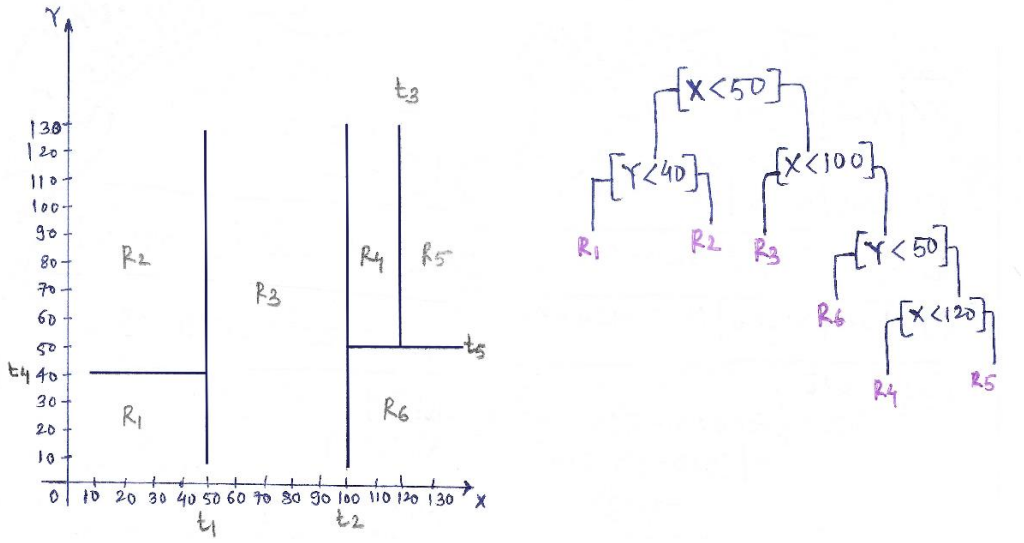
Recitation Exercises:

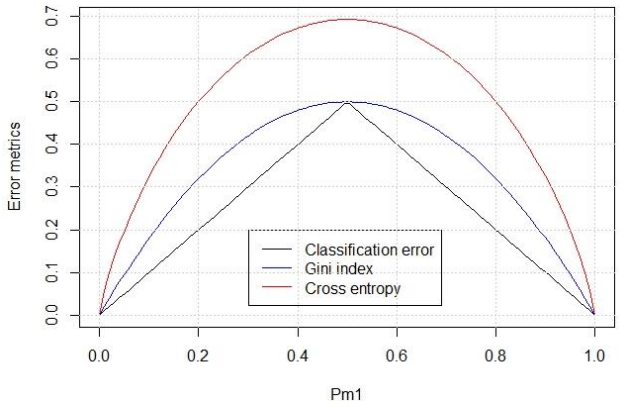
1.1 Chapter 8

1.1.1 Exercise 1



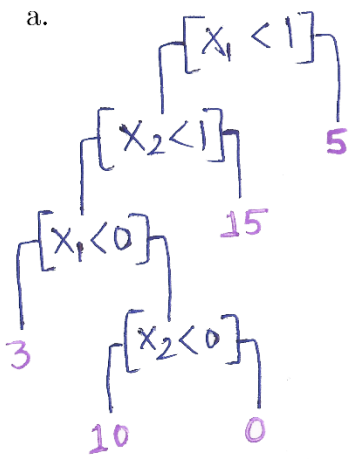
1.1.2 Exercise 3

```
p1 = seq(0 + 1e-06, 1 - 1e-06, length.out = 100)
p2 = 1 - p1

class_error = 1 - apply(rbind(p1, p2), 2, max)

gini_index = p1 * (1 - p1) + p2 * (1 - p2)

cross_entropy = -(p1 * log(p1) + p2 * log(p2))

plot(p1, class_error, type = "l", col = "black", xlab = "Pm1", ylab = "Error
metrics", ylim = c(min(c(class_error, gini_index, cross_entropy)), max(class_error,
gini_index, cross_entropy)))
lines(p1, gini_index, col = "blue")
lines(p1, cross_entropy, col = "red")
legend(0.3, 0.2, c("Classification error", "Gini index", "Cross entropy"), col =
c("black", "blue", "red"), lty = c(1, 1))
grid()
```

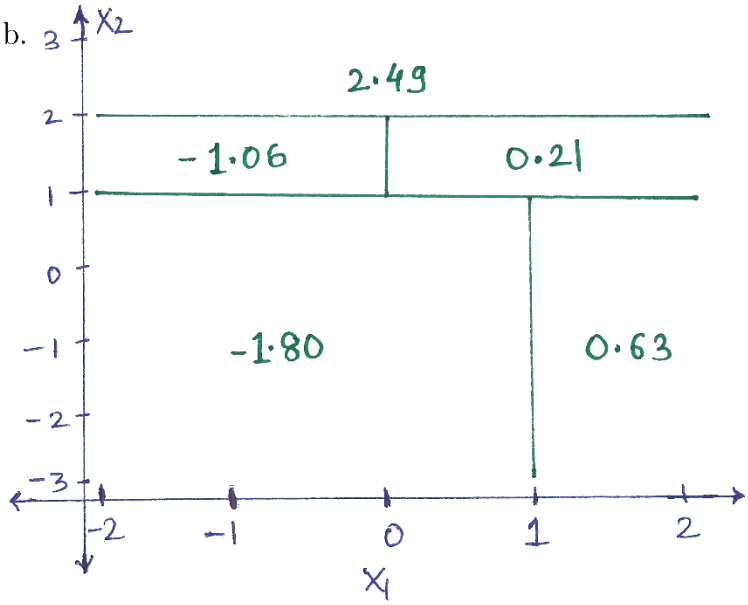


1.1.3 Exercise 4

a.



b.



1.1.4 Exercise 5

- Using Majority approach, X can be classified to the **Red Class**, as we have 6 out of 10 probabilities for Class Red (0.55, 0.6, 0.6, 0.65, 0.7, 0.75).
- Using Average probability approach, the average probability comes out to be, (0.1 + 0.15 + 0.2 + 0.2 + 0.55+ 0.6 + 0.6 + 0.65 + 0.7 + 0.75) / 10 = 0.45
  Thus, X can be classified to the **Green Class**.

1.2 Chapter 10

1.2.1 Exercise 1

a. Proof:

Given: $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$

LHS $= \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$

$= \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$ ...Expanding as $(a - b)^2 = a^2 - 2ab + b^2$

$= \sum_i \sum_j x^2{}_{ij} - 2 \sum_i \sum_j x_{ij} \bar{x}_{kj} + \sum_i \sum_j x^2{}_{ij}$

$= 2 \sum_i \sum_j x^2{}_{ij} - 2 |C_k| \sum_j \bar{x}^2{}_{kj}$ ... (1)

RHS $= 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$

$= 2 \sum_i \sum_j x^2{}_{ij} - 4 \sum_i \sum_j x_{ij} \bar{x}_{kj} + 2 \sum_i \sum_j \bar{x}^2{}_{kj}$ ...Expanding as $(a - b)^2 = a^2 - 2ab + b^2$

$= 2 \sum_i \sum_j x^2{}_{ij} - 2 |C_k| \sum_j \bar{x}^2{}_{kj}$ ... (2)

As from equations (1) and (2), LHS = RHS

Hence Proved.

b. Objective 10.11 can be seen at the step 2.b of Algorithm 10.1. It says, when we assign a new observation to the cluster whose centroid is closest, we decrease the right member of the identity. So, we decrease the left member of the identity which is our objective. Overall the identity 10.11 shows that minimizing the sum of the squared Euclidean distance for each cluster is the same as minimizing the within-cluster variance for each cluster.

1.2.2 Exercise 2

Following calculations describes how Single and Complete linkage is performed:

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 0.3 & 0.4 & 0.7 \\ 2 & 0.3 & 0 & 0.5 & 0.8 \\ 3 & 0.4 & 0.5 & 0 & 0.45 \\ 4 & 0.7 & 0.8 & 0.45 & 0 \end{array}$$

Complete Linkage

smallest inter-cluster dist = 0.3, clustering point 1 and 2

$$\begin{array}{c|ccc} & [1,2] & 3 & 4 \\ [1,2] & 0 & 0.5 & 0.8 \\ 3 & 0.5 & 0 & 0.45 \\ 4 & 0.8 & 0.45 & 0 \end{array}$$

$\rightarrow d([1,2], 3)$
$= \max[d(1,3), d(2,3)]$
$= \max[0.4, 0.5] = \underline{0.5}$

$\rightarrow d([1,2], 4)$
$= \max[d(1,4), d(2,4)]$
$= \max[0.7, 0.8] = \underline{0.8}$

Single Linkage

smallest inter-cluster dis = 0.3, clustering pair 1 and 2

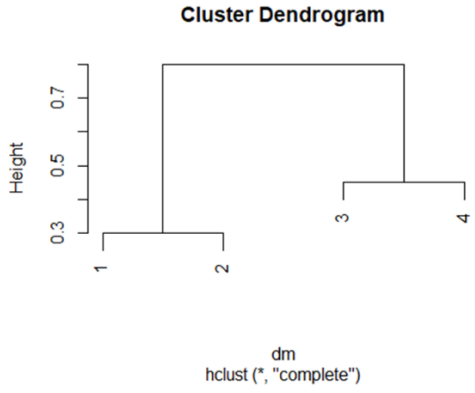
$$\begin{array}{c|ccc} & [1,2] & 3 & 4 \\ [1,2] & 0 & 0.4 & 0.7 \\ 3 & 0.4 & 0 & 0.45 \\ 4 & 0.7 & 0.45 & 0 \end{array}$$

$\rightarrow d([1,2], 3)$
$= \min[d(1,3), d(2,3)]$
$= \min[0.4, 0.5] = \underline{0.4}$

$\rightarrow d([1,2], 4)$
$= \min[d(1,2), d(2,4)]$
$= \min[0.7, 0.8] = \underline{0.7}$

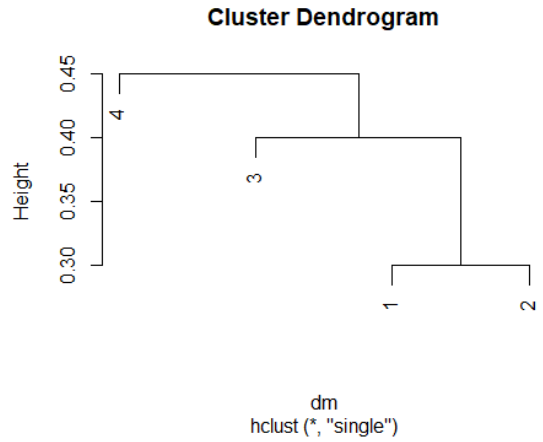
a.

```
dm = as.dist(matrix(c(0, 0.3, 0.4, 0.7,
                      0.3, 0, 0.5, 0.8,
                      0.4, 0.5, 0.0, 0.45,
                      0.7, 0.8, 0.45, 0.0), nrow = 4))

# Complete Linkage
plot(hclust(dm, method = "complete"))
```



b.

```
# Single Linkage
plot(hclust(dm, method = "single"))
```

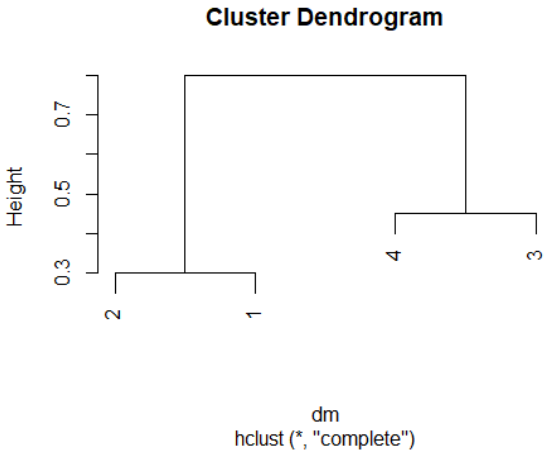


c. In Cluster 1: (1,2) and in Cluster 2: (3,4)

d. In Cluster 1: (4) and in Cluster 2: (3,(1,2))

e.

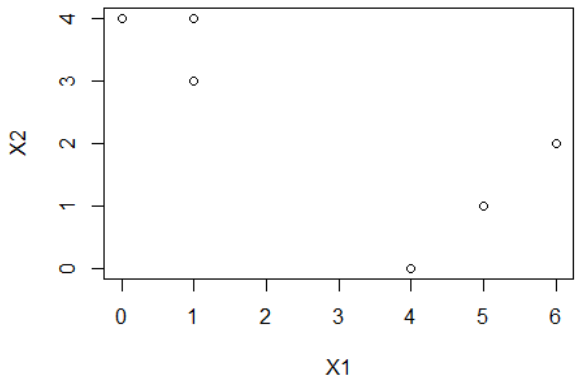
```
# Swapping the positions
plot(hclust(dm, method = "complete"), labels = c(2,1,4,3))
```



1.2.3 Exercise 3

a.

```
# Plotting the Observations
x = cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
plot(x[,1], x[,2], xlab = "X1", ylab = "X2")
```



b.

```r
# Radomly assigning a cluster to each observation
set.seed(1)
labels = sample(2, nrow(x), replace = T)
labels

## [1] 1 2 1 1 2 1

plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2, xlab = "X1", ylab
= "X2")
```

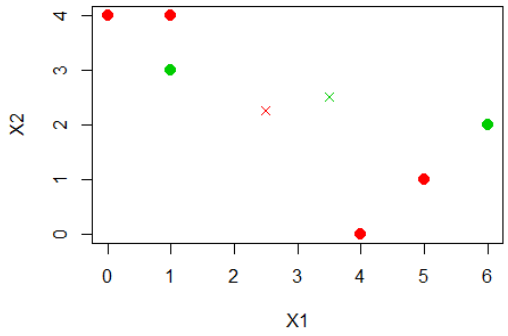
c. Centroid for Green Cluster

$$= \bar{x}_{11} = \frac{1}{3}(0+4+5) = 3$$
$$= \bar{x}_{12} = \frac{1}{3}(4+0+1) = \frac{5}{3} = 1.67$$
$$\text{Centroid}_{\text{green}} = (3, 1.67)$$

Centroid for Red Cluster

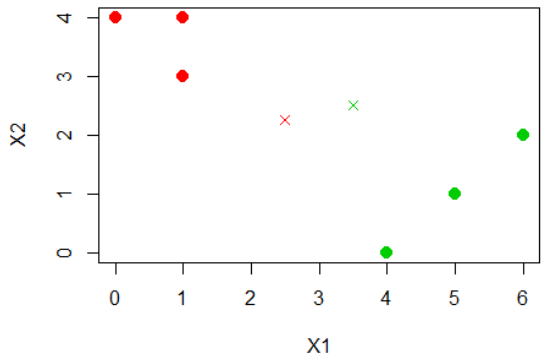
$$= \bar{x}_{21} = \frac{1}{3}(1+1+6) = \frac{8}{3} = 2.67$$
$$= \bar{x}_{22} = \frac{1}{3}(2+4+3) = 3$$
$$\text{Centroid}_{\text{red}} = (2.67, 3)$$

```r
# Computing Centroid for each cluster
centroid1 = c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 = c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2, xlab = "X1", ylab =
"X2")
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



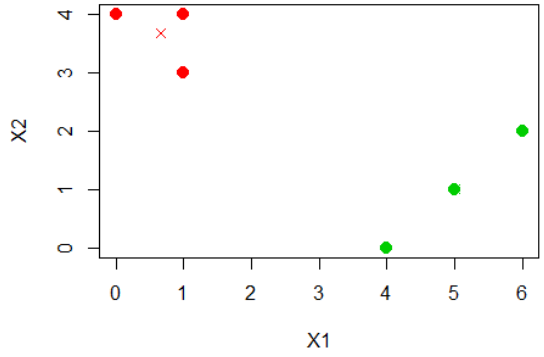
d.

```r
# Assigning each observation to the centroid to which it is closest, in terms
of Euclidean distance
labels = c(1, 1, 1, 2, 2, 2)
plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2, xlab = "X1", ylab
= "X2")
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



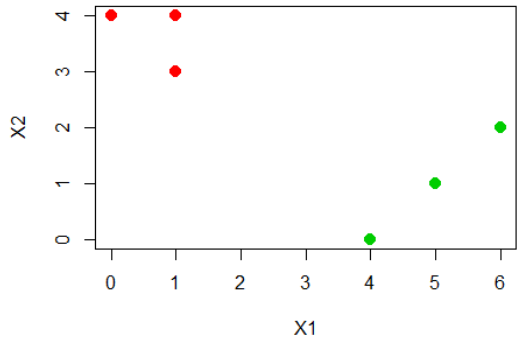
e.

```
# Repeating Steps (c) and (d) until the answers obtained stop changing
centroid1 = c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 = c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2, xlab = "X1", ylab =
"X2")
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



f.

```
# Coloring the observations according to the clusters labels obtained
plot(x[, 1], x[, 2], col=(labels + 1), pch = 20, cex = 2, xlab = "X1", ylab =
"X2")
```



1.2.4 Exercise 4

a. Given clusters: {1,2,3} and {4,5}

Complete Linkage:

= d[{1,2,3} ,{4,5}]

= max [d(1,4), d(1,5), d(2,4), d(2,5), d(3,4), d(3,5)] ... (1)

Single Linkage:

$= d[\{1,2,3\}, \{4,5\}]$

$= \min [d(1,4), d(1,5), d(2,4), d(2,5), d(3,4), d(3,5)]$ ... (2)

Now from (1) and (2) we have following two cases:

Case 1: If All the distances are **not** equal

So, situation (1) will give us a bigger value as it picks the maximum value and situation (2) will give us a smaller value as it picks the minimum value.

Hence, in this case **Complete Linkage will fuse at a higher height**.

Case 2: If All the distances are equal

As all the distances are equal both situation (1) and (2) will give us the same value.

Hence, in this case both **Complete and Single Linkage will fuse at the same height**.

b. Given clusters: {5} and {6}

As there are only single points in the clusters the dissimilarity will be the same for both Single and Complete Linkage and thus **both will fuse at the same height**.

1.2.5 Exercise 6

a. The 1st Principal Component explains 10% of the variation, which means, only 10% of the total information in the gene data is preserved, while the remaining 90% is lost by projecting the tissue sample observations onto the 1st Principal Component. This can also be stated as, the remaining 90% of variance in gene data is not explained by first principal component.

b. Given the flaw shown in pre-analysis of a time-wise linear trend amongst the tissue samples' first principal component, I would suggest to include the machine used (A vs B) as a feature of the data set. This should enhance the PVE of the first principal component before applying the two-sample t-test.

c.

```
set.seed(16)
Control = matrix(rnorm(50 * 1000), ncol = 50)
Treatment = matrix(rnorm(50 * 1000), ncol = 50)
X = cbind(Control, Treatment)
X[1, ] = seq(-18, 18 - .36, .36) # linear trend in one dimension
pr.out = prcomp(scale(X))
summary(pr.out)$importance[, 1]
```

```
##     Standard deviation Proportion of Variance  Cumulative Proportion
##               3.162982               0.100040               0.100040
```

We have 10.004% variance explained by the first principal component. Now, adding in A vs B via 10 vs 0 encoding.

```
X = rbind(X, c(rep(10, 50), rep(0, 50)))
pr.out = prcomp(scale(X))
summary(pr.out)$importance[, 1]
```

```
##     Standard deviation Proportion of Variance  Cumulative Proportion
##               3.409353               0.116240               0.116240
```

Now we have 11.624% variance explained by the first principal component. That's an improvement of 1.62%.

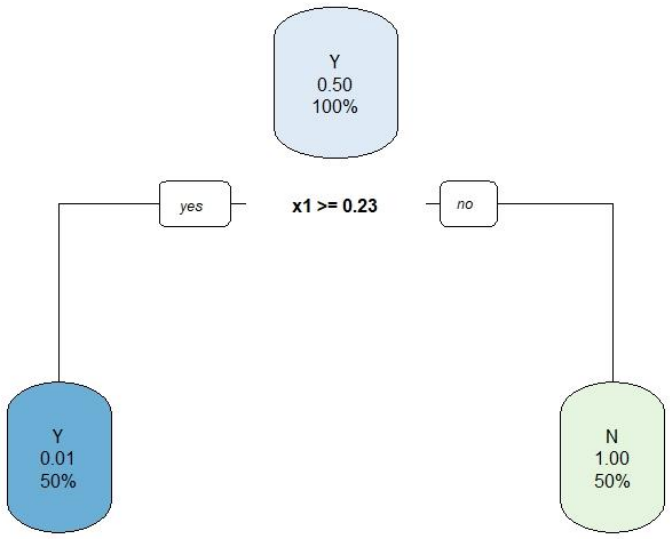
2 Practicum Problems:

2.1 Problem 1

```
library(rpart)
library(rpart.plot)


gini = function(p) {
  gini.index = 2 * p * (1 - p)
  return (gini.index)
}

entropy = function(p) {
  entropy = (p * log(p) + (1 - p) * log(1 - p))
  return (entropy)
}

df1 = data.frame(x1 = rnorm(150,5,2),class = rep("Y",150))
df2 = data.frame(x1 = rnorm(150,-5,2),class = rep("N",150))
dataset1 = rbind(df1,df2)

fit1 = rpart(class ~ x1, method="class", data = dataset1)
printcp(fit1)  # cp = cost complexity parameter in CART
```

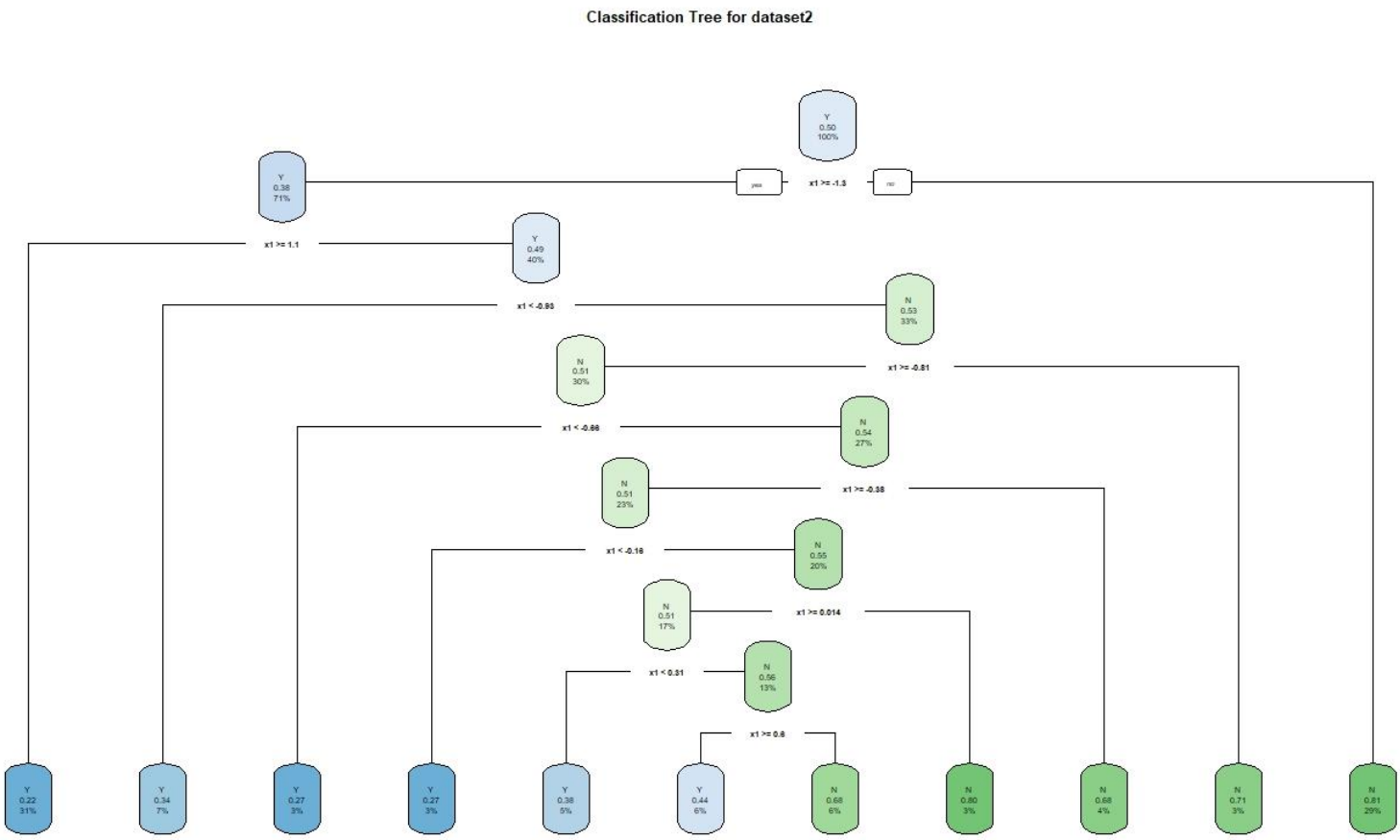
```
##
## Classification tree:
## rpart(formula = class ~ x1, data = dataset1, method = "class")
##
## Variables actually used in tree construction:
## [1] x1
##
## Root node error: 150/300 = 0.5
##
## n= 300
##
##         CP nsplit rel error   xerror      xstd
## 1 0.99333      0 1.0000000 1.120000 0.0573178
## 2 0.01000      1 0.0066667 0.013333 0.0093966
```

```
rpart.plot(fit1, main = "Classification Tree for dataset1")
```



In this case the threshold value for 1st split is 0.23. and the tree has 3 nodes 1 Root and 2 leaf. As in this case our dataset is created using normal distribution where 150 data samples are created with mean = 5 and standard deviation = 2 and 150 data sample are created with mean = -5 and standard deviation = 2 we get exact 50-50% class distribution.

```
p = c(.50, 0.01, 1)
gini_values = sapply(p, gini)
cat("Gini Values for dataset1: ",gini_values)

## Gini Values for dataset1:  0.5 0.0198 0

entropy_values = sapply(p, entropy)
cat("Entropy Values for dataset1: ",entropy_values)

## Entropy Values for dataset1:  -0.6931472 -0.05600153 NaN

df3 = data.frame(x1 = rnorm(250,1,2),class = rep("Y",250))
df4 = data.frame(x1 = rnorm(250,-1,2),class = rep("N",250))
dataset2 = rbind(df3,df4)

fit2 = rpart(class ~ x1, method="class", data = dataset2)
rpart.plot(fit2, main = "Classification Tree for dataset2")
```



This tree has 21 nodes and on pruning we get 3 nodes. Here our dataset is created with rnorm values of (1,2) and (-1,2) against the (5,2),(-5,2) in the previous case, we are getting poor splits as compared to the previous results. Here, a greater number of nodes are created because there are more values in a smaller range, which leads to overlapping thus, a greater number of split nodes are created. This may lead to overfitting when unseen data appears and hence we prune to tree to avoid this problem.

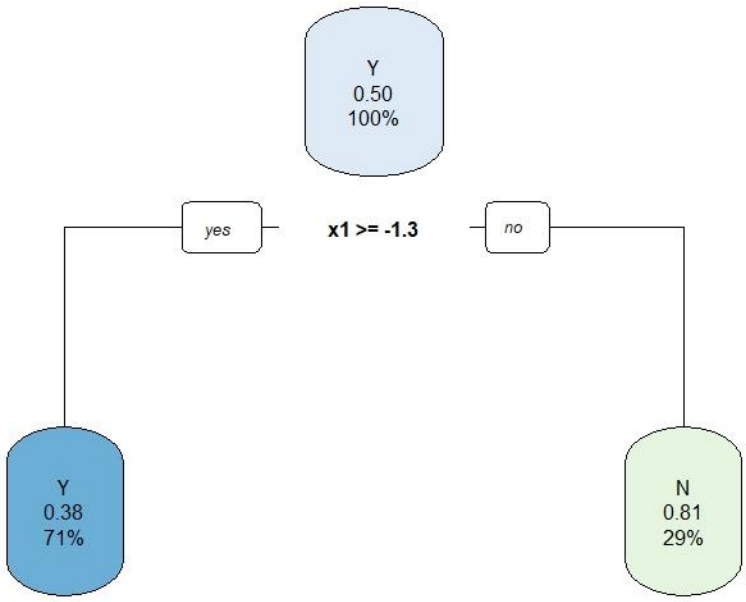
```
p = c(.50, 0.38,
0.49,0.53,0.51,0.54,0.51,0.55,0.51,0.56,0.22,0.34,0.27,0.27,0.38,0.44,0.68,0.80,0.68,0.71
,0.81)
gini_values = sapply(p, gini)
cat("Gini Values for dataset2: ",gini_values)
```

```
## Gini Values for dataset2:  0.5 0.4712 0.4998 0.4982 0.4998 0.4968 0.4998 0.495 0.4998
0.4928 0.3432 0.4488 0.3942 0.3942 0.4712 0.4928 0.4352 0.32 0.4352 0.4118 0.3078
```

```
entropy_values = sapply(p, entropy)
cat("Entropy Values for dataset2: ",entropy_values)
```

```
## Entropy Values for dataset2:  -0.6931472 -0.6640641 -0.6929472 -0.6913461 -0.6929472 -
0.6899438 -0.6929472 -0.6881388 -0.6929472 -0.6859298 -0.526908 -0.6410355 -0.5832588 -
0.5832588 -0.6640641 -0.6859298 -0.6268695 -0.5004024 -0.6268695 -0.6021517 -0.486223
```

```
# Pruning
fit2_prune = prune.rpart(fit2, cp = 0.1)
rpart.plot(fit2_prune, main = "Classification Tree for dataset2 after Pruning")
```



On Pruning the previous tree with complexity parameter, we get this above tree which has 3 nodes.

2.2 Problem 2

```
library(readr)
library(data.table)
library(corrplot)

## corrplot 0.84 loaded

wn_URL = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
wn_data = fread(wn_URL, header = FALSE)
wn_Header = c("class","Alcohol","Malic_Acid","Ash","Alcalinity","Magnesium",
"Total_Phenols","Flavanoids","Nonflavanoid" ,"Proanthocyanins","Color_Intensity", "Hue",
"OD280/OD315", "Proline")
colnames(wn_data) = wn_Header

apply(wn_data[,-1], 2, mean)

##         Alcohol      Malic_Acid             Ash      Alcalinity
##      13.0006180       2.3363483       2.3665169      19.4949438
##       Magnesium   Total_Phenols      Flavanoids    Nonflavanoid
##      99.7415730       2.2951124       2.0292697       0.3618539
## Proanthocyanins Color_Intensity             Hue     OD280/OD315
##       1.5908989       5.0580899       0.9574494       2.6116854
##         Proline
##     746.8932584
```

As the means of the attributes differ, we need to apply Scaling.

```r
wn_pca = prcomp(wn_data[,-1], scale = TRUE, center = TRUE)
summary(wn_pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.169 1.5802 1.2025 0.95863 0.92370 0.80103 0.74231
## Proportion of Variance 0.362 0.1921 0.1112 0.07069 0.06563 0.04936 0.04239
## Cumulative Proportion  0.362 0.5541 0.6653 0.73599 0.80162 0.85098 0.89337
##                           PC8     PC9   PC10    PC11    PC12    PC13
## Standard deviation     0.59034 0.53748 0.5009 0.47517 0.41082 0.32152
## Proportion of Variance 0.02681 0.02222 0.0193 0.01737 0.01298 0.00795
## Cumulative Proportion  0.92018 0.94240 0.9617 0.97907 0.99205 1.00000
```

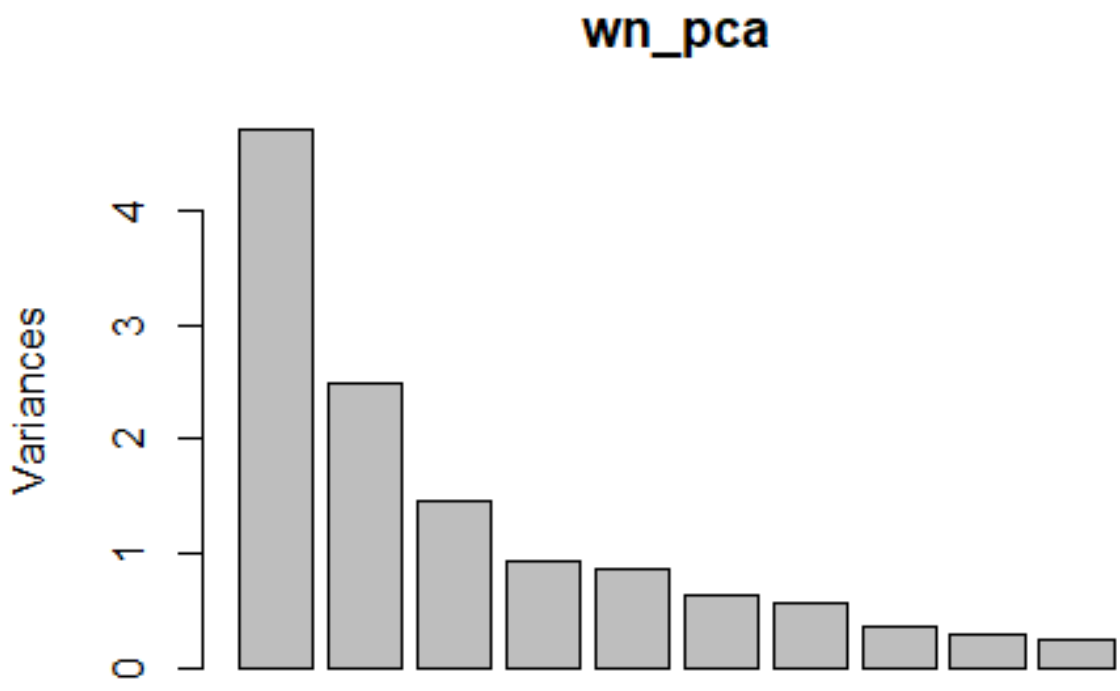
```r
wn_pca$center
```

```
##         Alcohol      Malic_Acid             Ash      Alcalinity
##      13.0006180       2.3363483       2.3665169      19.4949438
##       Magnesium   Total_Phenols      Flavanoids    Nonflavanoid
##      99.7415730       2.2951124       2.0292697       0.3618539
## Proanthocyanins Color_Intensity             Hue     OD280/OD315
##       1.5908989       5.0580899       0.9574494       2.6116854
##         Proline
##     746.8932584
```

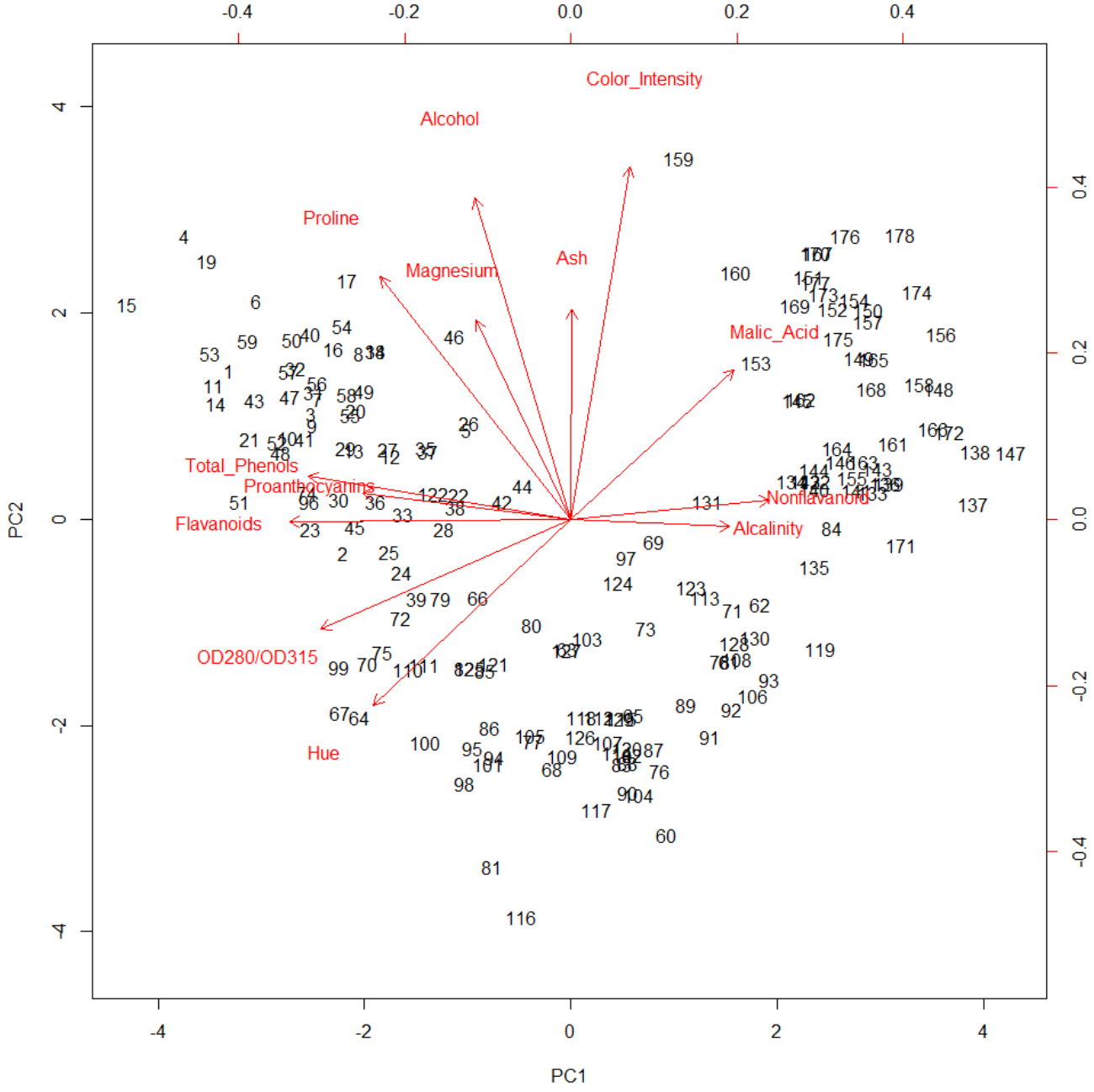
```r
wn_pca$scale
```

```
##         Alcohol      Malic_Acid             Ash      Alcalinity
##       0.8118265       1.1171461       0.2743440       3.3395638
##       Magnesium   Total_Phenols      Flavanoids    Nonflavanoid
##      14.2824835       0.6258510       0.9988587       0.1244533
## Proanthocyanins Color_Intensity             Hue     OD280/OD315
##       0.5723589       2.3182859       0.2285716       0.7099904
##         Proline
##     314.9074743
```

```r
plot(wn_pca)
```

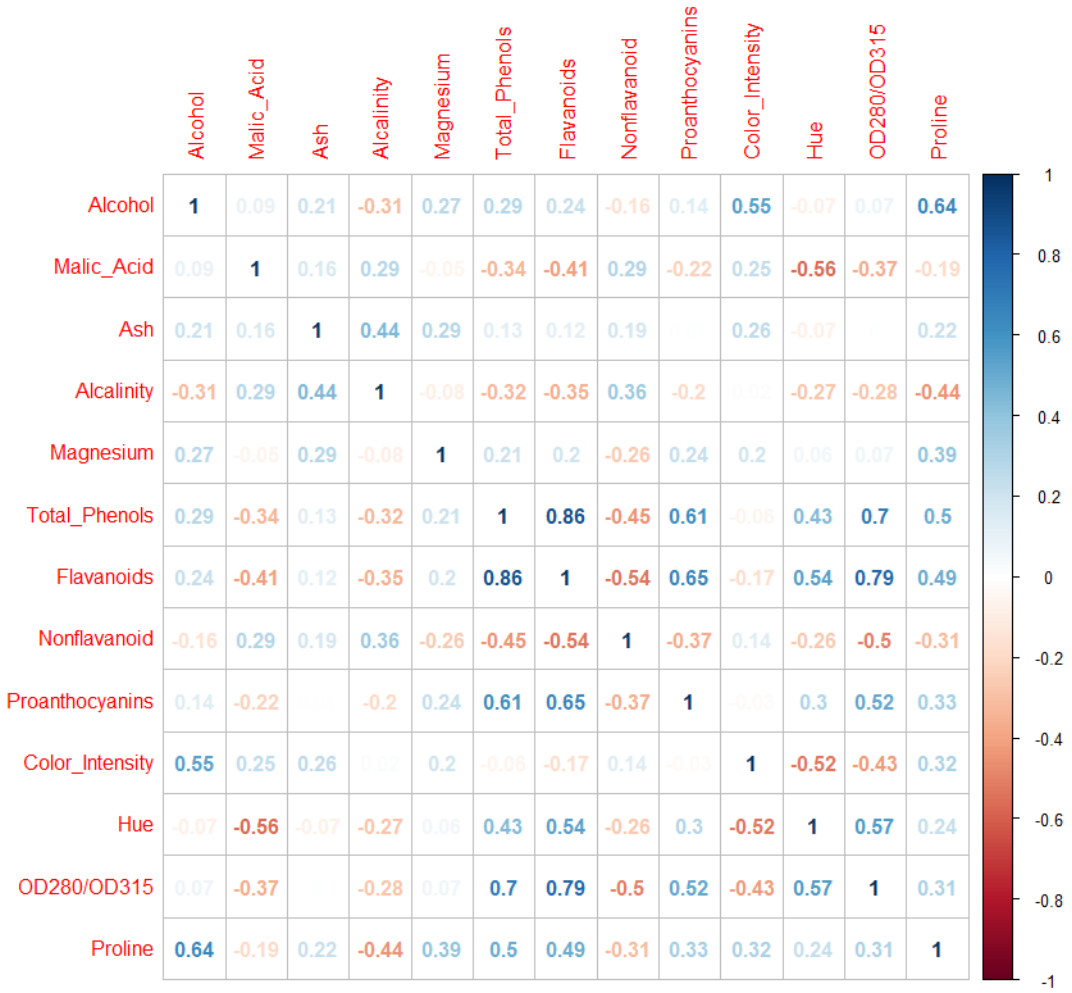


```
biplot(wn_pca, scale = 0)
```



As from the above biplot, we can see that the feature opposite to Hue is Malic Acid which implies that they are inversely proportional which means if the value of one increases then the other decreases. This is further supported with the following correlation plot.

```
corrplot(cor(wn_data[,-1]), method="number")
```

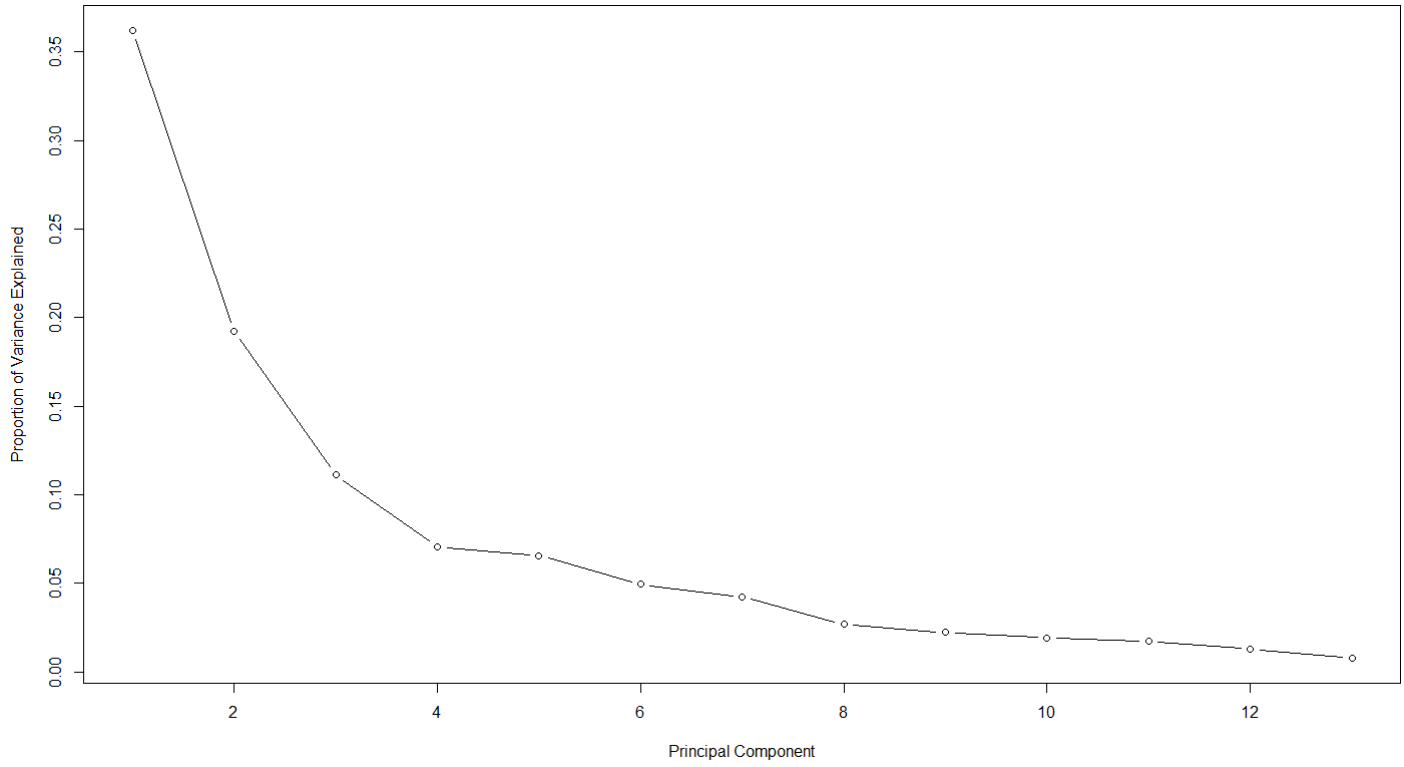



```
cat("Correlation between Malic_Acid and Hue: ",cor(wn_data$Malic_Acid, wn_data$Hue))
```

```
## Correlation between Malic_Acid and Hue:  -0.5612957
```

```
std_dev = wn_pca$sdev
pr_var = std_dev^2
prop_varex = pr_var/sum(pr_var)
```

```r
plot(prop_varex, xlab = "Principal Component", ylab = "Proportion of Variance
Explained",type = "b")
```




```r
cat("Percentage of total variance explained by PC1 and PC2 = ",
(prop_varex[1]+prop_varex[2])*100,"%")
```

```
## Percentage of total variance explained by PC1 and PC2 =  55.40634 %
```

2.3 Problem 3

```r
library("factoextra")
```

```
## Loading required package: ggplot2
```

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

```r
data = data.frame(USArrests)
states = row.names(data)
states
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
```

```
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"

names(data)

## [1] "Murder"   "Assault"  "UrbanPop" "Rape"

# Before Scaling
head(data)

##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7

summary(data)

##      Murder          Assault         UrbanPop          Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
##  Median : 7.250   Median :159.0   Median :66.00   Median :20.10
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

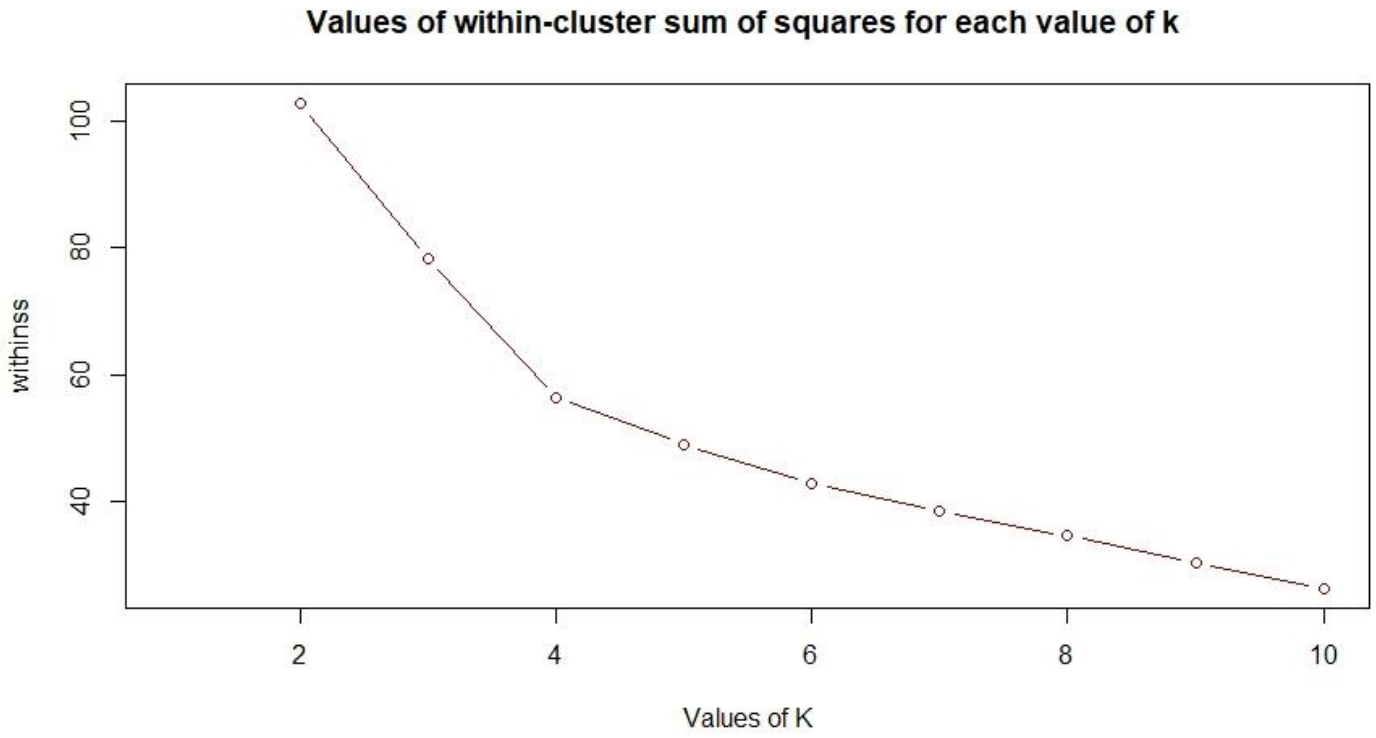
As the means of the attributes differ, we need to apply Scaling.

```
# Scaling
new_data = scale(data)

# After Scaling
head(new_data)

##                Murder   Assault   UrbanPop          Rape
## Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
## Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
## Arizona    0.07163341 1.4788032  0.9989801  1.042878388
## Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
## California 0.27826823 1.2628144  1.7589234  2.067820292
## Colorado   0.02571456 0.3988593  0.8608085  1.864967207

summary(new_data)

##      Murder           Assault           UrbanPop            Rape
##  Min.   :-1.6044   Min.   :-1.5090   Min.   :-2.31714   Min.   :-1.4874
##  1st Qu.:-0.8525   1st Qu.:-0.7411   1st Qu.:-0.76271   1st Qu.:-0.6574
##  Median :-0.1235   Median :-0.1411   Median : 0.03178   Median :-0.1209
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.7949   3rd Qu.: 0.9388   3rd Qu.: 0.84354   3rd Qu.: 0.5277
##  Max.   : 2.2069   Max.   : 1.9948   Max.   : 1.75892   Max.   : 2.6444

withinss = c()
i = 2
for (k in 2:10){
  set.seed(123)
```

```
  model = kmeans(new_data, k, nstart = 25)
  withinss[i] = model$tot.withinss
  cat(" k = ",k,"Withinss = ",withinss[i],"\n")
  i = i + 1
}
```

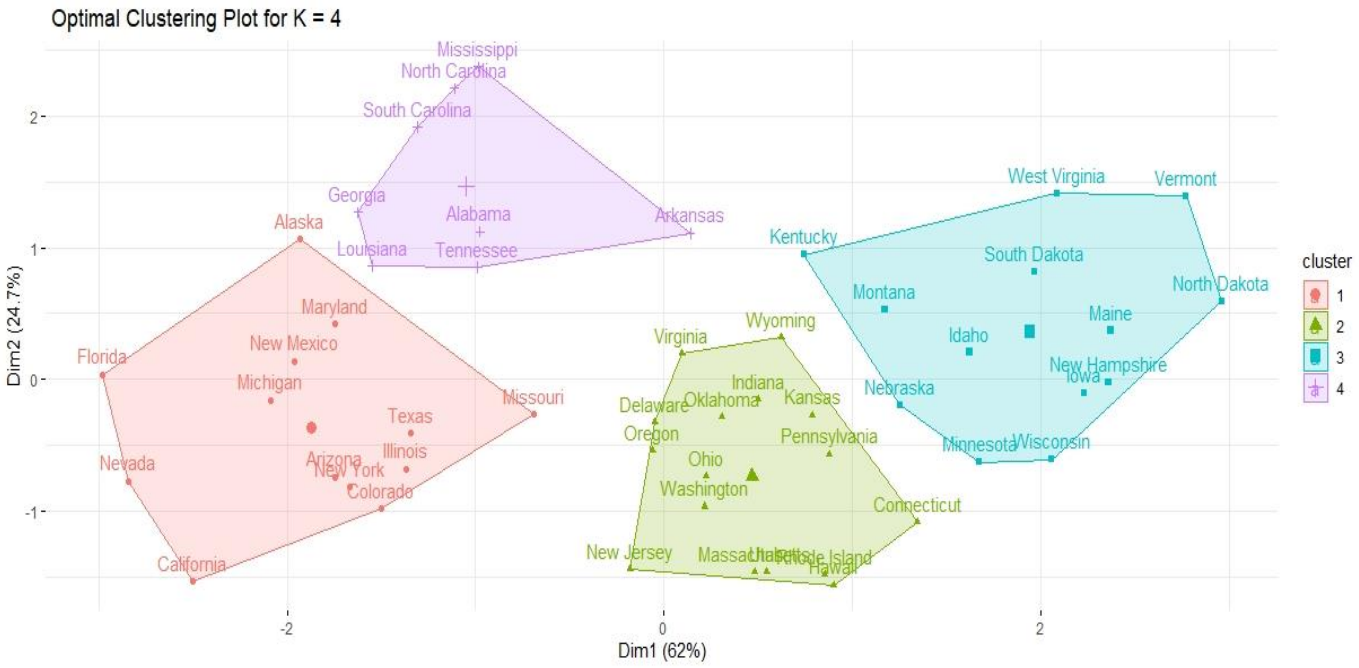
```
##  k =  2 Withinss =  102.8624
##  k =  3 Withinss =  78.32327
##  k =  4 Withinss =  56.40317
##  k =  5 Withinss =  48.9442
##  k =  6 Withinss =  42.83303
##  k =  7 Withinss =  38.25764
##  k =  8 Withinss =  34.44327
##  k =  9 Withinss =  30.17403
##  k =  10 Withinss =  26.18348
```

```
plot(withinss, xlab = "Values of K", main = "Values of within-cluster sum of squares for
each value of k ", type = "b", col = "dark red")
```



As we can see there is a sharp decrease at K = 4, and from there onwards the withinss keeps on decreasing, thus we select K = 4 as optimal value.

```
model = kmeans(new_data, 4, nstart = 25)
fviz_cluster(model, data = new_data,ggtheme = theme_minimal(),main = " Optimal Clustering
Plot for K = 4")
```



2.4 Problem 4

```
library(readr)
library(data.table)
library(corrplot)

## corrplot 0.84 loaded

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

wq_URL = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-white.csv"
wq_raw_data = fread(wq_URL, header = TRUE)
wq_data = wq_raw_data[,-12]

head(wq_data)

##    fixed acidity volatile acidity citric acid residual sugar chlorides
## 1:           7.0             0.27        0.36           20.7     0.045
## 2:           6.3             0.30        0.34            1.6     0.049
## 3:           8.1             0.28        0.40            6.9     0.050
## 4:           7.2             0.23        0.32            8.5     0.058
```

```
## 5:           7.2             0.23        0.32            8.5     0.058
## 6:           8.1             0.28        0.40            6.9     0.050
##    free sulfur dioxide total sulfur dioxide density   pH sulphates alcohol
## 1:                  45                  170  1.0010 3.00      0.45     8.8
## 2:                  14                  132  0.9940 3.30      0.49     9.5
## 3:                  30                   97  0.9951 3.26      0.44    10.1
## 4:                  47                  186  0.9956 3.19      0.40     9.9
## 5:                  47                  186  0.9956 3.19      0.40     9.9
## 6:                  30                   97  0.9951 3.26      0.44    10.1
```

```
summary(wq_data)
```

```
##  fixed acidity    volatile acidity  citric acid     residual sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free sulfur dioxide total sulfur dioxide
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0
##  Median :0.04300   Median : 34.00      Median :134.0
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0
##     density             pH          sulphates         alcohol
##  Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00
##  1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50
##  Median :0.9937   Median :3.180   Median :0.4700   Median :10.40
##  Mean   :0.9940   Mean   :3.188   Mean   :0.4898   Mean   :10.51
##  3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40
##  Max.   :1.0390   Max.   :3.820   Max.   :1.0800   Max.   :14.20
```

As the means of the attributes differ, we need to apply Scaling.

```
wq_data_scale = scale(wq_data)
```

```
head(wq_data_scale)
```

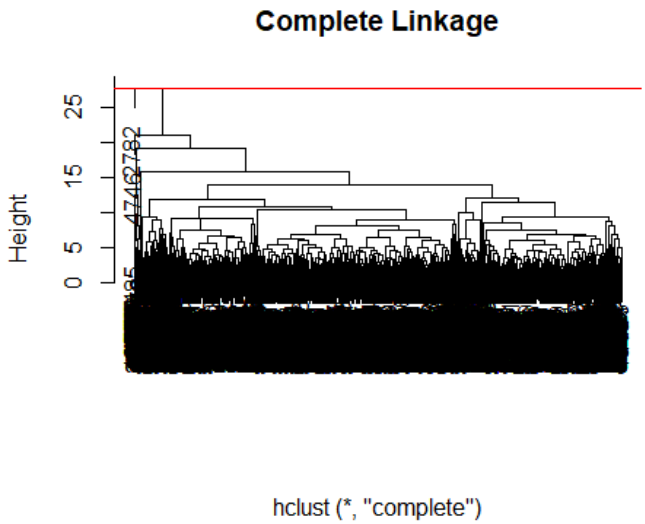
```
##      fixed acidity volatile acidity citric acid residual sugar   chlorides
## [1,]     0.1720794      -0.08176155  0.21325843      2.8210611 -0.03535139
## [2,]    -0.6574340       0.21587359  0.04799622     -0.9446688  0.14773200
## [3,]     1.4756004       0.01745016  0.54378284      0.1002720  0.19350284
## [4,]     0.4090832      -0.47860841 -0.11726599      0.4157258  0.55966962
## [5,]     0.4090832      -0.47860841 -0.11726599      0.4157258  0.55966962
## [6,]     1.4756004       0.01745016  0.54378284      0.1002720  0.19350284
##      free sulfur dioxide total sulfur dioxide      density          pH
## [1,]           0.5698734            0.7444890  2.331273996 -1.24679399
## [2,]          -1.2528907           -0.1496693 -0.009153237  0.73995309
## [3,]          -0.3121093           -0.9732363  0.358628185  0.47505348
## [4,]           0.6874711            1.1209768  0.525801559  0.01147916
## [5,]           0.6874711            1.1209768  0.525801559  0.01147916
## [6,]          -0.3121093           -0.9732363  0.358628185  0.47505348
##        sulphates    alcohol
## [1,] -0.34914861 -1.3930102
```

```
## [2,]  0.00134171 -0.8241915
## [3,] -0.43677119 -0.3366326
## [4,] -0.78726151 -0.4991523
## [5,] -0.78726151 -0.4991523
## [6,] -0.43677119 -0.3366326
```

```
summary(wq_data_scale)
```

```
##  fixed acidity      volatile acidity   citric acid      residual sugar
##  Min.   :-3.61998   Min.   :-1.9668   Min.   :-2.7615   Min.   :-1.1418
##  1st Qu.:-0.65743   1st Qu.:-0.6770   1st Qu.:-0.5304   1st Qu.:-0.9250
##  Median :-0.06492   Median :-0.1810   Median :-0.1173   Median :-0.2349
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.52758   3rd Qu.: 0.4143   3rd Qu.: 0.4612   3rd Qu.: 0.6917
##  Max.   : 8.70422   Max.   : 8.1528   Max.   :10.9553   Max.   :11.7129
##    chlorides       free sulfur dioxide total sulfur dioxide
##  Min.   :-1.6831   Min.   :-1.95848    Min.   :-3.0439
##  1st Qu.:-0.4473   1st Qu.:-0.72370    1st Qu.:-0.7144
##  Median :-0.1269   Median :-0.07691    Median :-0.1026
##  Mean   : 0.0000   Mean   : 0.00000    Mean   : 0.0000
##  3rd Qu.: 0.1935   3rd Qu.: 0.62867    3rd Qu.: 0.6739
##  Max.   :13.7417   Max.   :14.91679    Max.   : 7.0977
##     density                pH             sulphates
##  Min.   :-2.31280   Min.   :-3.10109   Min.   :-2.3645
##  1st Qu.:-0.77063   1st Qu.:-0.65077   1st Qu.:-0.6996
##  Median :-0.09608   Median :-0.05475   Median :-0.1739
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.69298   3rd Qu.: 0.60750   3rd Qu.: 0.5271
##  Max.   :15.02976   Max.   : 4.18365   Max.   : 5.1711
##     alcohol
##  Min.   :-2.04309
##  1st Qu.:-0.82419
##  Median :-0.09285
##  Mean   : 0.00000
##  3rd Qu.: 0.71974
##  Max.   : 2.99502
```

```
hc.complete = hclust (dist(wq_data_scale), method = "complete")
plot(hc.complete, main =" Complete Linkage ", xlab = "")
abline(h =tail(hc.complete$height,1), col="red")
```



```r
cat("Distance value where the two penultimate clusters merge =",
tail(hc.complete$height,1),"\n")
```

```
## Distance value where the two penultimate clusters merge = 27.73476
```

```r
hc.completeCut = cutree(hc.complete, 2)
table(hc.completeCut)
```

```
## hc.completeCut
##    1    2
## 4897    1
```

```r
wq_data2 = wq_data
wq_data2$Clusters = hc.completeCut
wq_data2 = dplyr::group_by(wq_data2,Clusters)
a = dplyr::summarise_each(wq_data2,funs(mean))
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

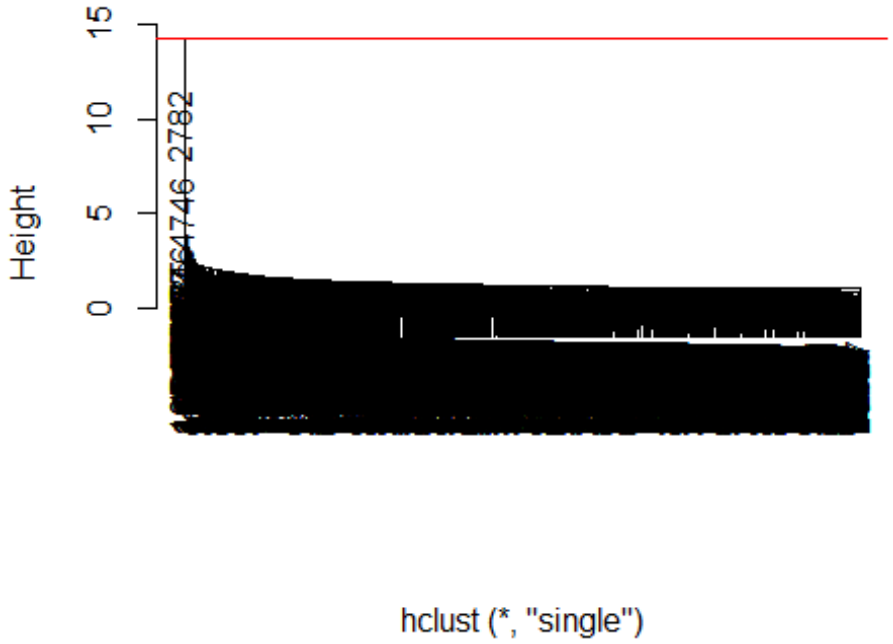
```r
print.data.frame(a)
```

```
##   Clusters fixed acidity volatile acidity citric acid residual sugar
## 1        1      6.854595        0.2781009   0.3341372       6.379283
## 2        2      7.800000        0.9650000   0.6000000      65.800000
##    chlorides free sulfur dioxide total sulfur dioxide   density       pH
## 1 0.04576659            35.31366             138.3562 0.9940182 3.188225
## 2 0.07400000             8.00000             160.0000 1.0389800 3.390000
##   sulphates  alcohol
## 1  0.489806 10.51402
## 2  0.690000 11.70000
```

```r
# Feature Means Difference
abs(a[2,-1]-a[1,-1])
```

```
##   fixed acidity volatile acidity citric acid residual sugar  chlorides
## 1     0.9454054        0.6868991   0.2658628       59.42072 0.02823341
##   free sulfur dioxide total sulfur dioxide   density        pH sulphates
## 1            27.31366             21.64376 0.0449618 0.2017746  0.200194
##    alcohol
## 1 1.185975
```

```r
hc.single = hclust (dist(wq_data_scale), method = "single")
plot(hc.single, main =" Single Linkage ", xlab = "")
abline(h =tail(hc.single$height,1), col="red")
```




```
cat("Distance value where the two penultimate clusters merge =",
tail(hc.single$height,1),"\n")

## Distance value where the two penultimate clusters merge = 14.25323

hc.singleCut = cutree(hc.single, 2)
table(hc.singleCut)

## hc.singleCut
##    1    2
## 4897    1

wq_data3 = wq_data
wq_data3$Clusters = hc.singleCut
wq_data3 = dplyr::group_by(wq_data3,Clusters)
b = dplyr::summarise_each(wq_data3,funs(mean))
print.data.frame(b)

##   Clusters fixed acidity volatile acidity citric acid residual sugar
## 1        1      6.854595        0.2781009   0.3341372       6.379283
## 2        2      7.800000        0.9650000   0.6000000      65.800000
##    chlorides free sulfur dioxide total sulfur dioxide   density       pH
## 1 0.04576659            35.31366             138.3562 0.9940182 3.188225
## 2 0.07400000             8.00000             160.0000 1.0389800 3.390000
##   sulphates  alcohol
## 1  0.489806 10.51402
## 2  0.690000 11.70000

# Feature Means Difference
abs(b[2,-1]-b[1,-1])

##   fixed acidity volatile acidity citric acid residual sugar  chlorides
## 1     0.9454054        0.6868991   0.2658628       59.42072 0.02823341
##   free sulfur dioxide total sulfur dioxide   density        pH sulphates
## 1            27.31366             21.64376 0.0449618 0.2017746  0.200194
```

```
##    alcohol
## 1 1.185975
```

As we can see, the Distance value where the two penultimate clusters merge for Complete Linkage is 27.73476 and the Distance value where the two penultimate clusters merge for Single Linkage is 14.25323. However, we are getting same number of observations in both cases, i.e. 1st Cluster contains 4897 observations and the 2nd Clusters contains 1 observation. Also, from Summary statistics, in both cases the feature `residual sugar` has the largest possible difference. Thus, complete linkage produces more balanced tree than single linkage.