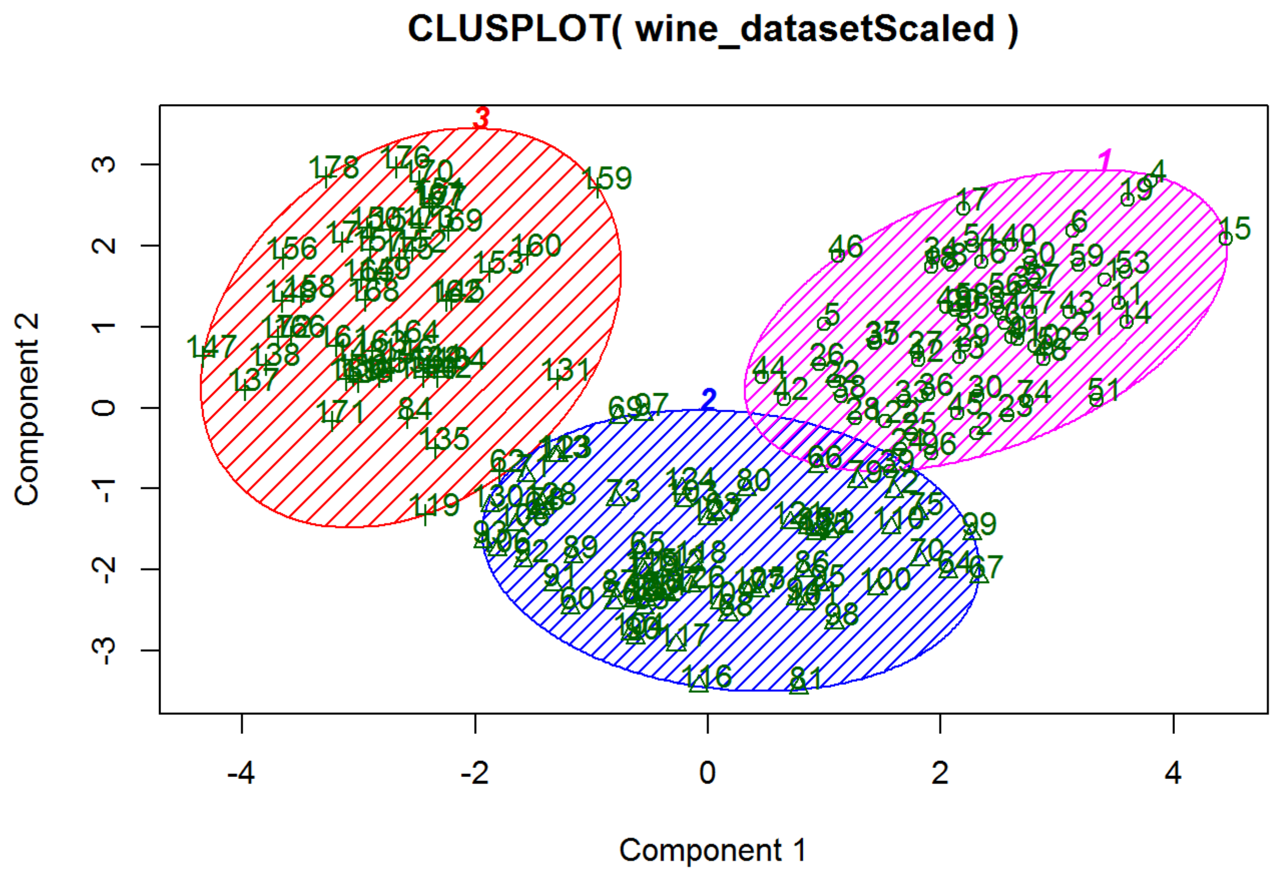
```
plot(as.matrix(wine_datasetScaled), col=wines_km.out$cluster, main = "K mean with 3 cluster",
     xlab="", ylab="")
```



```
library(cluster)
 library(fpc)
 plotcluster(wine_datasetScaled,wines_km.out$cluster)
points(wines_km.out$centers,col=1:8,pch=16)
```

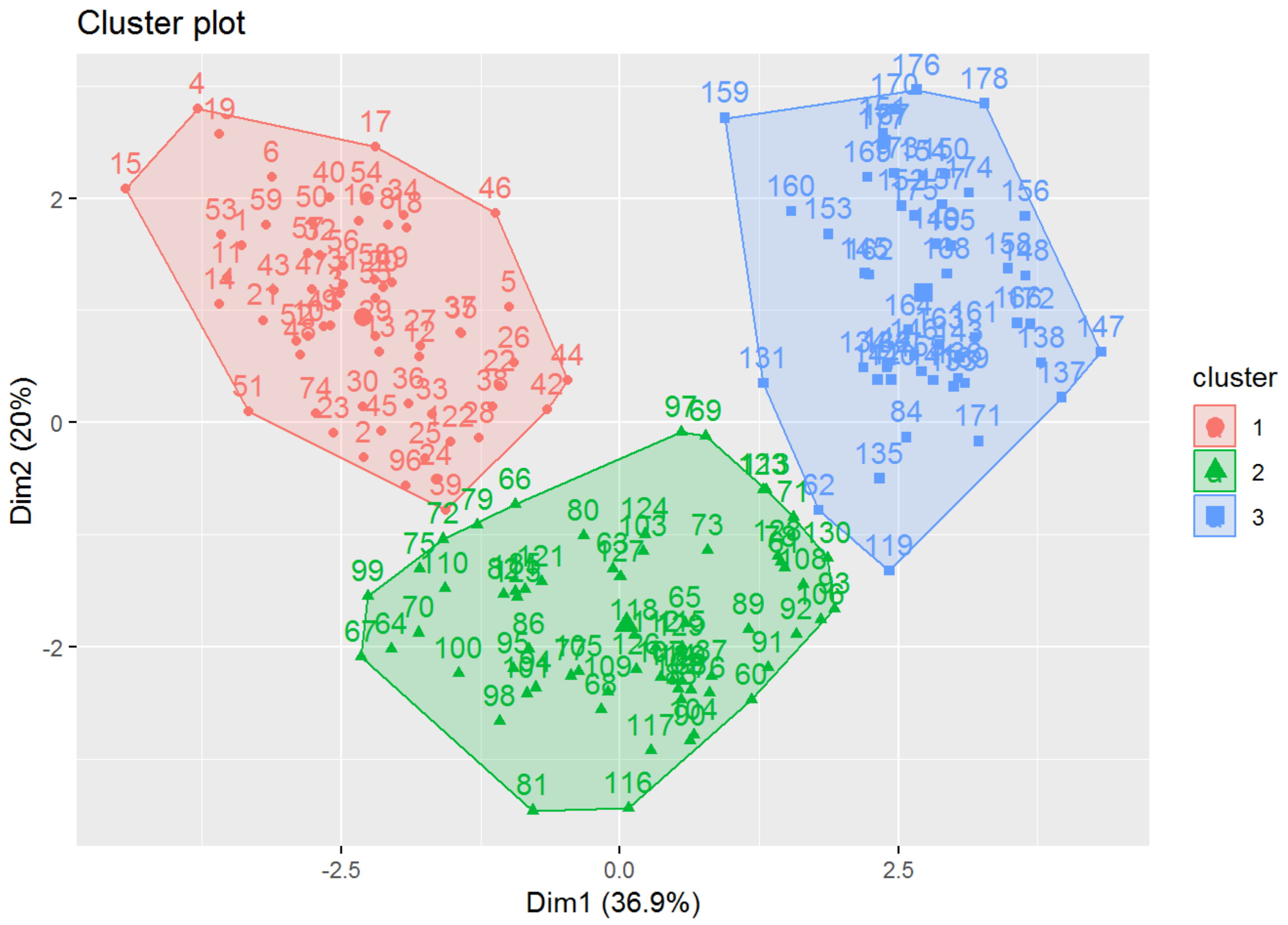


```
clusplot(wine_datasetScaled, wines_km.out$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



These two components explain 56.94 % of the point variability.

```
fviz_cluster(wines_km.out, data = wine_datasetScaled)
```



```
wine_datasetScaled %>%
  mutate(Cluster = wines_km.out$cluster) %>%
  group_by(Cluster) %>%
  summarise_all("mean")
```

```
## # A tibble: 3 x 14
##   Cluster Alcohol Malic_Acid    Ash Ash_Alcanity Magnesium Total_Phenols
##     <int>   <dbl>      <dbl>  <dbl>        <dbl>     <dbl>         <dbl>
## 1       1   0.833     -0.298  0.342       -0.679    0.615         0.883
## 2       2  -0.923     -0.412 -0.492        0.199   -0.552        -0.0758
## 3       3   0.164      0.888  0.211        0.572   -0.0440       -0.977
## # ... with 7 more variables: Flavanoids <dbl>, Nonflavanoid_Phenols <dbl>,
## #   Proanthocyanins <dbl>, Color_Intensity <dbl>, Hue <dbl>, OD280 <dbl>,
## #   Proline <dbl>
```

We have seen the wine data set has 3 cluster which separate the wine property.

Thank you!!!.