Name:- PARTH RATHOD CWID:- A20458817 HW :- HW5 COURSE:- DPA CS571

Recitation Question 1 a)

```
knitr::include_graphics("1a-1.jpeg")
```

Chapter 12

1) a) To prove:-

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

where, $\bar{x}_j = \frac{1}{|C_k|}\sum_{i\in C_k} x_{ij}$

is mean of feature j in cluster C_k.

L.H.S

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

Since:- $(a-b)^2 = a^2 - 2ab + b^2$

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

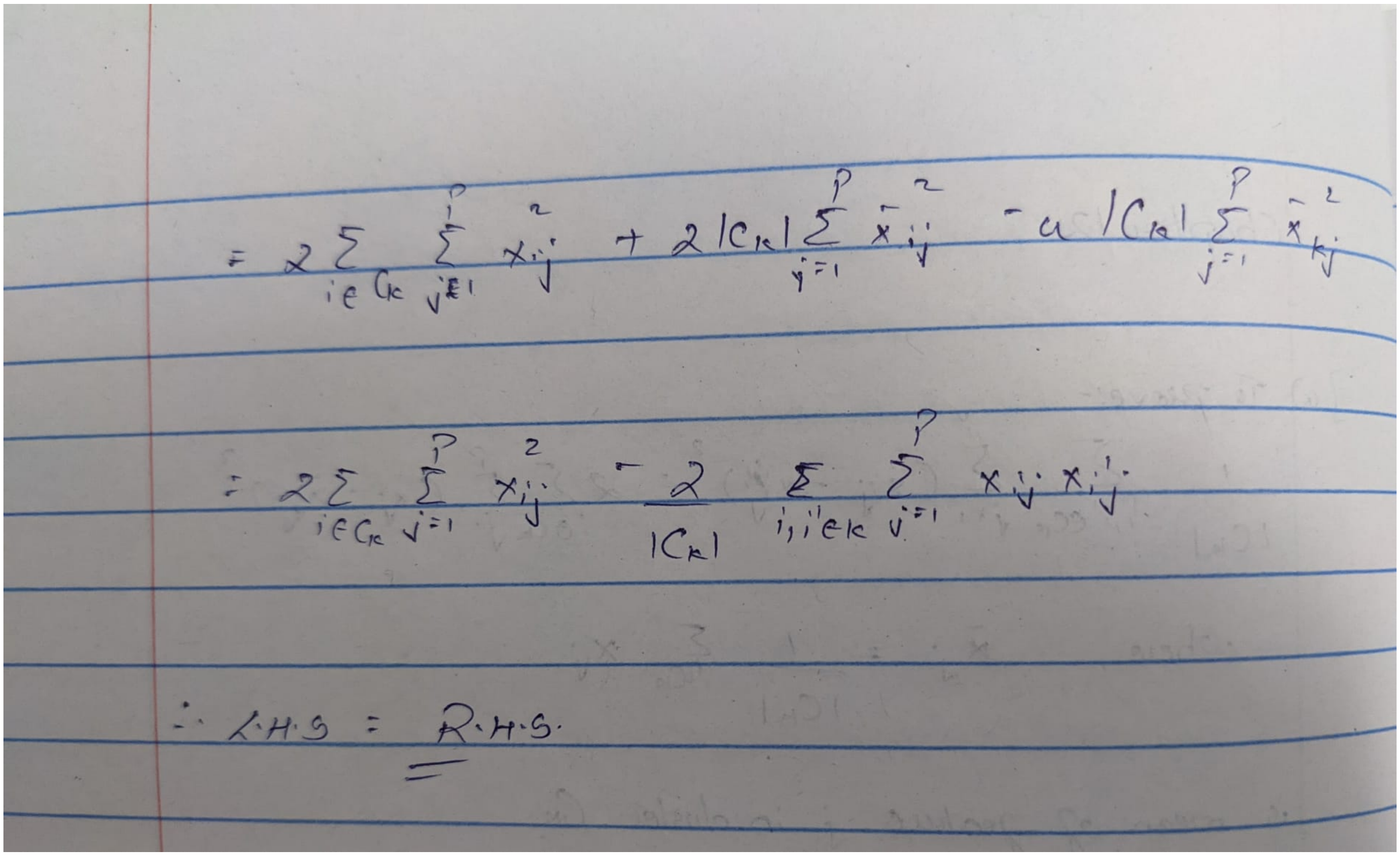
$$= \frac{2}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}x_{ij}^2 - \frac{2}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}x_{ij}\,x_{i'j}$$

R.H.S

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

$$= 2\sum_{i\in C_k}\sum_{j=1}^{p}x_{ij}^2 + 2\sum_{i\in C_k}\sum_{j=1}^{p}\bar{x}_{ij}^2 - 4\sum_{i\in C_k}\sum_{j=1}^{p}x_{ij}\,\bar{x}_{i'j}$$

```
knitr::include_graphics("1a-2.jpeg")
```



$$= 2\sum_{i\in C_k}\sum_{j=1}^{p} x_{ij}^2 + 2|C_k|\sum_{j=1}^{p}\bar{x}_{ij}^2 - 4|C_k|\sum_{j=1}^{p}\bar{x}_{kj}^2$$

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

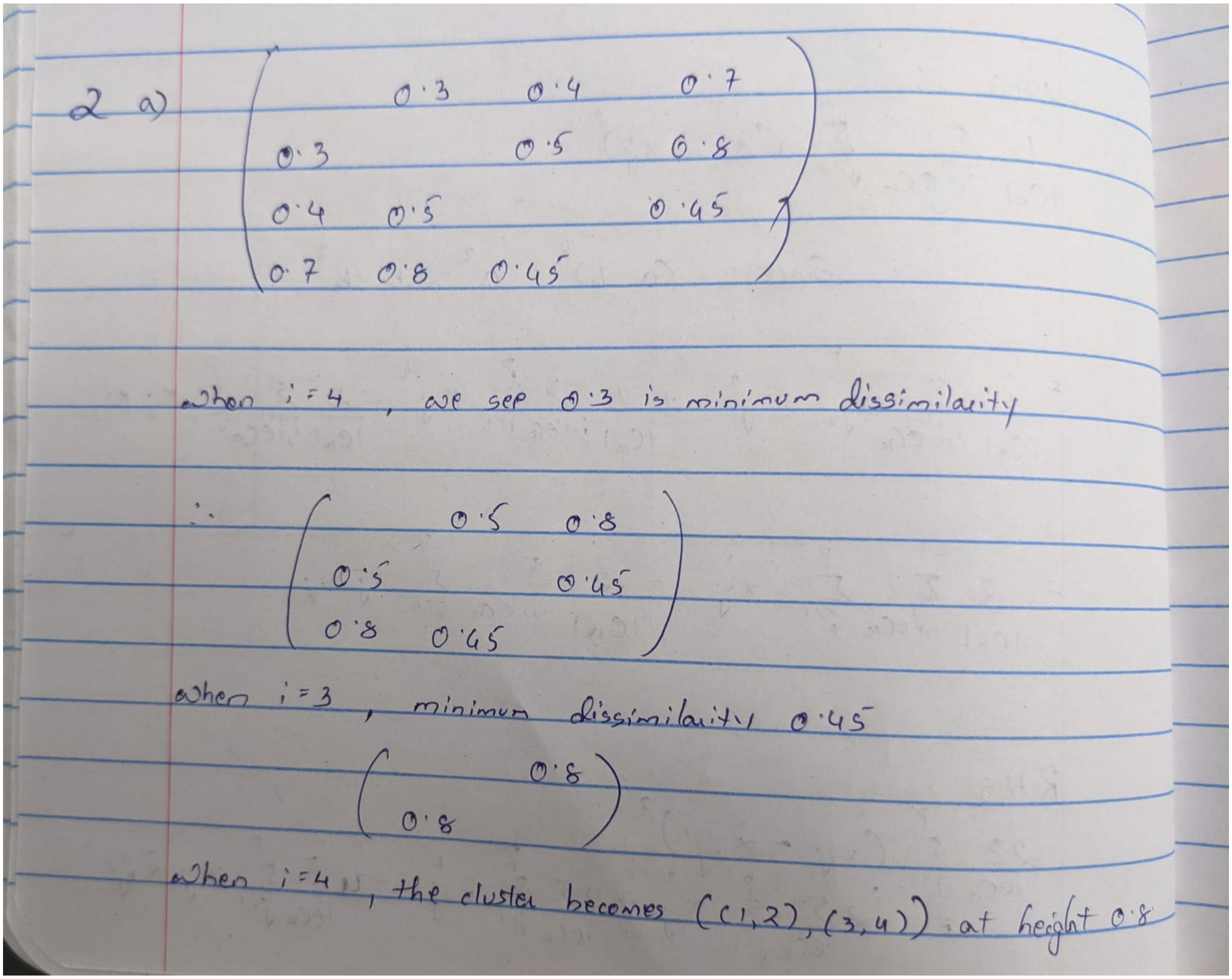
$\therefore$ L.H.S = R.H.S.

b.

In K-means clustering algorithm, at each iteration, an observation is assigned to its nearest cluster. Due to which after each iteration the value of RHS will decreases as this quantity is sum of squared distance of each observation from the cluster mean. Hence, in this way the k-means will decrease the objective in each iteration.

Question 2 a)

```
knitr::include_graphics("2a.jpeg")
```



2 a)

$$\begin{pmatrix} & 0.3 & 0.4 & 0.7 \\ 0.3 & & 0.5 & 0.8 \\ 0.4 & 0.5 & & 0.45 \\ 0.7 & 0.8 & 0.45 & \end{pmatrix}$$

when i = 4 , we see 0·3 is minimum dissimilarity

∴

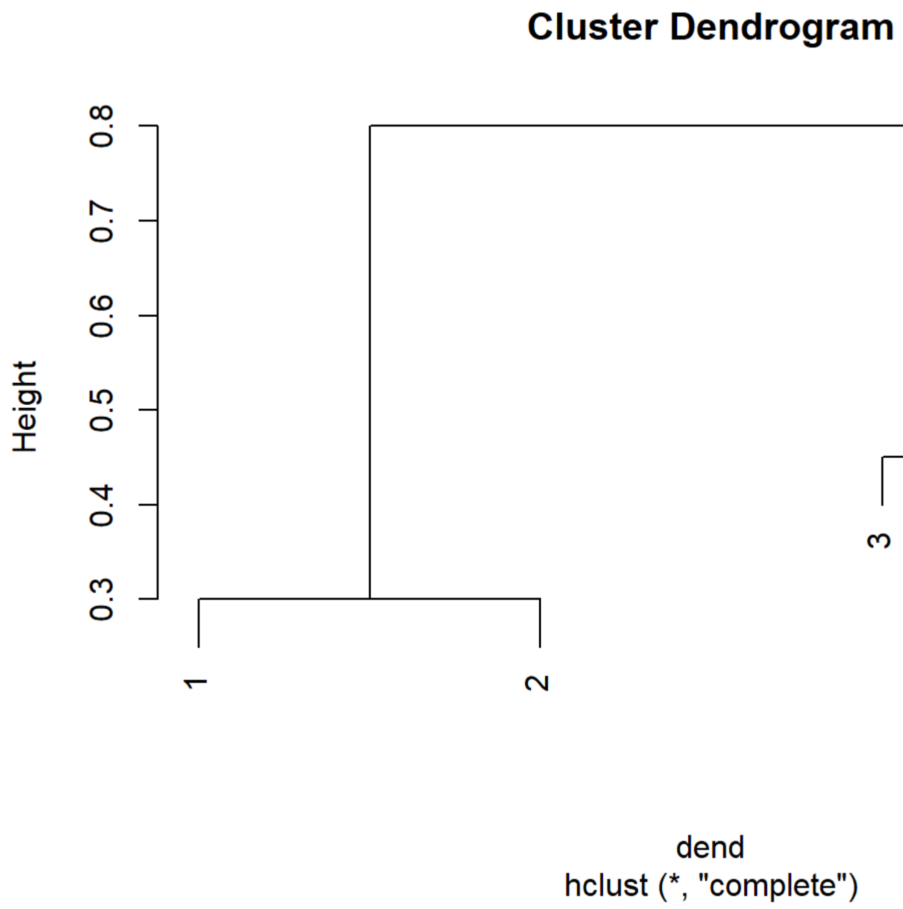
$$\begin{pmatrix} & 0.5 & 0.8 \\ 0.5 & & 0.45 \\ 0.8 & 0.45 & \end{pmatrix}$$

when i = 3 , minimum dissimilarity 0·45

$$\begin{pmatrix} & 0.8 \\ 0.8 & \end{pmatrix}$$

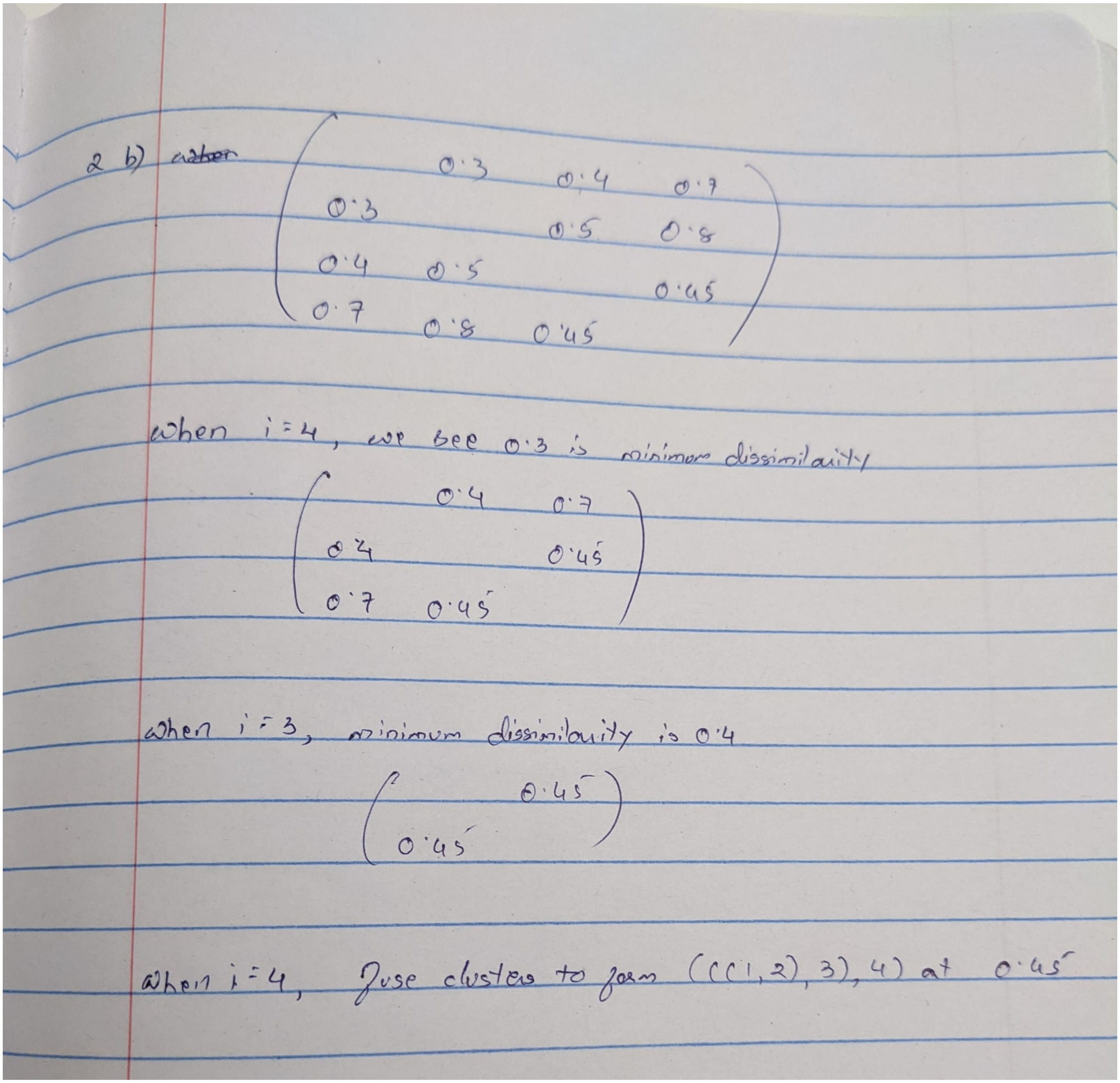
when i = 4 , the cluster becomes ((1,2),(3,4)) at height 0·8

```
dend = 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))
plot(hclust(dend, method = "complete"))
```



b.

```
knitr::include_graphics("2b.jpeg")
```



2 b) ~~when~~

$$\begin{pmatrix} & 0.3 & 0.4 & 0.7 \\ 0.3 & & 0.5 & 0.8 \\ 0.4 & 0.5 & & 0.45 \\ 0.7 & 0.8 & 0.45 & \end{pmatrix}$$

when i=4, we see 0·3 is minimum dissimilarity

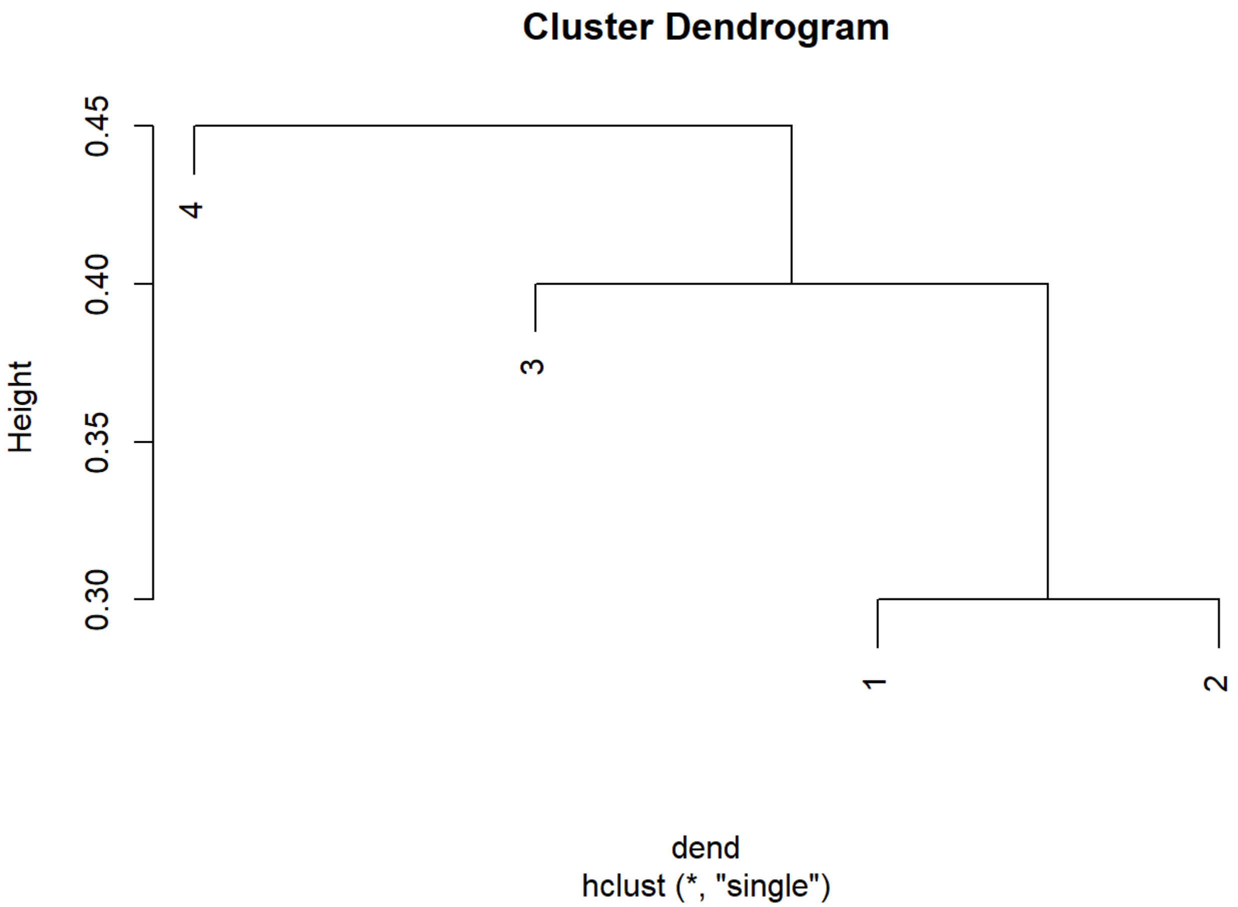
$$\begin{pmatrix} & 0.4 & 0.7 \\ 0.4 & & 0.45 \\ 0.7 & 0.45 & \end{pmatrix}$$

when i=3, minimum dissimilarity is 0·4

$$\begin{pmatrix} & 0.45 \\ 0.45 & \end{pmatrix}$$

when i=4, fuse clusters to form (((1,2),3),4) at 0·45

```
plot(hclust(dend, method = "single"))
```

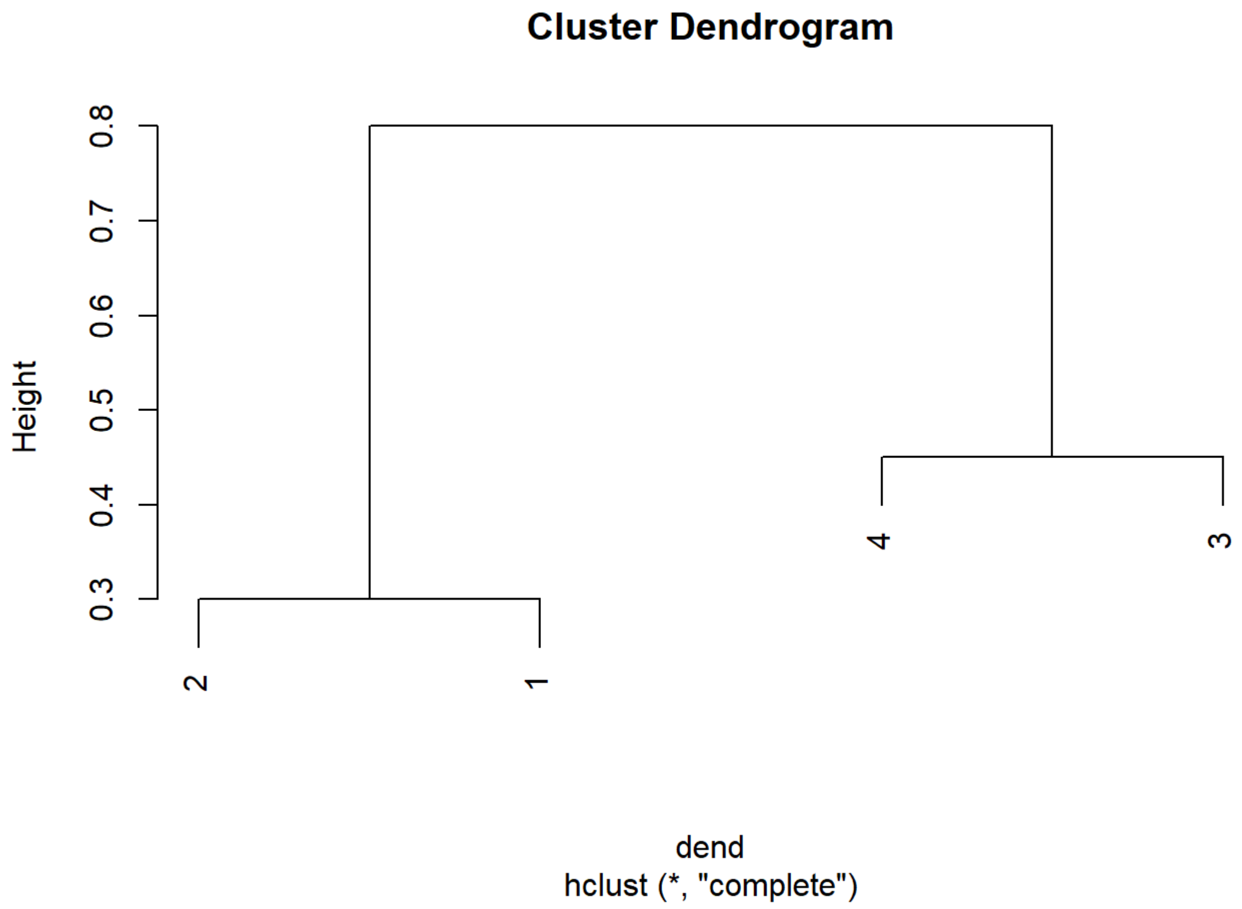


c. In this case, we have clusters (1,2) and (3,4).

d. In this case, we have clusters ((1,2),3) and (4).

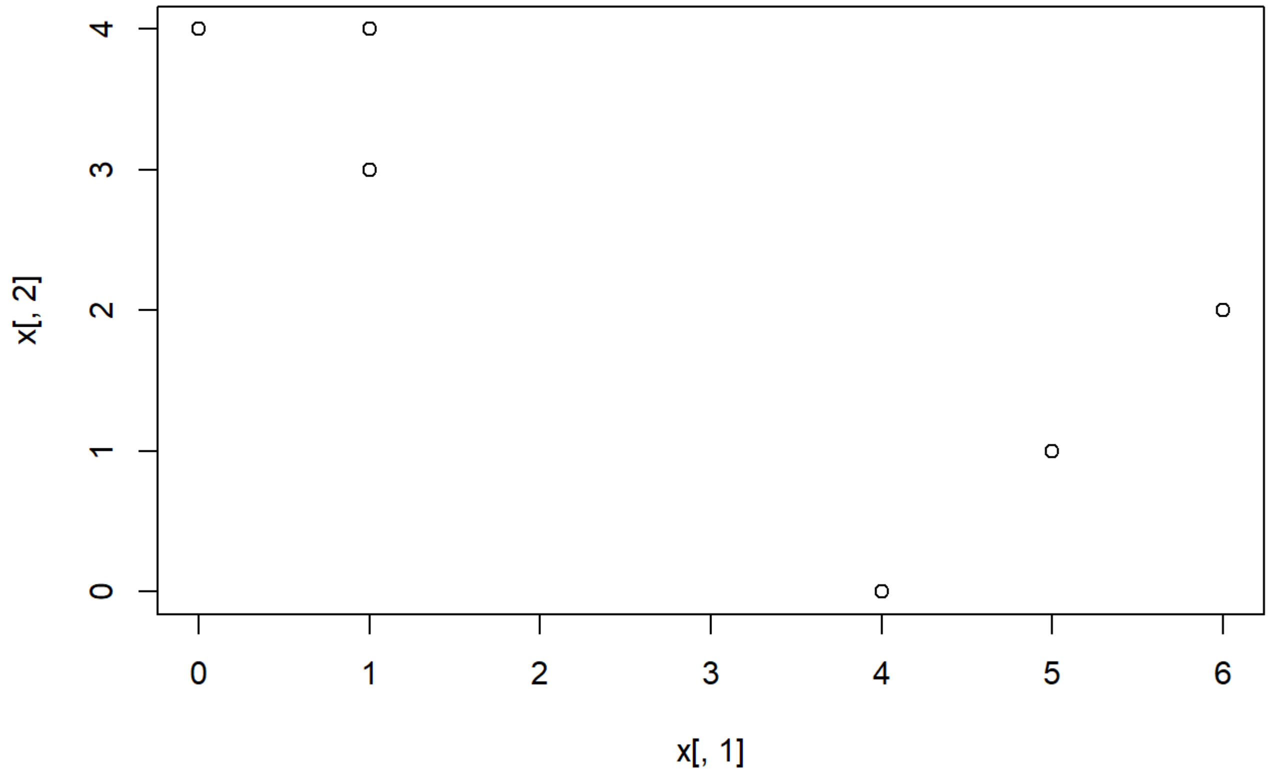
e.

```
plot(hclust(dend, method = "complete"), labels = c(2,1,4,3))
```



Question 3 a)

```
x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
plot(x[,1], x[,2])
```

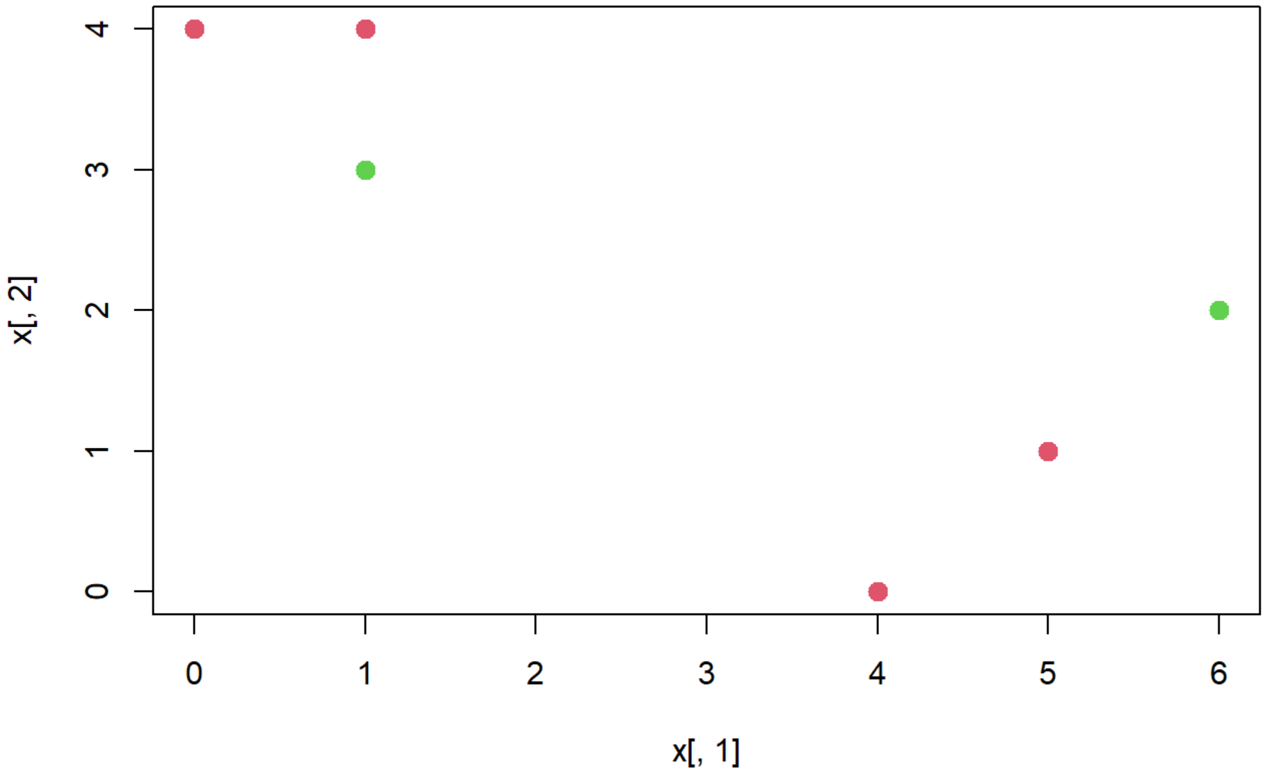


b.

```
set.seed(1)
labels <- sample(2, nrow(x), replace = T)
labels
```

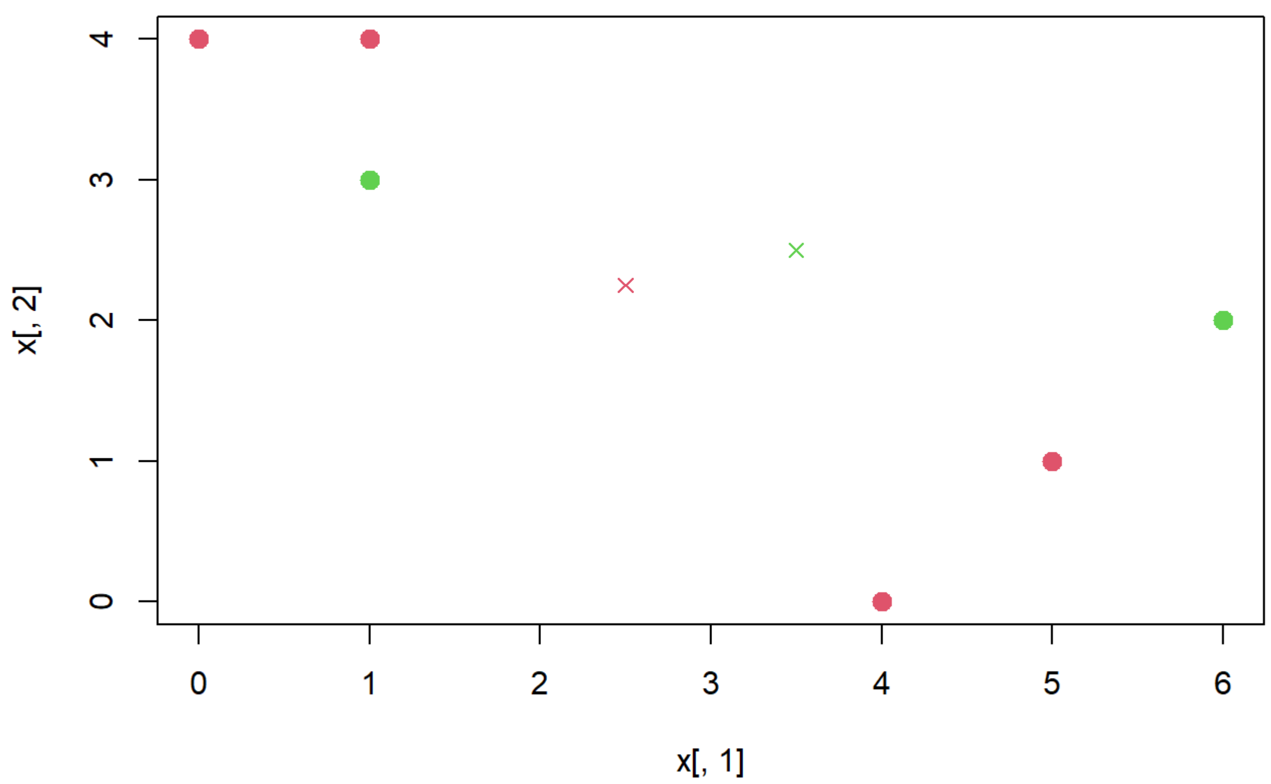
```
## [1] 1 2 1 1 2 1
```

```
plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)
```



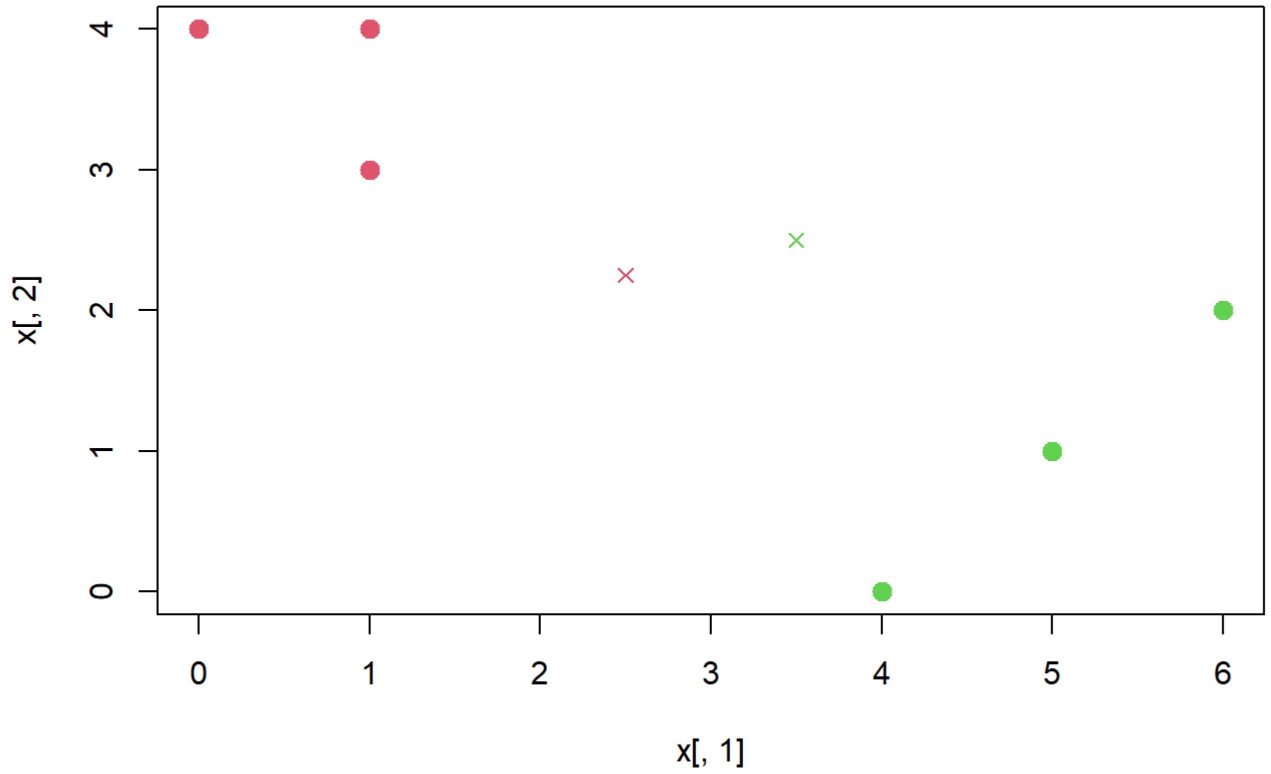
c.

```
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)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



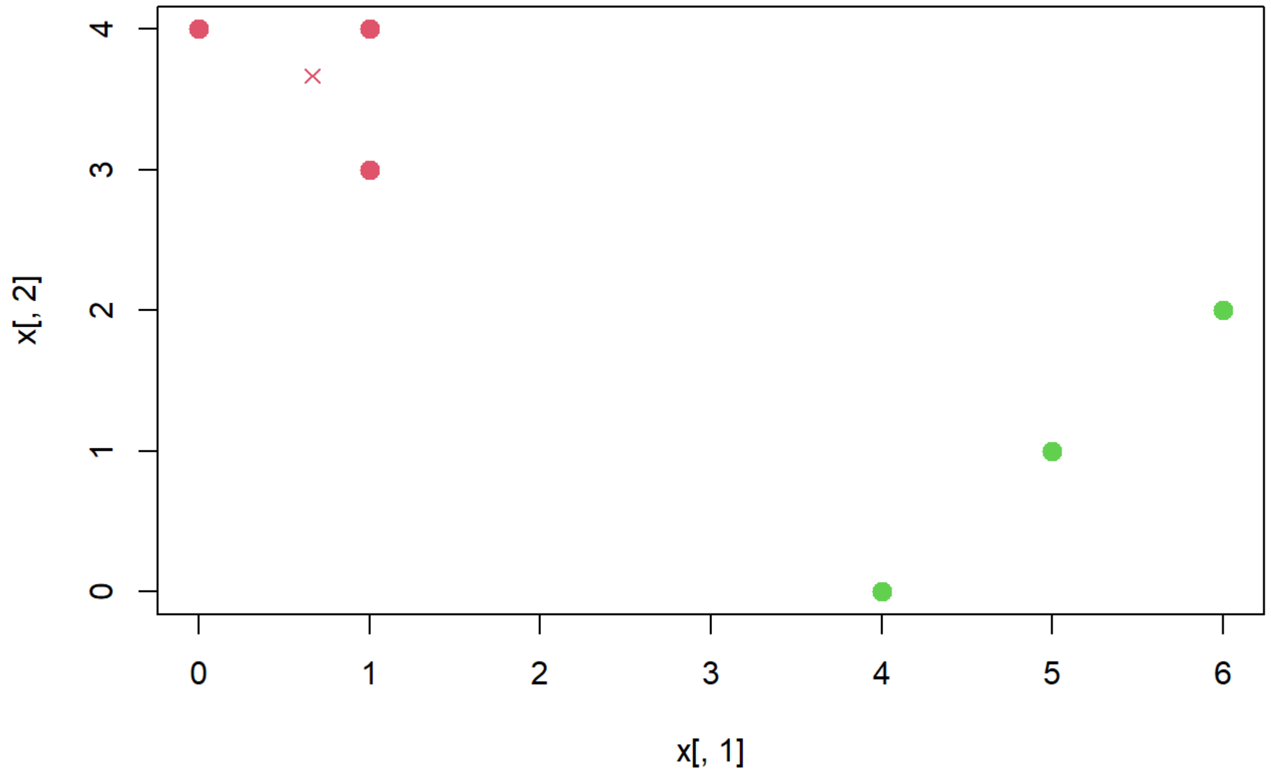
d.

```
labels <- c(1, 1, 1, 2, 2, 2)
plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



e.

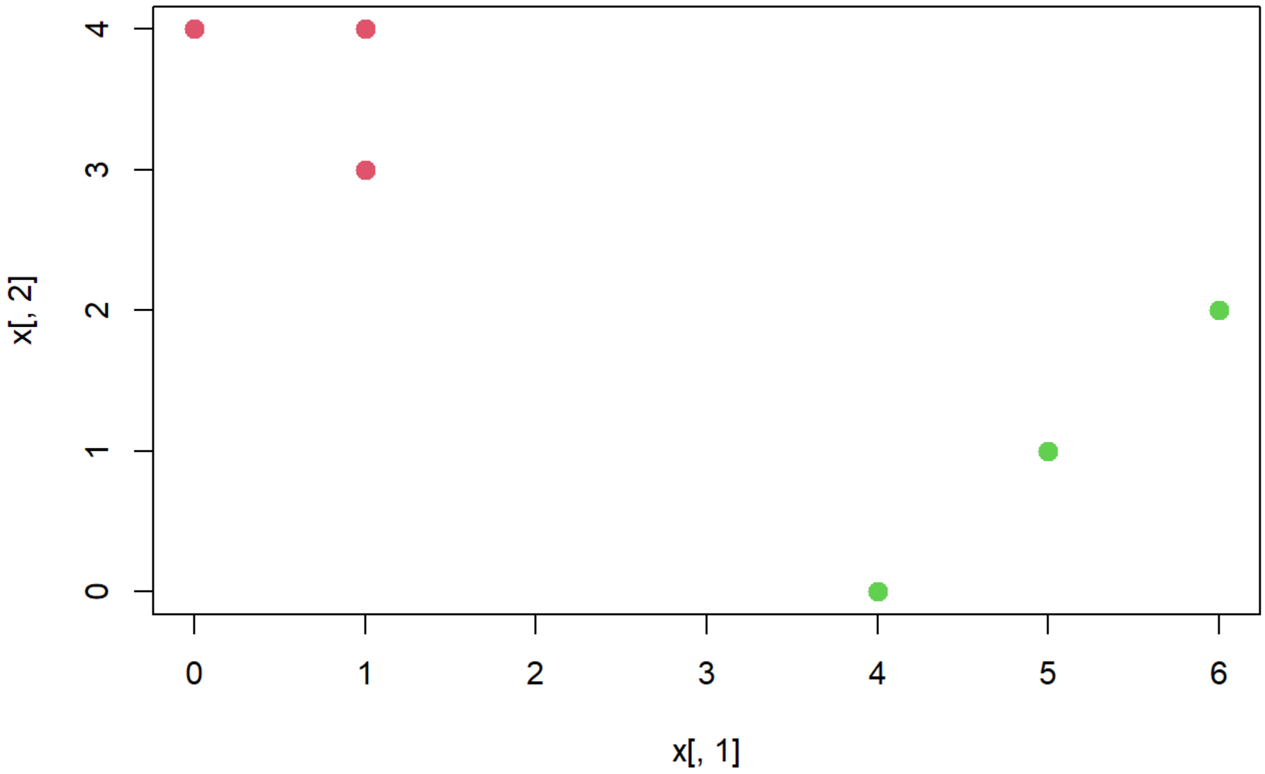
```
centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



If we assign each observation to the centroid to which it is closest, nothing changes, so the algorithm is terminated at this step.

f.

```
plot(x[, 1], x[, 2], col=(labels + 1), pch = 20, cex = 2)
```



Question 4

a. There is not enough information to tell. For example, if d(1,4)=2, d(1,5)=3, d(2,4)=1, d(2,5)=3, d(3,4)=4 and d(3,5)=1, the single linkage dissimilarity between {1,2,3} and {4,5} would be equal to 1 and the complete linkage dissimilarity between {1,2,3} and {4,5} would be equal to 4. So, with single linkage, they would fuse at a height of 1, and with complete linkage, they would fuse at a height of 4. But, if all inter-observations distance are equal to 2, we would have that the single and complete linkage dissimilarities between {1,2,3} and {4,5} are equal to 2.
b. They would fuse at the same height. For example, if d(5,6)=2, the single and complete linkage dissimilarities between {5} and {6} would be equal to 2. So, they would fuse at a height of 2 for single and complete linkage.

Practicum Problems

Problem 1

```
URL <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
data <- read.table(URL,sep=",")
colnames(data) <- c("class","alcohol","malic_acid","ash","alcalinity","magnesium","total_phenol
s","flavanoids",
                     "nonfalvanoid","roanthocyanins","color_intensity","hue","OD280/OD315","proli
ne")
#display top six rows
head(data)
```

```
##   class alcohol malic_acid  ash alcalinity magnesium total_phenols flavanoids
## 1     1   14.23       1.71 2.43       15.6       127          2.80       3.06
## 2     1   13.20       1.78 2.14       11.2       100          2.65       2.76
## 3     1   13.16       2.36 2.67       18.6       101          2.80       3.24
## 4     1   14.37       1.95 2.50       16.8       113          3.85       3.49
## 5     1   13.24       2.59 2.87       21.0       118          2.80       2.69
## 6     1   14.20       1.76 2.45       15.2       112          3.27       3.39
##   nonfalvanoid roanthocyanins color_intensity  hue OD280/OD315 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
## 6         0.34           1.97            6.75 1.05        2.85    1450
```

```
print("Mean")
```

```
## [1] "Mean"
```

```
#check the means of predictors
apply(data[,-1],2,mean)
```

```
##         alcohol      malic_acid             ash      alcalinity       magnesium
##      13.0006180       2.3363483       2.3665169      19.4949438      99.7415730
##   total_phenols      flavanoids    nonfalvanoid roanthocyanins color_intensity
##       2.2951124       2.0292697       0.3618539       1.5908989       5.0580899
##             hue     OD280/OD315         proline
##       0.9574494       2.6116854     746.8932584
```

```
print("-----------------------------------------------------------------------------------------
---------------------")
```

```
## [1] "----------------------------------------------------------------------------------------
----------------------"
```

```
print("Varaince")
```

```
## [1] "Varaince"
```

```
#check the variance of the predictors
apply(data[,-1],2,var)
```

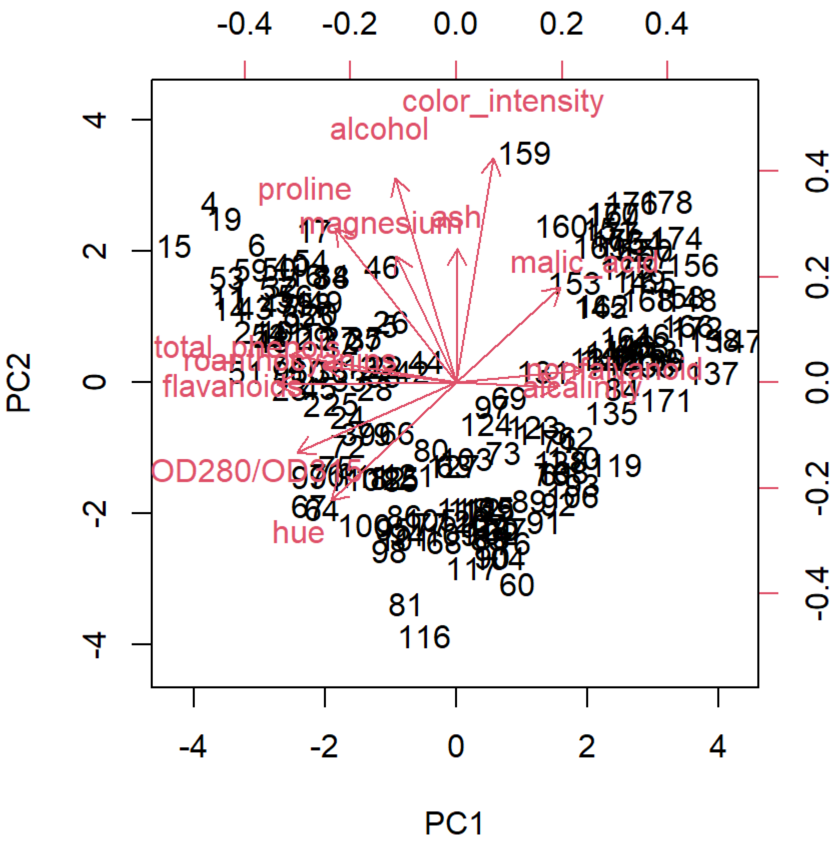
```
##         alcohol      malic_acid             ash      alcalinity       magnesium 
##    6.590623e-01    1.248015e+00    7.526464e-02    1.115269e+01    2.039893e+02 
##   total_phenols      flavanoids    nonfalvanoid  roanthocyanins color_intensity 
##    3.916895e-01    9.977187e-01    1.548863e-02    3.275947e-01    5.374449e+00 
##             hue     OD280/OD315         proline 
##    5.224496e-02    5.040864e-01    9.916672e+04
```

From the above mean and variance values it is clear that values are on different scale. So, we need to perform scaling before applying PCA to our dataset.

```
#using prcomp to perform PCA
output <- prcomp(data[,-1],scale=TRUE)
output$rotation
```

```
##                          PC1          PC2         PC3         PC4         PC5
## alcohol         -0.144329395  0.483651548 -0.20738262  0.01785630 -0.26566365
## malic_acid       0.245187580  0.224930935  0.08901289 -0.53689028  0.03521363
## ash              0.002051061  0.316068814  0.62622390  0.21417556 -0.14302547
## alcalinity       0.239320405 -0.010590502  0.61208035 -0.06085941  0.06610294
## magnesium       -0.141992042  0.299634003  0.13075693  0.35179658  0.72704851
## total_phenols   -0.394660845  0.065039512  0.14617896 -0.19806835 -0.14931841
## flavanoids      -0.422934297 -0.003359812  0.15068190 -0.15229479 -0.10902584
## nonfalvanoid     0.298533103  0.028779488  0.17036816  0.20330102 -0.50070298
## roanthocyanins  -0.313429488  0.039301722  0.14945431 -0.39905653  0.13685982
## color_intensity  0.088616705  0.529995672 -0.13730621 -0.06592568 -0.07643678
## hue             -0.296714564 -0.279235148  0.08522192  0.42777141 -0.17361452
## OD280/OD315     -0.376167411 -0.164496193  0.16600459 -0.18412074 -0.10116099
## proline         -0.286752227  0.364902832 -0.12674592  0.23207086 -0.15786880
##                         PC6         PC7         PC8         PC9        PC10
## alcohol          0.21353865 -0.05639636  0.39613926 -0.50861912  0.21160473
## malic_acid       0.53681385  0.42052391  0.06582674  0.07528304 -0.30907994
## ash              0.15447466 -0.14917061 -0.17026002  0.30769445 -0.02712539
## alcalinity      -0.10082451 -0.28696914  0.42797018 -0.20044931  0.05279942
## magnesium        0.03814394  0.32288330 -0.15636143 -0.27140257  0.06787022
## total_phenols   -0.08412230 -0.02792498 -0.40593409 -0.28603452 -0.32013135
## flavanoids      -0.01892002 -0.06068521 -0.18724536 -0.04957849 -0.16315051
## nonfalvanoid    -0.25859401  0.59544729 -0.23328465 -0.19550132  0.21553507
## roanthocyanins  -0.53379539  0.37213935  0.36822675  0.20914487  0.13418390
## color_intensity -0.41864414 -0.22771214 -0.03379692 -0.05621752 -0.29077518
## hue              0.10598274  0.23207564  0.43662362 -0.08582839 -0.52239889
## OD280/OD315      0.26585107 -0.04476370 -0.07810789 -0.13722690  0.52370587
## proline          0.11972557  0.07680450  0.12002267  0.57578611  0.16211600
##                        PC11        PC12        PC13
## alcohol          0.22591696 -0.26628645  0.01496997
## malic_acid      -0.07648554  0.12169604  0.02596375
## ash              0.49869142 -0.04962237 -0.14121803
## alcalinity      -0.47931378 -0.05574287  0.09168285
## magnesium       -0.07128891  0.06222011  0.05677422
## total_phenols   -0.30434119 -0.30388245 -0.46390791
## flavanoids       0.02569409 -0.04289883  0.83225706
## nonfalvanoid    -0.11689586  0.04235219  0.11403985
## roanthocyanins   0.23736257 -0.09555303 -0.11691707
## color_intensity -0.03183880  0.60422163 -0.01199280
## hue              0.04821201  0.25921400 -0.08988884
## OD280/OD315     -0.04642330  0.60095872 -0.15671813
## proline         -0.53926983 -0.07940162  0.01444734
```

```
#biplot
biplot(output,scale=0)
```

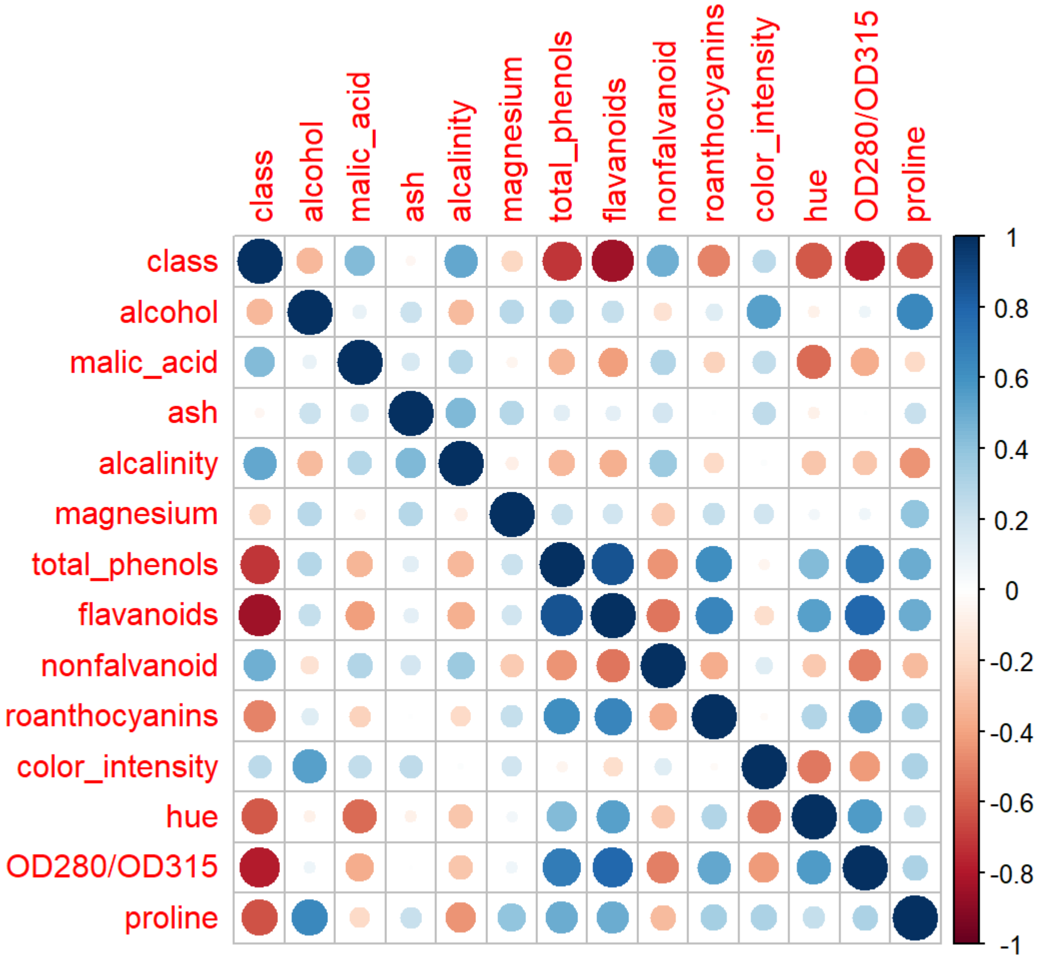


From the above plot we can see that feature malic_acid is pointed in opposite direction to the feature hue.

```
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```
M <- cor(data)
corrplot(M)
```

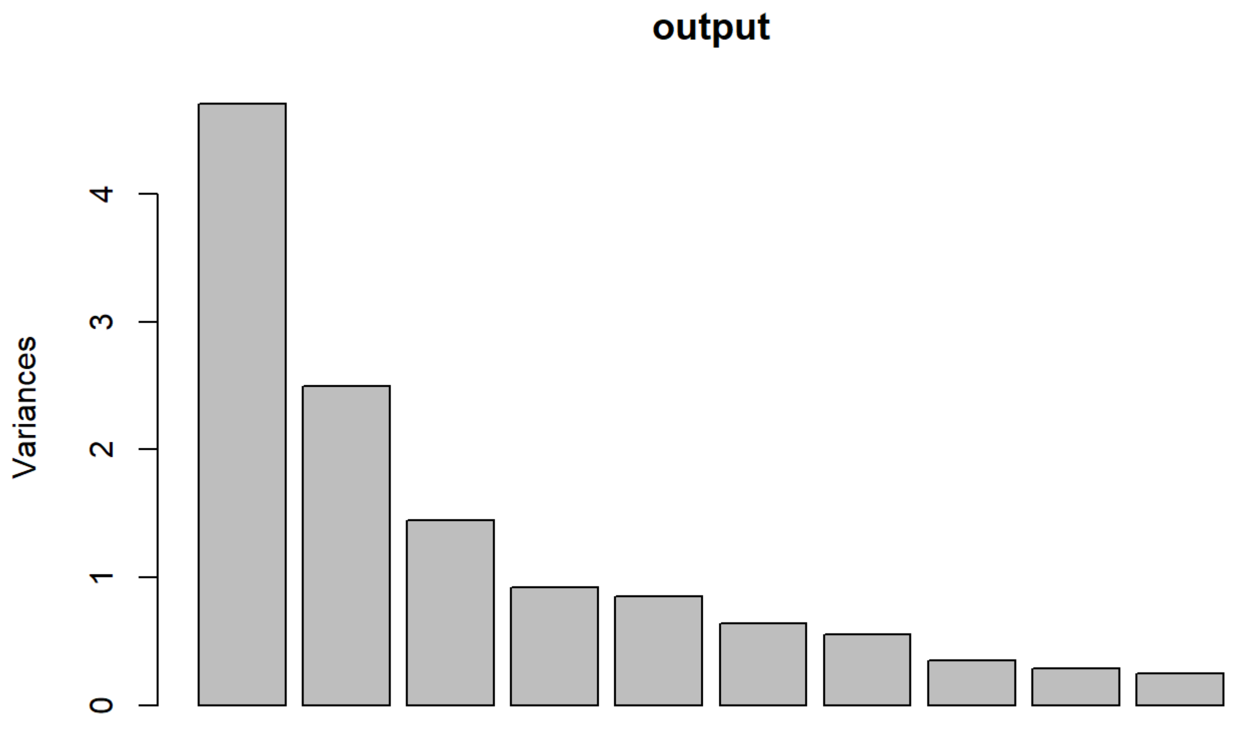


```
cor(data$malic_acid,data$hue)
```

```
## [1] -0.5612957
```

From the correlation value between feature hue and malic acid it is clear that as the one variable increases the other variable decreases with the almost same extent.

```
screeplot(output)
```

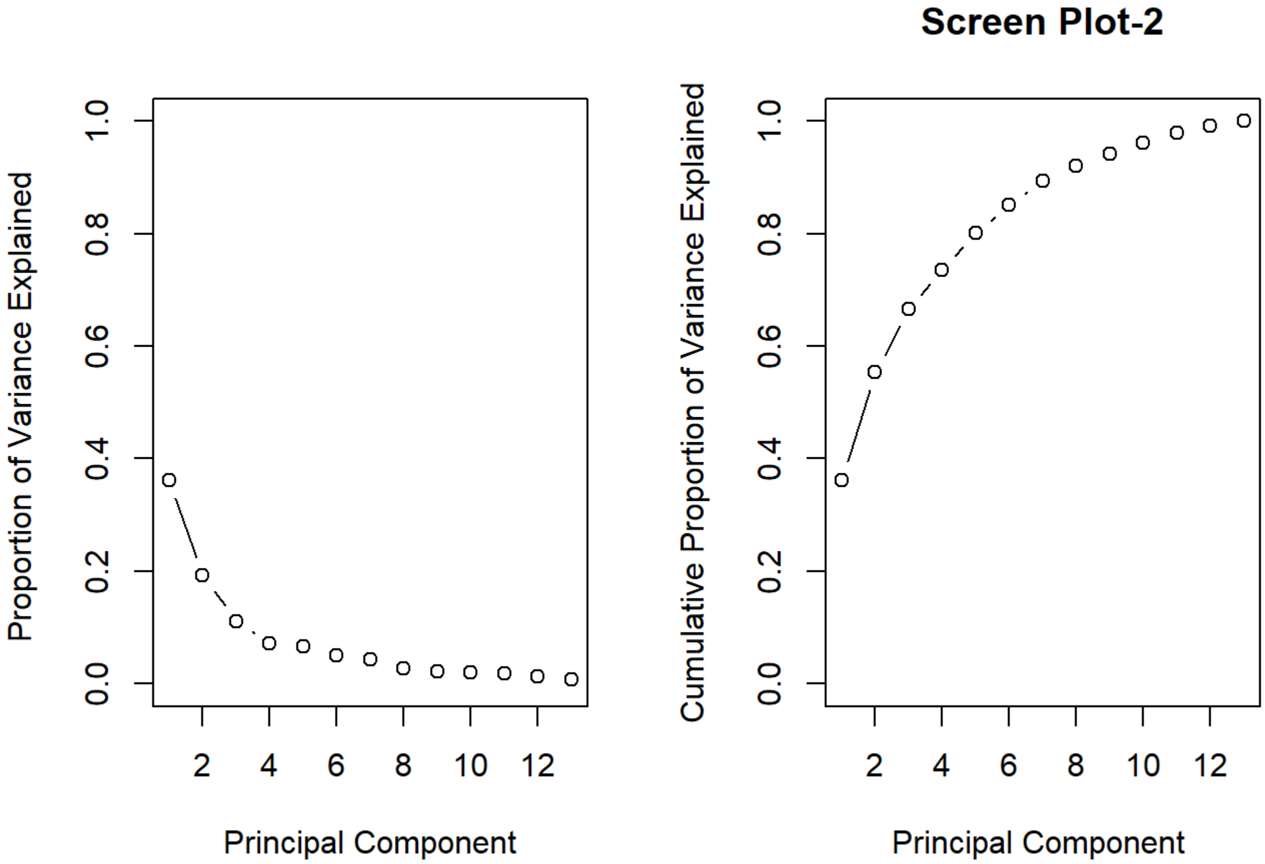


```
summary(output)
```

```
## 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
```

```
#calculating proportion of variance for each principle component
variance <- output$sdev^2
pve <- variance/sum(variance)
```

```
#screenplot
par(mfrow=c(1,2))
plot(pve, xlab="Principal Component", ylab="Proportion of Variance Explained ",ylim=c(0,1),type=
'b')
plot(cumsum(pve), xlab="Principal Component ", ylab=" Cumulative Proportion of Variance Explaine
d ",main="Screen Plot-2", ylim=c(0,1), type='b')
```



```
#Proportion of variance expalined by PC1 and PC2
temp<-pve[1:2]*100
temp
```

```
## [1] 36.19885 19.20749
```

```
sum(temp)
```

```
## [1] 55.40634
```

Thus, from the above results it is clear that PC1 and PC2 has explained total of 55.40% of variance.

Problem 2

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

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

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#Load the dataset
data("USArrests")

#convert the dataset to a dataframe
data <- data.frame(USArrests)
```

```
print("Mean")
```

```
## [1] "Mean"
```

```
#checking the mean of the predictors
apply(data,2,mean)
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
print("-----------------------------------------------------------------------------------------
--------------")
```

```
## [1] "----------------------------------------------------------------------------------------
---------------"
```

```
print("Varaince")
```

```
## [1] "Varaince"
```

```
#checking the variance of the predictors
apply(data,2,var)
```

```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

In the above mean and variance values it is clear that values are on different scale. So, we need to perform scaling before applying k-means to our dataset.

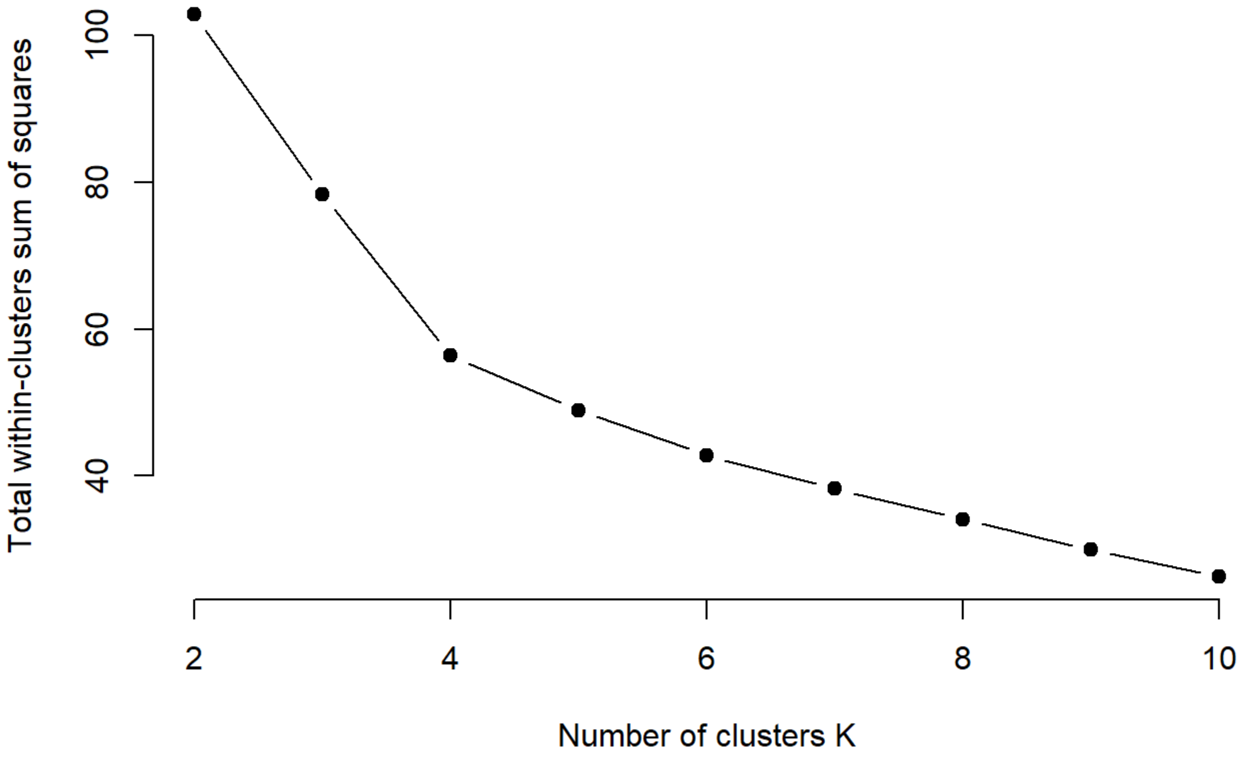
```
#scaling the dataset
n_data <- scale(data, center = TRUE, scale = TRUE)
```

```
#Applying K-Means
result <- function(k)
{
  kmeans(n_data,centers=k,nstart=20)$tot.withinss
}
# values of k form 2 to 10
k <- 2:10
```

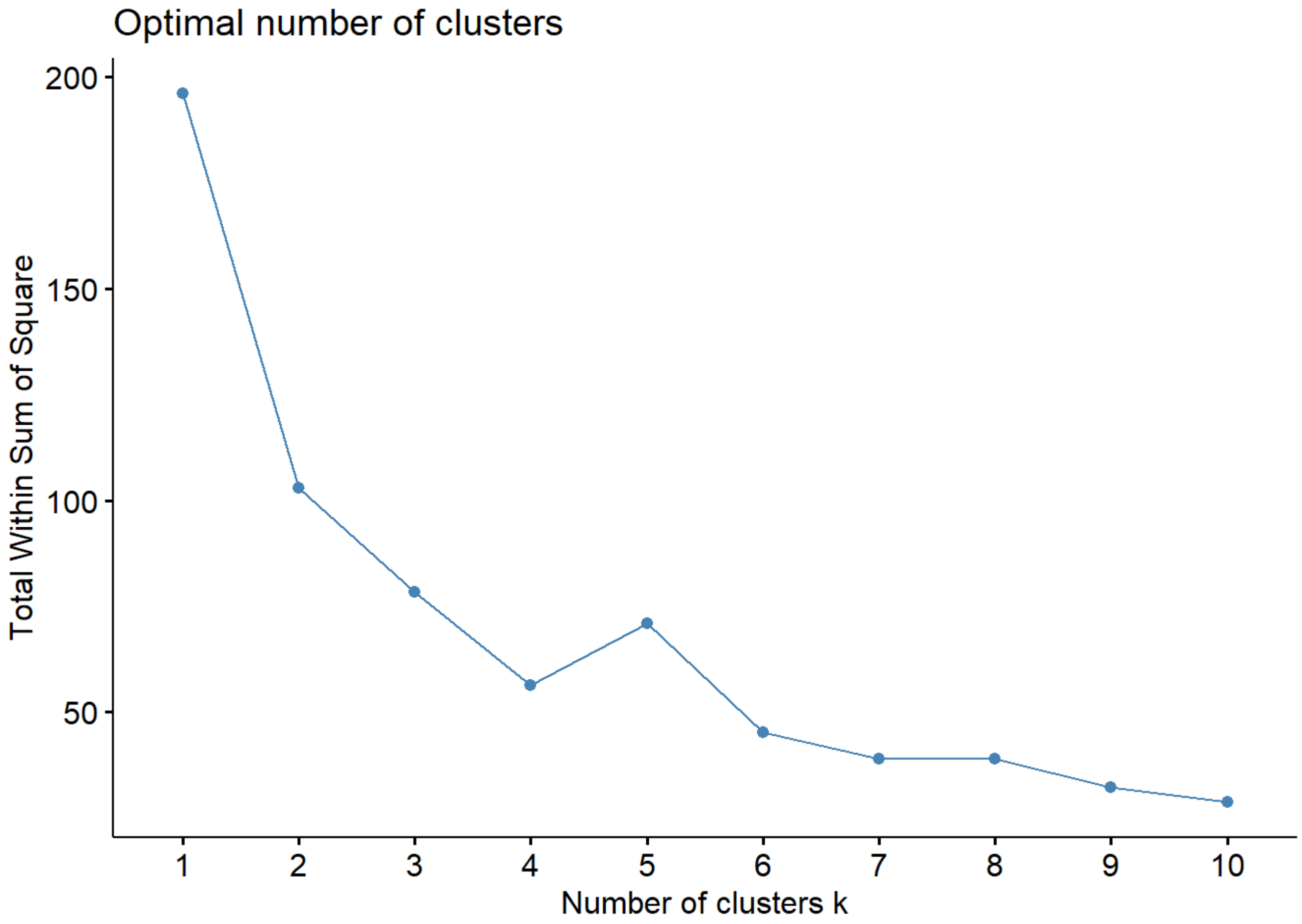
```
#compute total within-cluster sum of square values of k from 2 to 10
wss_val <- map_dbl(k, result)
wss_val
```

```
## [1] 102.86240  78.32327  56.40317  48.94420  42.83303  38.25764  34.10865
## [8]  29.94611  26.26171
```

```
#Using elbow method to find optimal K value
plot(k, wss_val,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```

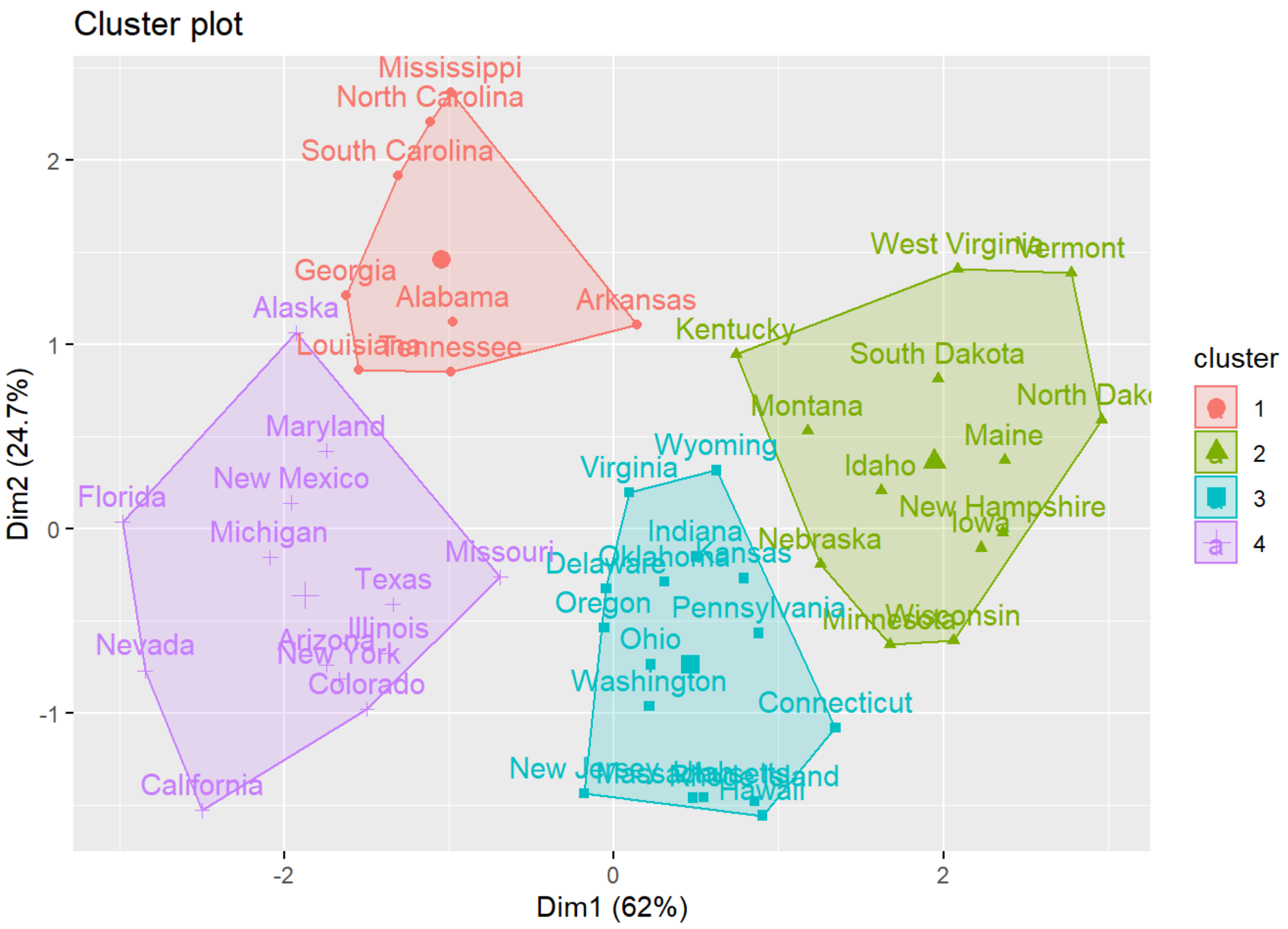


```
#another Method
fviz_nbclust(n_data, kmeans, method = "wss")
```



From the above two graph it is clear that if we consider major drop in total within-clusters sum of square values then the optimal value of k in this case will be 4.

```
#plots
optimal <- kmeans(n_data, centers = 4, nstart = 20)
fviz_cluster(optimal, data = n_data)
```



Problem 3

```
URL <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-whit
e.csv"
wine <- read.csv(URL,sep=";")


#display dataset
head(wine)
```

```
##   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
##   quality
## 1       6
## 2       6
## 3       6
## 4       6
## 5       6
## 6       6
```

```
#exclude quality variable
dataset <- wine[,-12]
```

```
print("Mean")
```

```
## [1] "Mean"
```

```
#check the mean
apply(dataset,2,mean)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##           6.85478767           0.27824112           0.33419151
##       residual.sugar            chlorides  free.sulfur.dioxide
##           6.39141486           0.04577236          35.30808493
## total.sulfur.dioxide              density                   pH
##         138.36065741           0.99402738           3.18826664
##            sulphates              alcohol
##           0.48984688          10.51426705
```

```
print("-----------------------------------------------------------------------------------------
-------------")
```

```
## [1] "----------------------------------------------------------------------------------------
--------------"
```

```
print("Varaince")
```

```
## [1] "Varaince"
```

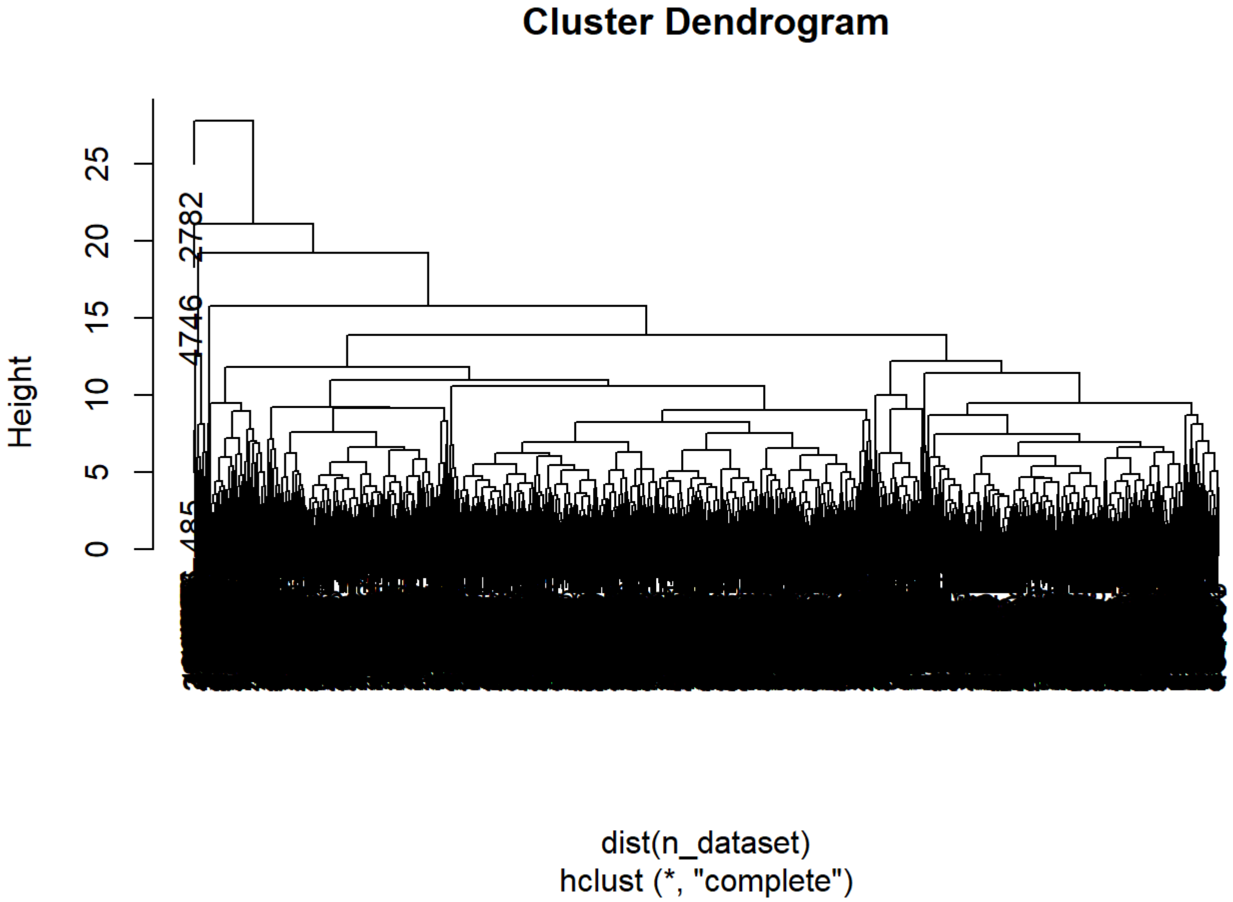
```
#check the variance
apply(dataset,2,var)
```

```
##        fixed.acidity     volatile.acidity          citric.acid 
##         7.121136e-01         1.015954e-02         1.464579e-02 
##       residual.sugar            chlorides  free.sulfur.dioxide 
##         2.572577e+01         4.773337e-04         2.892427e+02 
## total.sulfur.dioxide              density                   pH 
##         1.806085e+03         8.945524e-06         2.280118e-02 
##            sulphates              alcohol 
##         1.302471e-02         1.514427e+00
```

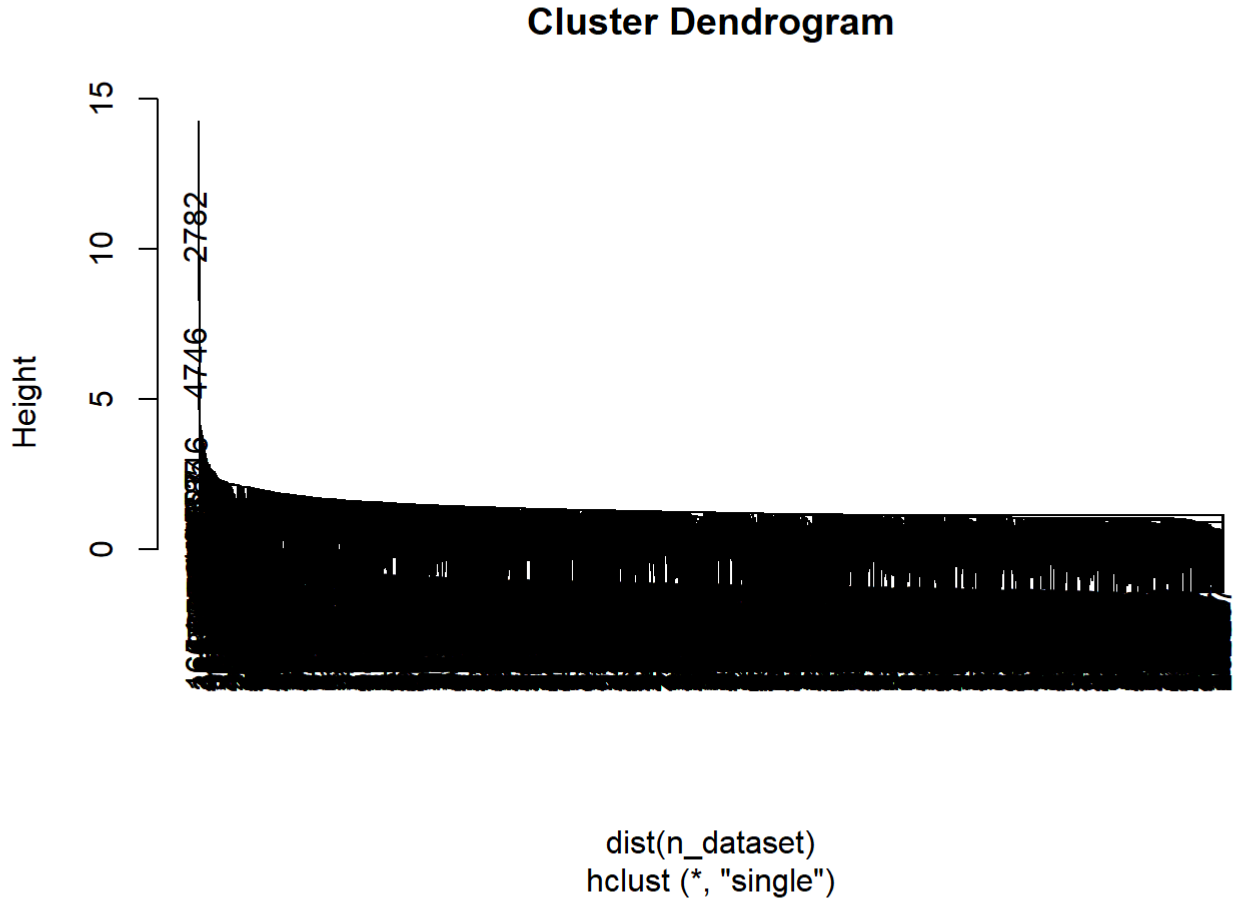
In the above mean and variance values it is clear that values are on different scale. So, we need to perform scaling before applying hclust to our dataset.

```
#apply scaling
n_dataset <- scale(dataset,center = TRUE,scale=TRUE)
```

```
#Performing hierarchical clustering using complete linkage
hc.complete <- hclust(dist(n_dataset),method="complete")
#dendogram of complete linkage
plot(hc.complete)
```



```
#Performing hierarchical clustering using single linkage
hc.single <- hclust(dist(n_dataset),method="single")
#dendogram of single linkage
plot(hc.single)
```



```
#for complete linkage
tail(hc.complete$height,1)
```

```
## [1] 27.73476
```

For complete linkage two penultimate clusters will merge at 27.73476

```
#for single linkage
tail(hc.single$height,1)
```

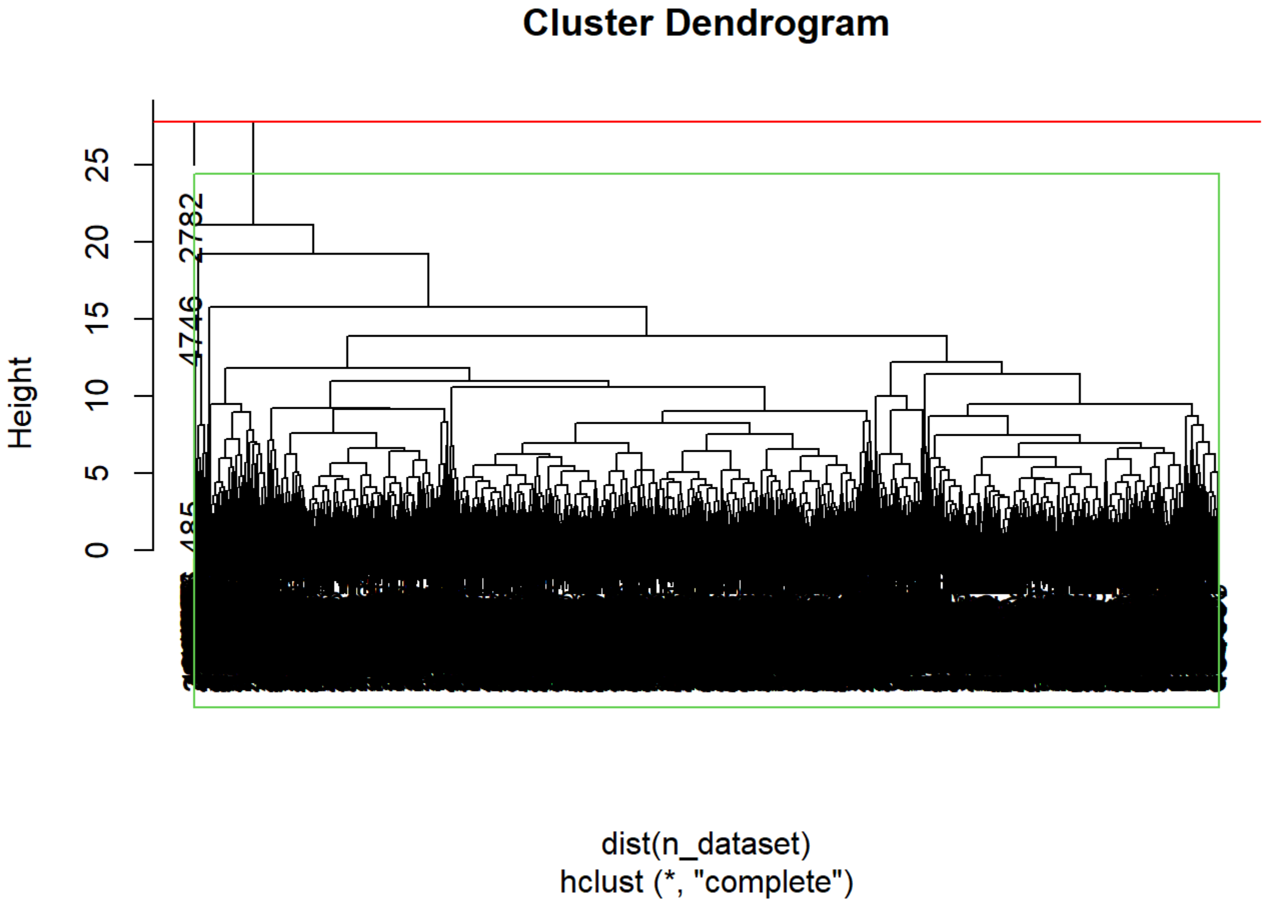
```
## [1] 14.25323
```

For single linkage two penultimate clusters will merge at 14.25325

```
#applying cutree method on complete linkage
cut.complete <- cutree(hc.complete,h=27.73476)
#Number of clusters formed
table(cut.complete)
```

```
## cut.complete
##    1    2
## 4897    1
```

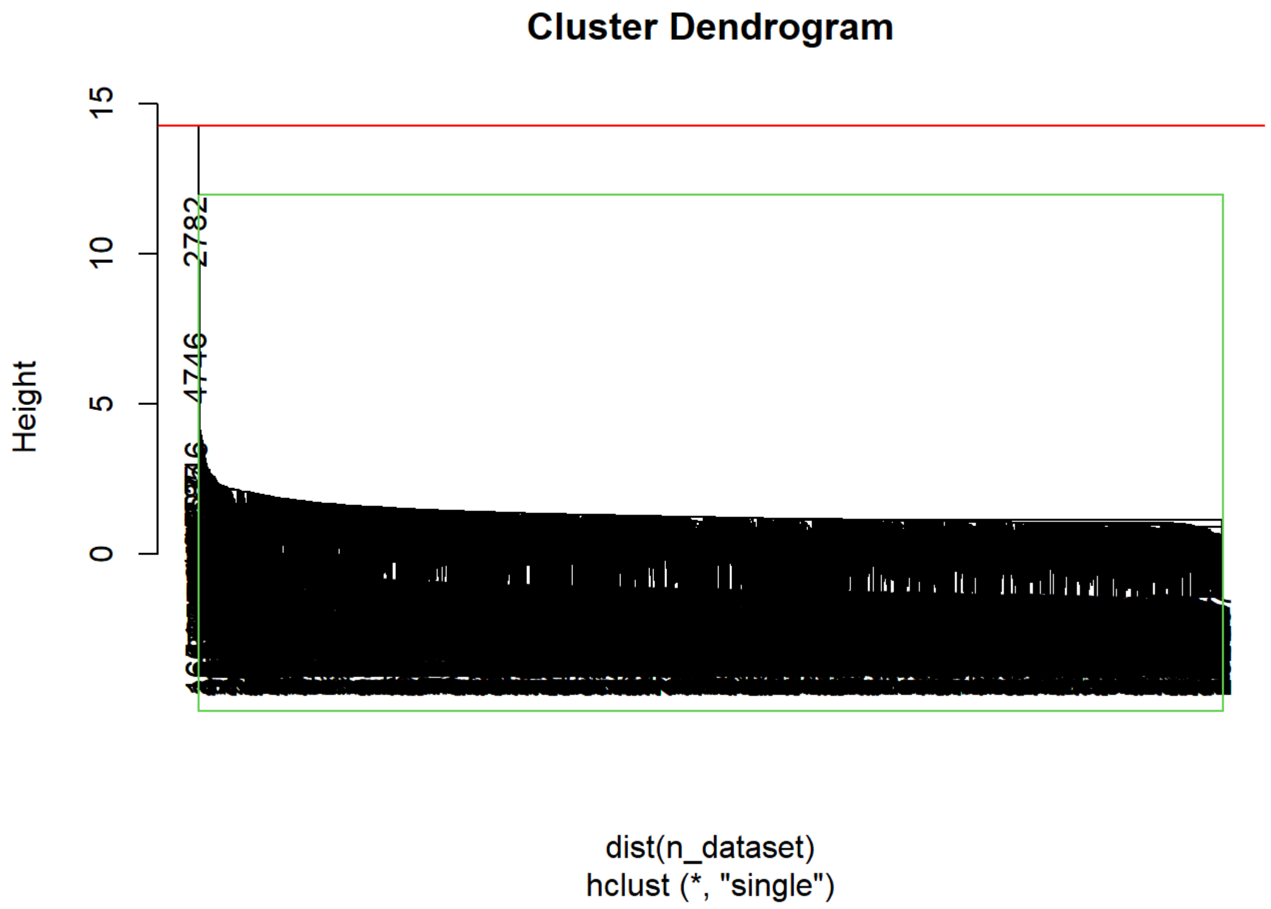
```
plot(hc.complete)
rect.hclust(hc.complete ,h=27.73476, border = 2:6)
abline(h =27.73476, col = 'red')
```



```
#applying cutree method on single linkage
cut.single <- cutree(hc.single,h=14.25323)
#Number of clusters formed
table(cut.single)
```

```
## cut.single
##    1    2
## 4897    1
```

```
plot(hc.single)
rect.hclust(hc.single ,h=14.25323, border = 2:6)
abline(h =14.25323, col = 'red')
```



```
#summary Statistics for complete linkage
dataset$Clusters <- cut.complete
unique(dataset$Clusters)
```

```
## [1] 1 2
```

```
dataset <- dplyr::group_by(dataset,Clusters)
a <- dplyr::summarise_each(dataset, funs(mean))
```

```
## Warning: `summarise_each_()` was deprecated in dplyr 0.7.0.
## Please use `across()` instead.
```

```
## Warning: `funs()` was deprecated in 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))
```

```
#Difference in feature means for complete linkage
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
```

```
#summary Statistics for single linkage
dataset$Clusters <- cut.single
unique(dataset$Clusters)
```

```
## [1] 1 2
```

```
dataset <- dplyr::group_by(dataset,Clusters)
b <- dplyr::summarise_each(dataset, funs(mean))
```

```
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
```

From the above results we can see that feature residual.sugar has maximum means difference. Also, from the above two plots of Complete and Single linkage we can conclude that Complete linkage produces more balanced clustering.