**HOMEWORK #4**

**Submitted by:**

Sreekhar Ale, M.Eng CS,   
Course: Intelligent Data Analysis – fall 2015  
Language used: R

**ANSWER 1 (a):**

*List of steps*

1. Enter the data as mentioned in the question
2. Enter the conditions as described in the question i.e., Number of clusters and threshold theta
3. Create a function to calculate the minimum distance along with its center values
4. Calculate the clusters value (data frame) with centers and also with its original data points depending on the threshold values and the number of clusters
5. Create the clustered plot using the above obtained clustered values and the original data using the clusplot function in cluster library
6. Find below the source code

**R SOURCE CODE:**

**#ENTERING THE GIVEN DATA AS MENTIONED IN THE QUESTION**

XValues <- c(6, 19, 15, 11, 18, 9, 19, 18, 5, 4, 7, 21, 1, 1, 0, 5)

YValues <- c(12, 7, 4, 0, 12, 20, 22, 17, 11, 18, 15, 18, 19, 4, 9, 11)

TwoDdata <- data.frame(XValues, YValues)

Max\_NumberOfClusters <- 4

Threshold\_Theta <- 12

**#CREATE A FUNCTION TO CALCULATE THE MINIMUM DISTANCE**

MinDistance\_WithCenter <- function(datapoints,centers,tempcluster)

{

**#DECLARING THE VARIABLE WITH LENGTH 4 AS THE MAX NUMBER OF CLUSTERS IS 4**

**#FIRST ELEMENT INITIALIZED TO 0 AND REMAINING ELEMENTS ARE OF HIGHER VALUES**

distance <- c(0,250,250,250)

for(i in 1:tempcluster)

{

distance[i] <- sqrt((datapoints[1,1]-centers[i,1])^2+(datapoints[1,2]-centers[i,2])^2)

}

**#CALCULATING MINIMUM DISTANCE**

MinDistance <- min(distance,na.rm = TRUE);

Center\_Element <- which(distance == MinDistance)

Center\_Element\_Value <- centers[Center\_Element,]

MinDistance\_WithCenter\_Value <- c(MinDistance,as.matrix(Center\_Element\_Value))

return(MinDistance\_WithCenter\_Value)

}

**#DECLARE THE VARIABLE FOR STORING CLUSTERS ALONG WITH THEIR CENTERS**

Clusters\_with\_center <- data.frame(matrix(NA, nrow = 16, ncol = 5))

**#START FROM THE FIRST DATA POINT TO THE LAST**

**#AS THE FIRST DATA POINT IS (6,12)**

Clusters\_with\_center[1,] <- c(6,12,6,12,1)

Cluster\_Center <- TwoDdata[1,]

centers <- TwoDdata[1,]

tempcluster <- 1

for(i in 2 : nrow(TwoDdata))

{

datapoints <- TwoDdata[i,]

MinDistance\_WithCenter\_Value <- MinDistance\_WithCenter(datapoints,centers,tempcluster)

if(MinDistance\_WithCenter\_Value[1]>Threshold\_Theta && tempcluster < Max\_NumberOfClusters)

{

tempcluster <- tempcluster + 1

centers[tempcluster,] = datapoints

Cluster\_Center[tempcluster]= 1

Clusters\_with\_center[i,1] <- datapoints[1,1]

Clusters\_with\_center[i,2] <- datapoints[1,2]

Clusters\_with\_center[i,3] <- centers[tempcluster,1]

Clusters\_with\_center[i,4] <- centers[tempcluster,2]

Clusters\_with\_center[i,5] <- tempcluster

}

else

{

Clusters\_with\_center[i,1] <- datapoints[1,1]

Clusters\_with\_center[i,2] <- datapoints[1,2]

Clusters\_with\_center[i,3] <- MinDistance\_WithCenter\_Value[2]

Clusters\_with\_center[i,4] <- MinDistance\_WithCenter\_Value[3]

for(j in 1:tempcluster)

{

if(centers[j,1]==MinDistance\_WithCenter\_Value[2] && centers[j,2]== MinDistance\_WithCenter\_Value[3])

{

Clusters\_with\_center[i,5]=j

a1=which(Clusters\_with\_center[,5]==j)

temp\_x=0

temp\_y=0

for(temp\_ce in a1)

{

temp\_x =temp\_x + Clusters\_with\_center[temp\_ce,1]

temp\_y =temp\_y + Clusters\_with\_center[temp\_ce,2]

}

centers[j,1]=temp\_x/length(a1)

centers[j,2]=temp\_y/length(a1)

}

}

}

}

**# CREATE A CLUSTERED PLOT USING CLUSPLOT FUNCTION IN CLUSTER PACKAGE**

library(cluster)

ScaledData <- scale(TwoDdata)

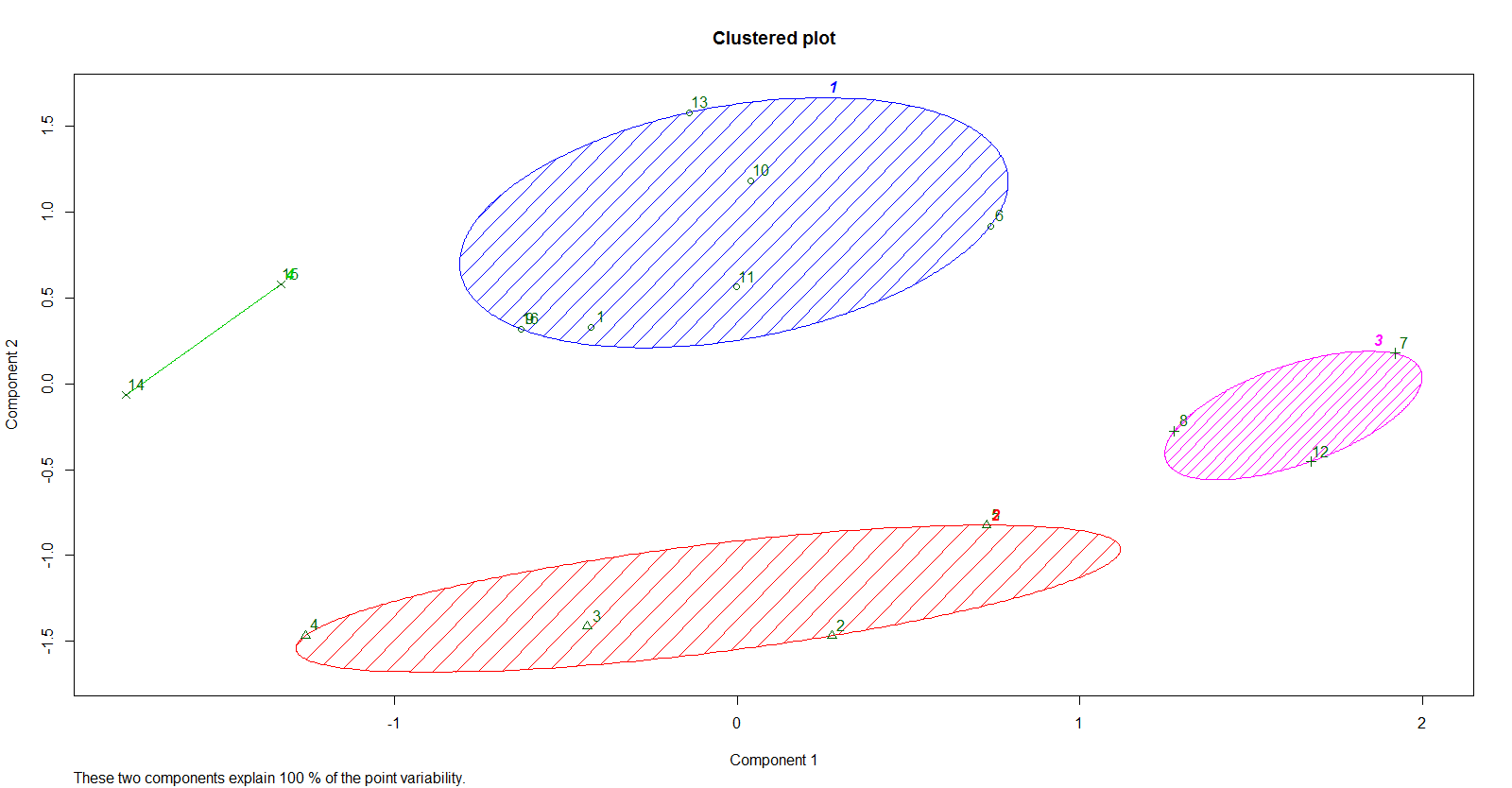
clusplot(ScaledData, Clusters\_with\_center[,5], color=TRUE, shade=TRUE, labels=2, lines = 0,

main = "Clustered plot")

**RESULTS:**

Clusters\_with\_center[,5] = [1] 1 2 2 2 2 1 3 3 1 1 1 3 1 4 4 1

**SCREEN SHOT:**



**ANSWER 1 (b):**

*List of steps*

1. Data points should be reversed as mentioned in the question, hence enter the data in a reversed fashion.
2. Use the function created in the previous question to calculate the minimum distance along with the data points.
3. Conditions remain the same and hence the previous problems conditions can be repeated.
4. Calculate the clusters value (data frame) with centers and also with its original data points depending on the threshold values and the number of clusters
5. Create the clustered plot using the above obtained clustered values and the original data using the clusplot function in cluster library
6. Find below the source code

**R SOURCE CODE:**

**#REVERSING THE DATA AS MENTIONED IN THE QUESTION**

XValues\_Rev <- rev(XValues)

YValues\_Rev <- rev(YValues)

TwoDdata\_Rev = data.frame(XValues\_Rev, YValues\_Rev)

**#DECLARE THE VARIABLE FOR STORING CLUSTERS ALONG WITH THEIR CENTERS**

Clusters\_with\_center\_Rev <- data.frame(matrix(NA, nrow = 16, ncol = 5))

**#START FROM THE FIRST DATA POINT TO THE LAST (INITIAL VALUES ARE REVERSED)**

**#AS THE FIRST DATA POINT IS (5,11)**

Clusters\_with\_center\_Rev[1,] <- c(5,11,5,11,1)

Cluster\_Center <- TwoDdata\_Rev[1,]

centers <- TwoDdata\_Rev[1,]

tempcluster <- 1

for(i in 2 : nrow(TwoDdata\_Rev))

{

datapoints <- TwoDdata\_Rev[i,]

MinDistance\_WithCenter\_Value <- MinDistance\_WithCenter(datapoints,centers,tempcluster)

if(MinDistance\_WithCenter\_Value[1]>Threshold\_Theta && tempcluster < Max\_NumberOfClusters)

{

tempcluster <- tempcluster + 1

centers[tempcluster,] = datapoints

Cluster\_Center[tempcluster]= 1

Clusters\_with\_center\_Rev[i,1] <- datapoints[1,1]

Clusters\_with\_center\_Rev[i,2] <- datapoints[1,2]

Clusters\_with\_center\_Rev[i,3] <- centers[tempcluster,1]

Clusters\_with\_center\_Rev[i,4] <- centers[tempcluster,2]

Clusters\_with\_center\_Rev[i,5] <- tempcluster

}

else

{

Clusters\_with\_center\_Rev[i,1] <- datapoints[1,1]

Clusters\_with\_center\_Rev[i,2] <- datapoints[1,2]

Clusters\_with\_center\_Rev[i,3] <- MinDistance\_WithCenter\_Value[2]

Clusters\_with\_center\_Rev[i,4] <- MinDistance\_WithCenter\_Value[3]

for(k in 1:tempcluster)

{

if(centers[k,1]==MinDistance\_WithCenter\_Value[2] && centers[k,2]==MinDistance\_WithCenter\_Value[3])

Clusters\_with\_center\_Rev[i,5]=k

}

}

}

**# CREATE A CLUSTERED PLOT USING CLUSPLOT FUNCTION IN CLUSTER PACKAGE**

library(cluster)

ScaledData\_Rev <- scale(TwoDdata\_Rev)

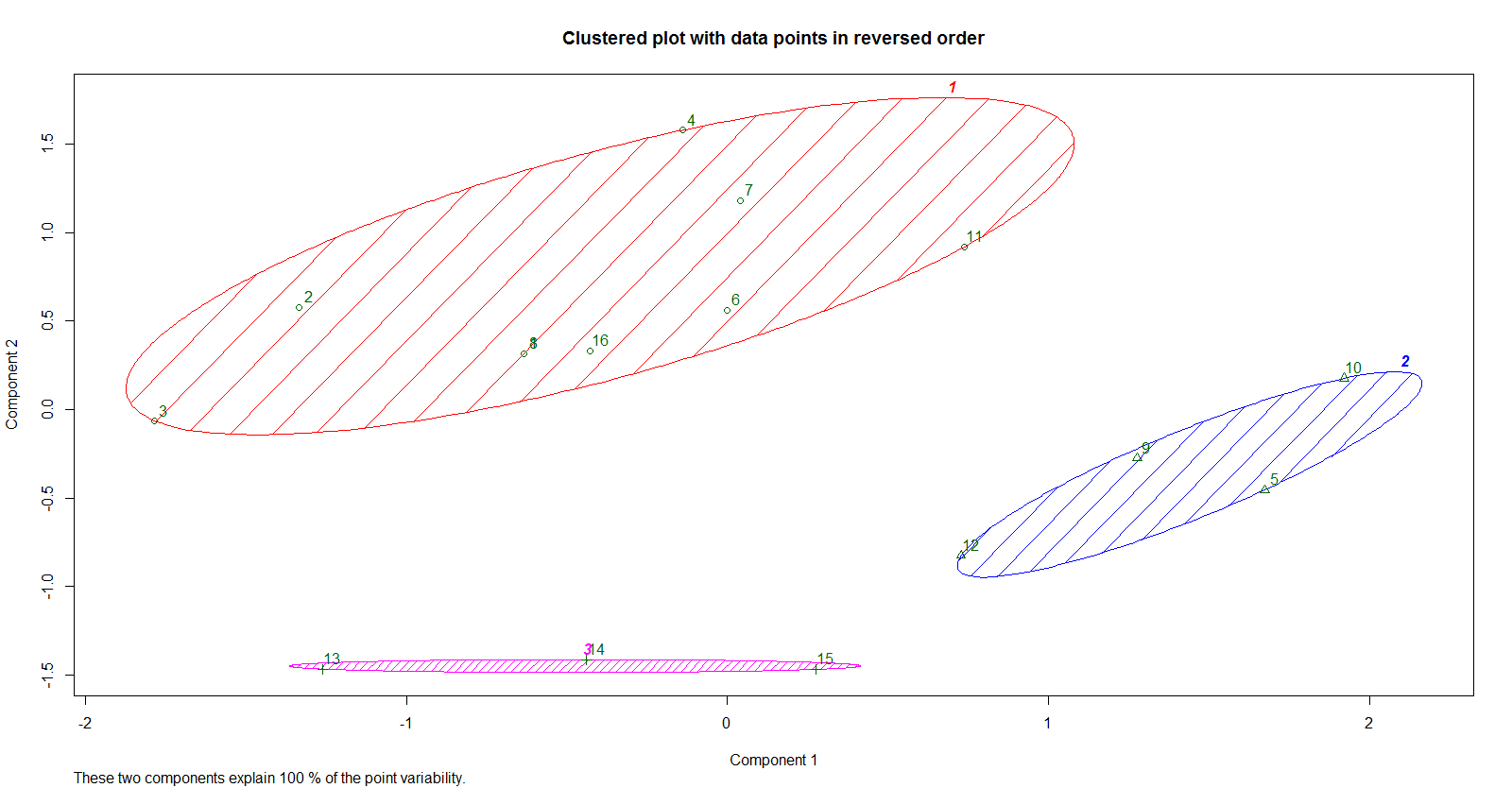
clusplot(ScaledData\_Rev, Clusters\_with\_center\_Rev[,5], color=TRUE, shade=TRUE, labels=2, lines = 0,

main = "Clustered plot with data points in reversed order")

**RESULTS:**

Clusters\_with\_center\_Rev [,5] = [1] 1 1 1 1 2 1 1 1 2 2 1 2 3 3 3 1

**SCREEN SHOT:**



**ANSWER 1 (c):**

*List of steps*

1. Using the fossil package in R to get the rand.index function.
2. Using rand.index function and finding the difference between the two clustering by using the cluster values from previous questions.
3. Find below the source code

**R SOURCE CODE:**

**#USING THE CLUSTER VALUES FROM THE PREVIOUS QUESTIONS WE GET**

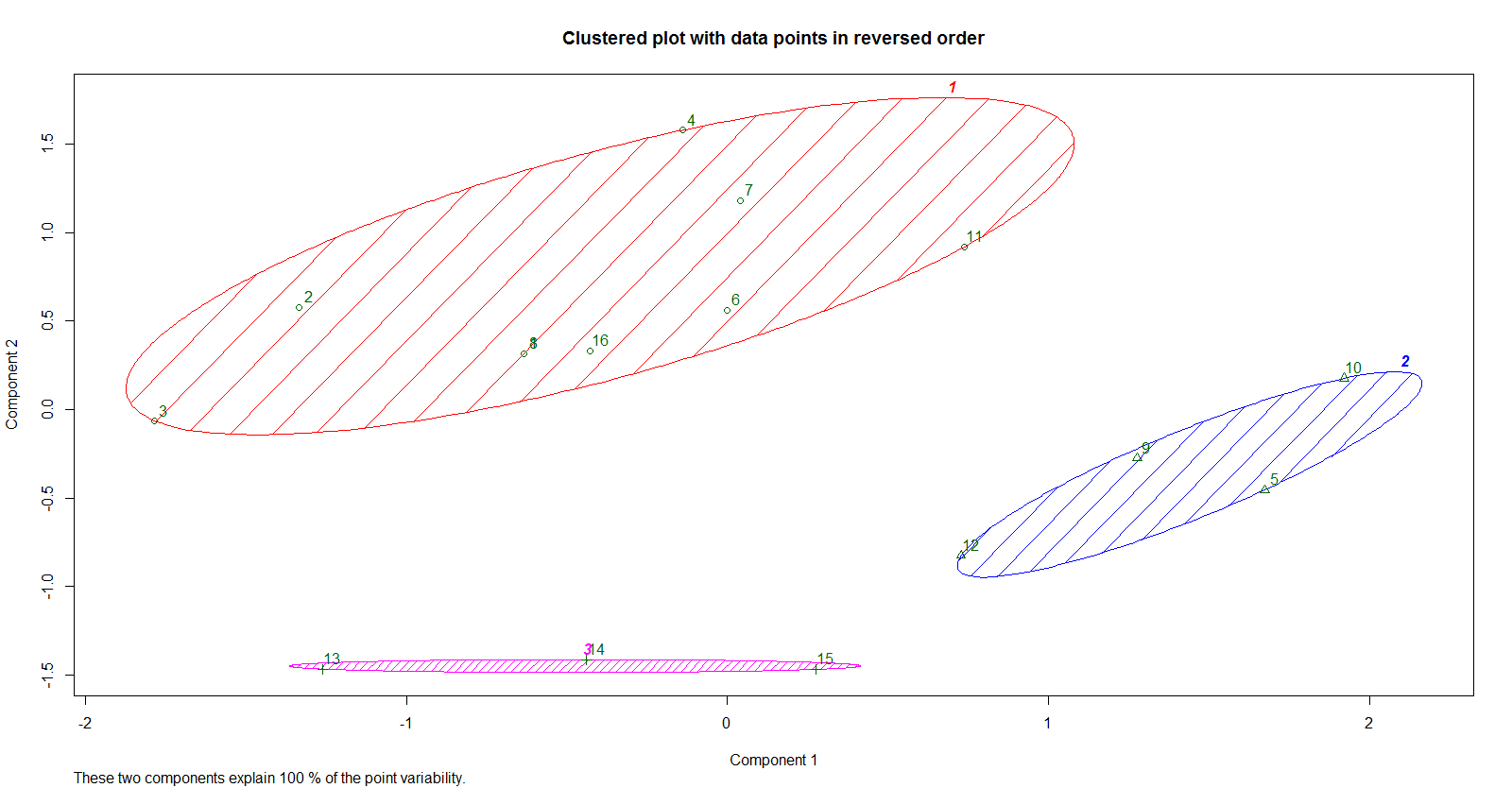
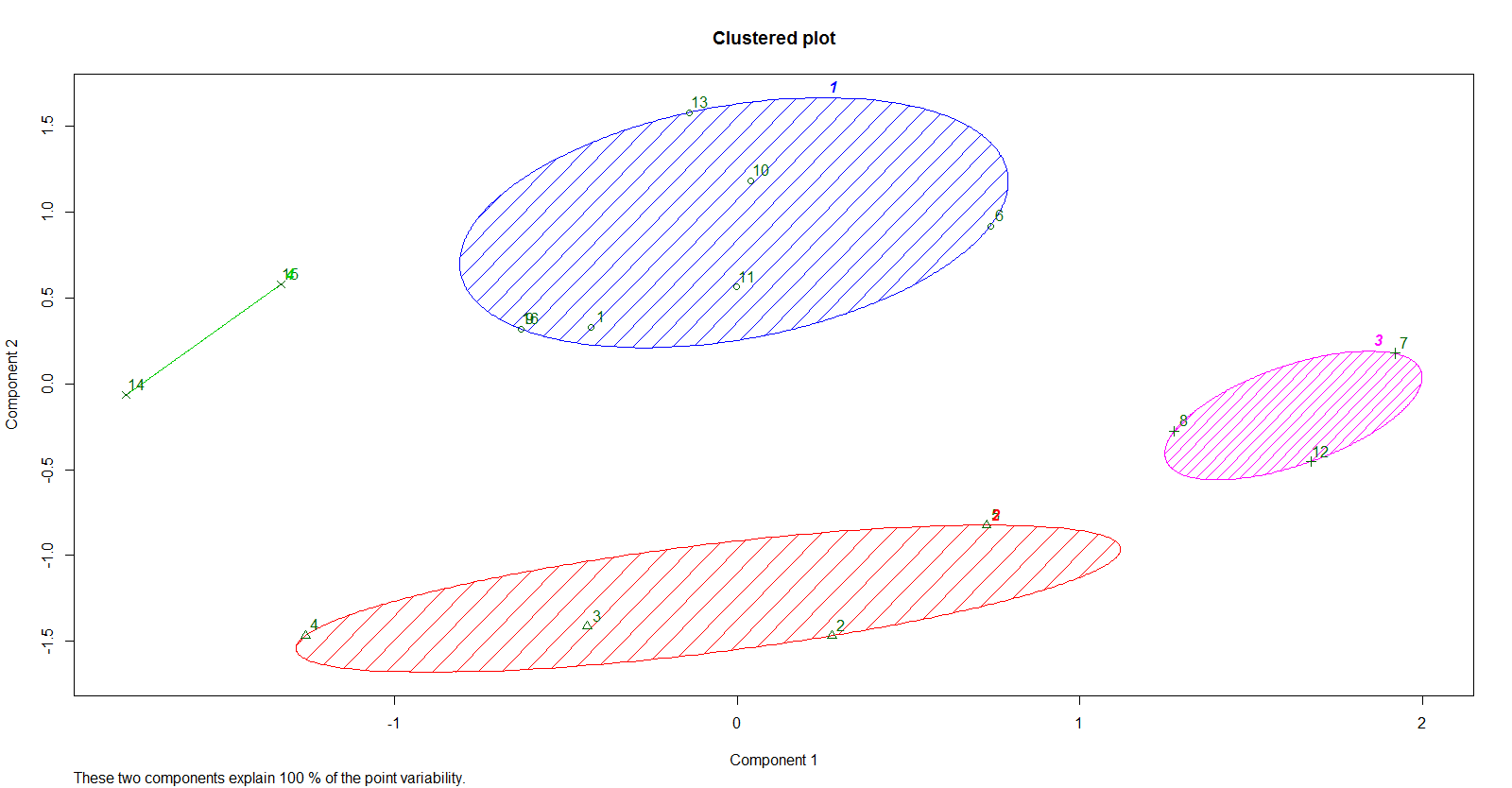
libray(fossil)

rand.index(Clusters\_with\_center[,5],Clusters\_with\_center\_Rev[,5])

**RESULTS:**

rand.index(Clusters\_with\_center[,5],Clusters\_with\_center\_Rev[,5]) = [1] 0.5666667

**IDENTIFICATION AND EXPLANATION:**



In bit (a), we get a plot with sequenced data, whereas in the second bit (b) the data is reversed and hence we can tell that the cluster is broken.

**ANSWER 2 (a):**

*List of steps*

1. Calculate the distance data frame using dist function of the given 2 dimensional data set
2. Using hclust function and method = single, find the hierarchical clustering
3. Later plot the clustering using plot function and find the dendrogram
4. Find the clusters for single linkage clustering using cutree function with number of clusters which is given in the question as 3
5. Plot the resulting clusters on a 2 dimensional space with boundaries around the clusters.
6. Find below the source code

**R SOURCE CODE:**

distance <- dist(TwoDdata)

Hierarchical\_clust\_single <- hclust(distance,method="single")

plot(Hierarchical\_clust\_single, main = "Hierarchical Clustering Dendrogram using single method")

clusters\_single <- cutree(Hierarchical\_clust\_single, k = 3)

rect.hclust(Hierarchical\_clust\_single, k = 3, border = "green")

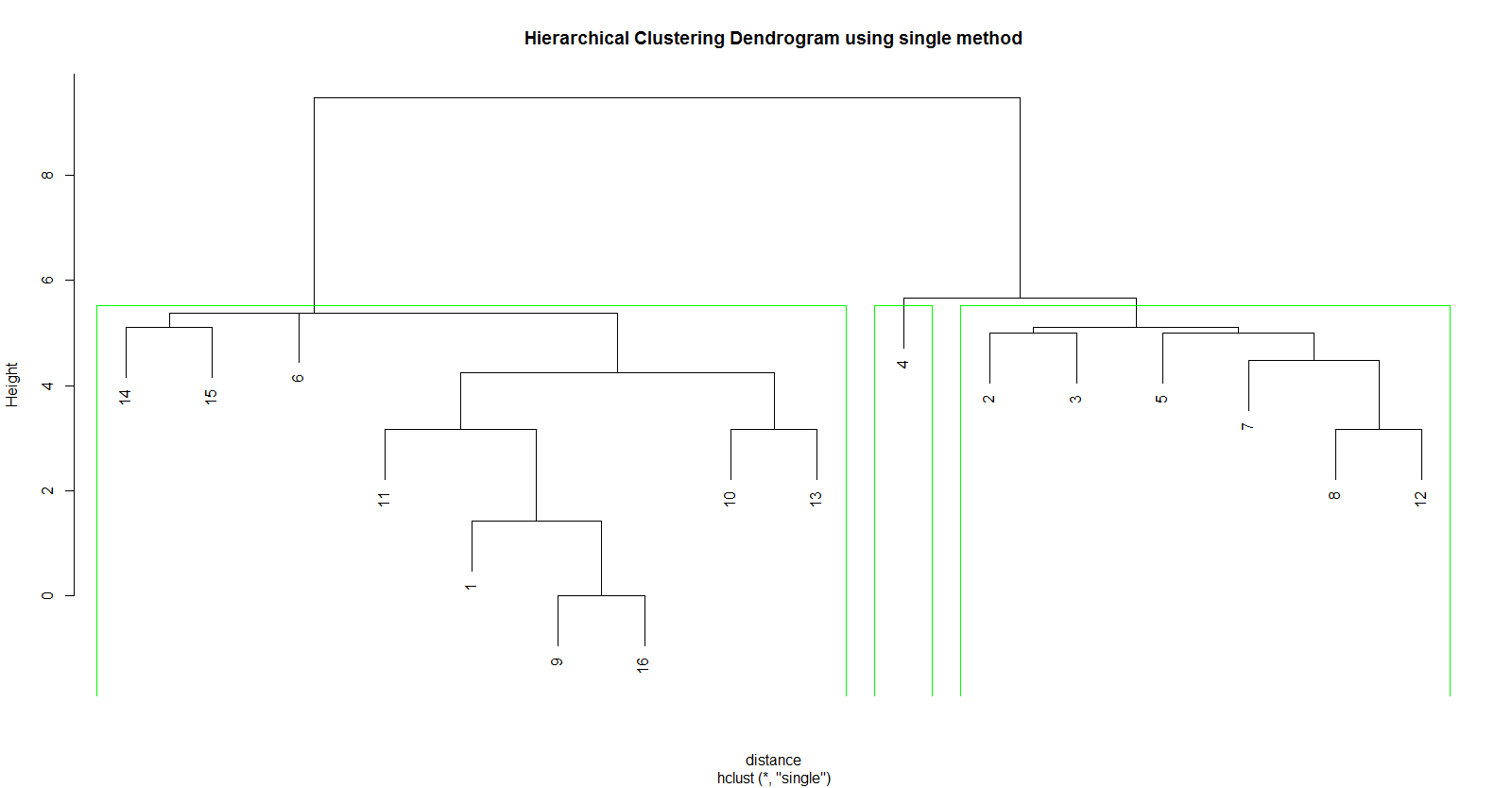
library(cluster)

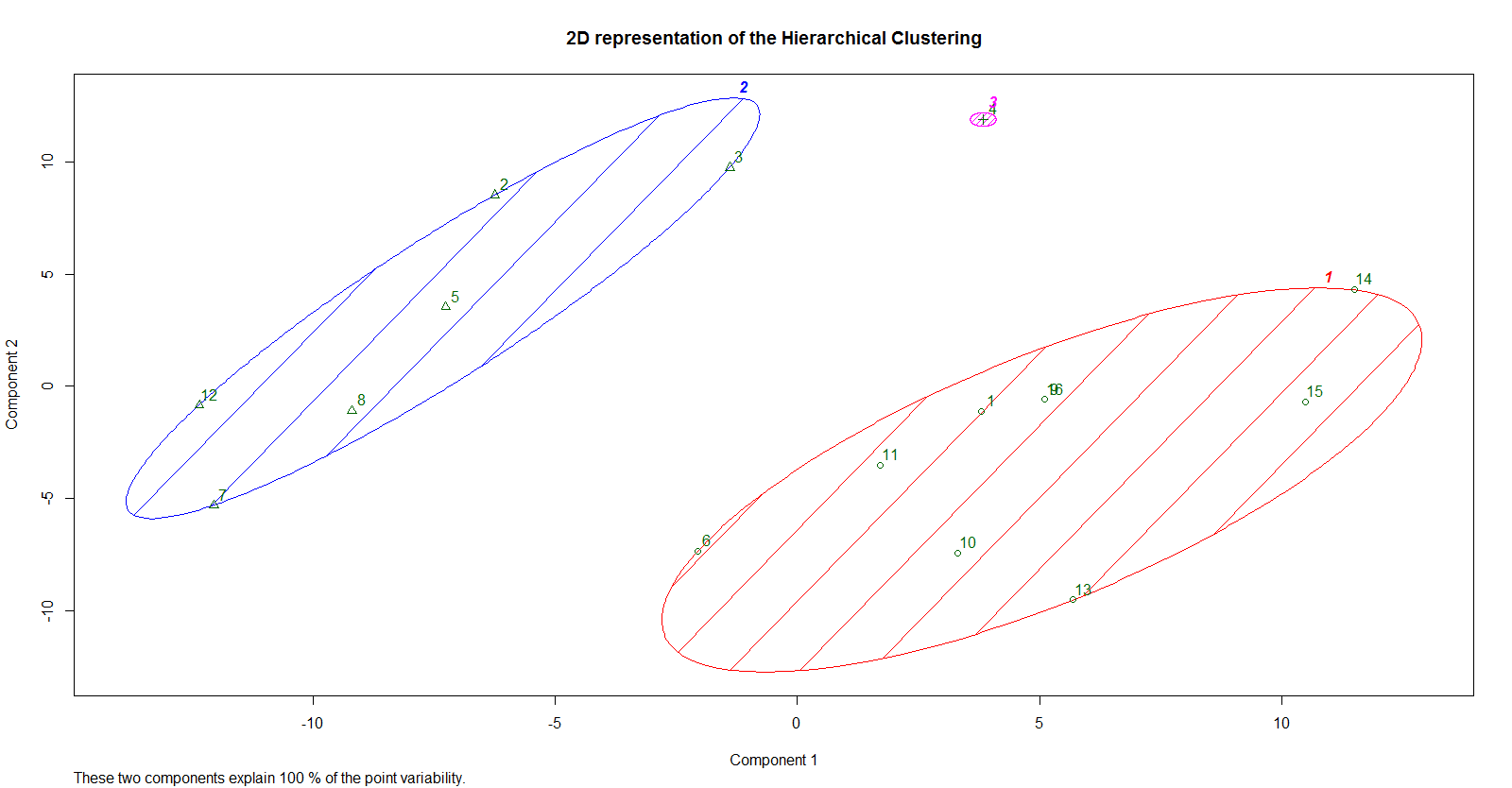
clusplot(TwoDdata, clusters\_single, main='2D representation of the Hierarchical Clustering',

color=TRUE, shade=TRUE, lines=0, labels=2)

**RESULTS:**

clusters\_single = [1] 1 2 2 3 2 1 2 2 1 1 1 2 1 1 1 1

**SCREEN SHOT:**



**ANSWER 2 (b):**

*List of steps*

1. Perform the same above steps as mentioned in Problem 2.a
2. Instead of method = single, use method = complete and find the hierarchical clustering
3. Repeat the steps in the previous problem
4. Find below the source code

**R SOURCE CODE:**

distance <- dist(TwoDdata)

Hierarchical\_clust\_complete <- hclust(distance,method="complete")

plot(Hierarchical\_clust\_complete, main = "Hierarchical Clustering Dendrogram using complete method")

clusters\_complete <- cutree(Hierarchical\_clust\_complete, k = 3)

rect.hclust(Hierarchical\_clust\_complete, k = 3, border = "red")

library(cluster)

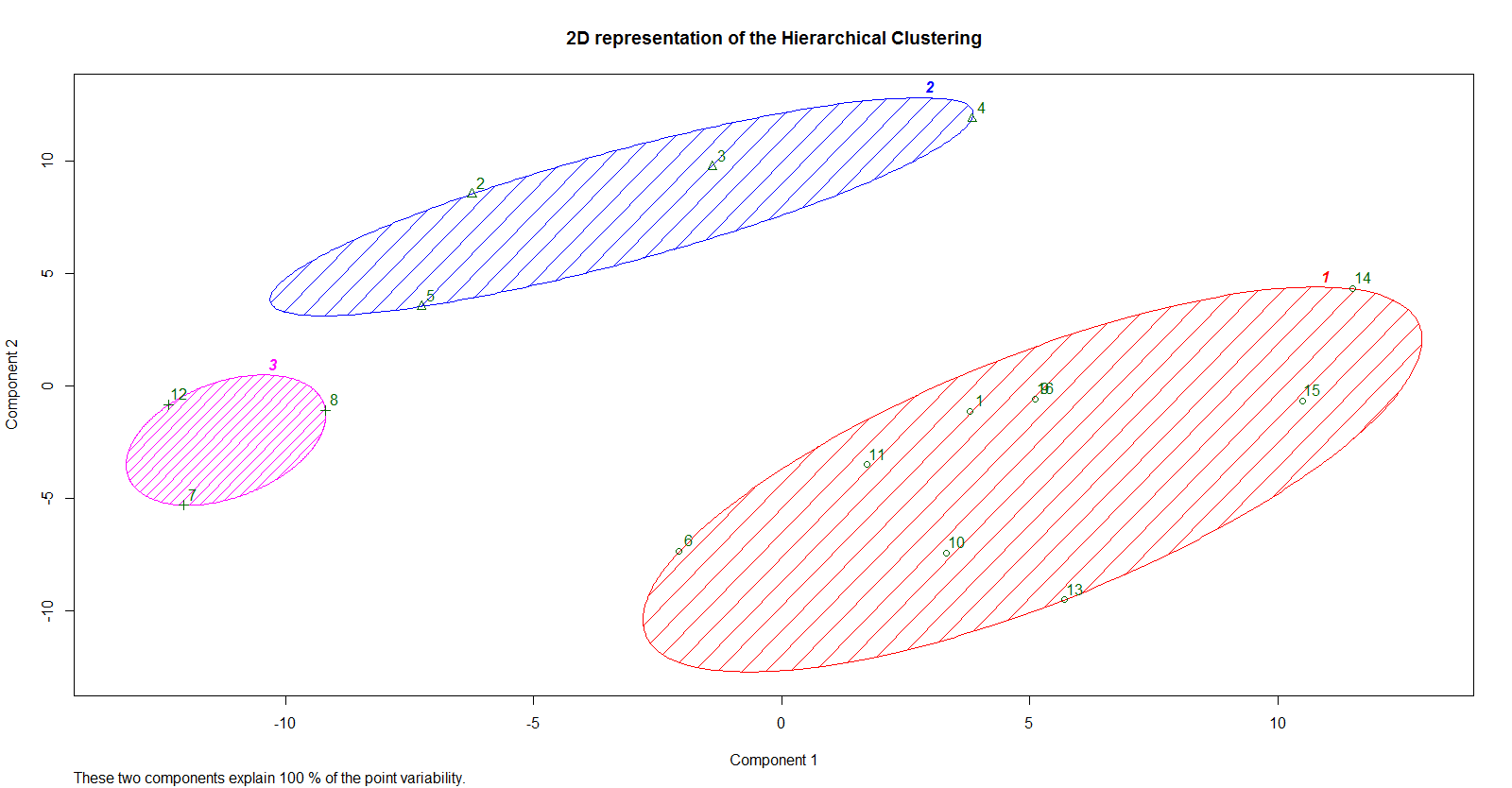
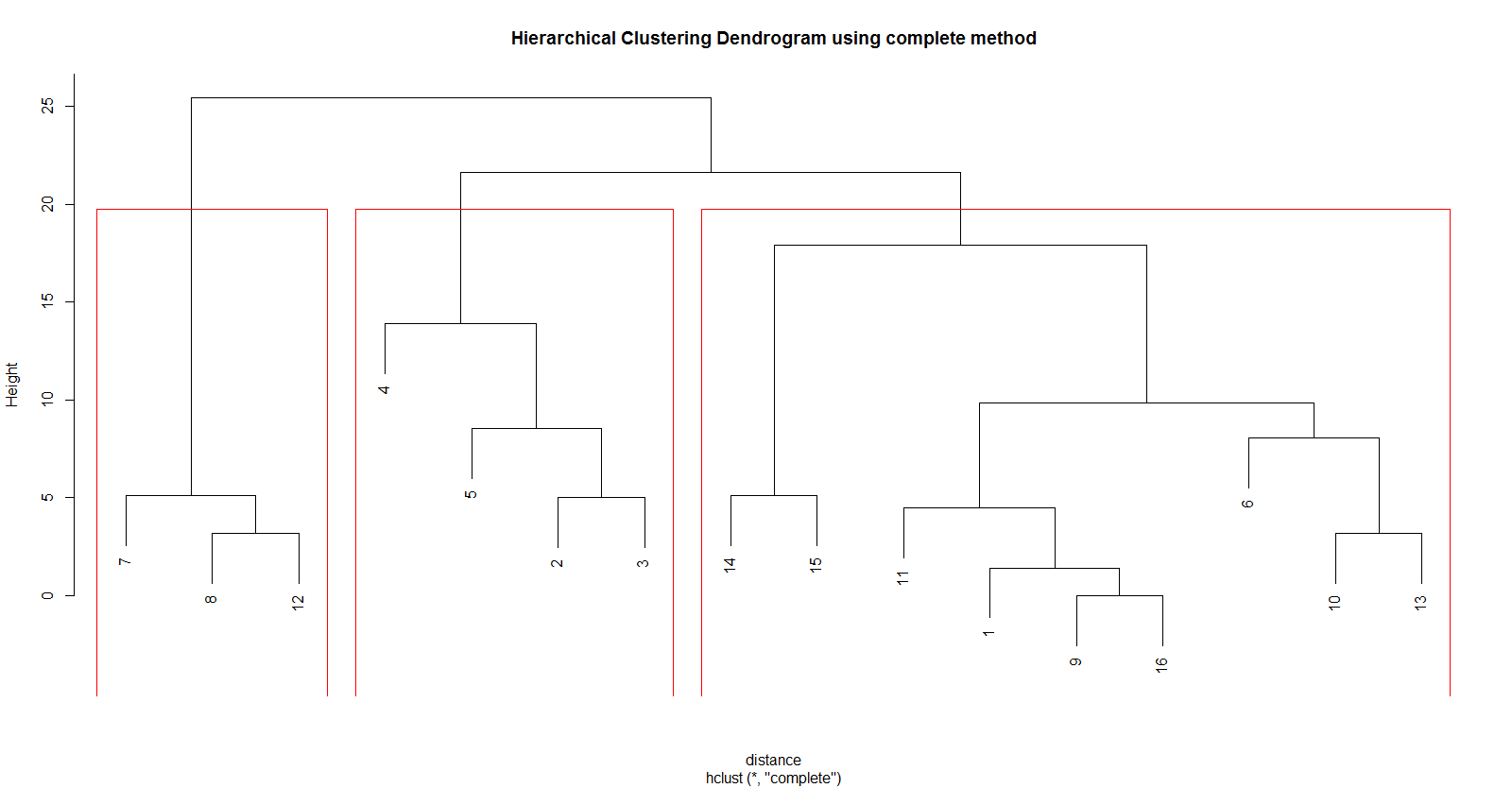
clusplot(TwoDdata, clusters\_complete, main='2D representation of the Hierarchical Clustering',

color=TRUE, shade=TRUE, lines=0, labels=2)

**RESULTS:**

clusters\_complete = [1] 1 2 2 2 2 1 3 3 1 1 1 3 1 1 1 1

**SCREEN SHOT:**



**ANSWER 2 (c):**

*List of steps*

1. To calculate the sum of square errors, we have to consider both the clusters that are obtained in above two questions
2. We have to initially consider X coordinates of the 2d data given and Y coordinates of the data given.
3. Finding the centroid of the whole data and then finding the distance between each cluster point to the centroid. Later the sum of distance between the centroid to the clusters is nothing but the sum of squared errors.
4. Finding the centroid can be done simply by adding the X and Y values and dividing it by the number of values.
5. Create a function for finding the distance between the centroid and the clusters
6. Find the total sum of the distance which is nothing but the Sum of squared errors.
7. Repeat the above steps for Single linkage as well as Complete linkage clusters.
8. Find below the source code

**R SOURCE CODE:**

#Single linkage clustering

CentroidXValues\_single <- array()

CentroidYValues\_single <- array()

for(i in 1:3)

{

CentroidXValues\_single[i] <- sum(XValues[which(clusters\_single==i)])/table(clusters\_single)[i]

CentroidYValues\_single[i] <- sum(YValues[which(clusters\_single==i)])/table(clusters\_single)[i]

}

DistanceFunctionSingle <- function(x,y)

{

x1 <- CentroidXValues\_single[y]

y1 <- CentroidYValues\_single[y]

x2 <- XValues[x]

y2 <- YValues[x]

return(sqrt((x1-x2)^2+(y1-y2)^2))

}

SSESingleArray <- array()

for(i in 1:3)

SSESingleArray[i] <- sum(DistanceFunctionSingle(which(clusters\_single==i),i))

SSESingle <- sum(SSESingleArray)

#Complete linkage clustering

CentroidXValues\_complete <- array()

CentroidYValues\_complete <- array()

for(i in 1:3)

{

CentroidXValues\_complete[i] <- sum(XValues[which(clusters\_complete==i)])/table(clusters\_complete)[i]

CentroidYValues\_complete[i] <- sum(YValues[which(clusters\_complete==i)])/table(clusters\_complete)[i]

}

DistanceFunctionComplete <- function(x,y)

{

x1 <- CentroidXValues\_complete[y]

y1 <- CentroidYValues\_complete[y]

x2 <- XValues[x]

y2 <- YValues[x]

return(sqrt((x1-x2)^2+(y1-y2)^2))

}

SSECompleteArray <- array()

for(i in 1:3)

SSECompleteArray[i] <- sum(DistanceFunctionComplete(which(clusters\_complete==i),i))

SSEComplete <- sum(SSECompleteArray)

**RESULTS:**

SSESingle = 80.99749  
SSEComplete = 72.44793

**COMPARE, CONTRAST AND IDENTIFY THE CLUSTER THAT IS CONTRIBUTING THE MOST:**

We get to know from the result that Sum of squared errors for Single linkage cluster is more than the sum of squared errors for the complete linkage cluster which explains that the number of values in a cluster increases, the value of the sum of squared errors also increases. From these values we see that, the first cluster has largest number of points, so it has a larger value of SSE when compared to the three other values. In the single link method, the SSE value of cluster 3 is zero, as it has only one point.

For identifying the cluster that is contributing the most we have to find the distances between the cluster and the centroid or squared error of each cluster which is already calculated above, the result which is shown below,

SSESingleArray = [1] 45.59519 35.40230 0.00000

SSECompleteArray = [1] 45.595192 19.486922 7.365813

From above it is clear that, the cluster 1 in both single link as well as complete link is the cluster which is contributing the most to the SSE value.

**ANSWER 2 (d):**

*List of steps*

1. For correlation analysis, we initially have to find the proximity matrix as well as the instance matrix.
2. Proximity matrix can be found using the dist function
3. For instance matrix, as mentioned in the question we have to build the binary matrix by marking 1’s wherever the point-pairs are in the same cluster.
4. Later using cor function (correlation function) we can find the correlation.
5. Find below the source code

**R SOURCE CODE:**

#Single linkage clustering

#proximity matrix

proximity <- dist(distance,upper=TRUE,diag=TRUE)

proximity\_matrix <- as.matrix(proximity)

#instance matrix for single linkage

instance\_matrix\_single <- matrix(nrow=length(XValues),ncol =length(YValues))

for(i in 1:length(XValues))

{

for(j in 1:length(YValues))

{

if(clusters\_single[i] == clusters\_single[j])

{

instance\_matrix\_single[i,j] <- 1

}

else

{

instance\_matrix\_single[i,j] <- 0

}

}

}

cor(c(instance\_matrix\_single),c(proximity\_matrix))

#instance matrix for complete linkage

instance\_matrix\_complete <- matrix(nrow=length(XValues),ncol =length(YValues))

for(i in 1:length(XValues))

{

for(j in 1:length(YValues))

{

if(clusters\_complete[i] == clusters\_complete[j])

{

instance\_matrix\_complete[i,j] <- 1

}

else

{

instance\_matrix\_complete[i,j] <- 0

}

}

}

cor(c(instance\_matrix\_complete),c(proximity\_matrix))

**RESULTS:**

For single linkage clustering,  
cor(c(instance\_matrix\_single),c(proximity\_matrix)) = -0.7700974

For complete linkage clustering,  
cor(c(instance\_matrix\_complete),c(proximity\_matrix)) = -0.7717141

**COMMENT AND INTUITIVE IDEAS ABOUT THE CLUSTERS:**

It is observed that the correlation of single linkage cluster is more than complete linkage cluster and also we get to know that both these clusters are negatively correlated.

**ANSWER 3 (a):**

*List of steps*

1. Enter the data as mentioned in the question
2. Use the package fpc and use the function dbscan and plot the result. Consider the conditions mentioned in the question that is epsilon = 4 and minimum points = 3.
3. Later use stripchart function to show the clusters on a number line
4. Find below the source code

**R SOURCE CODE:**

library(fpc)

OneDdata <- c(1, 3, 5, 6, 8, 11, 12, 13, 14, 15, 16, 22, 28, 32, 33, 34, 35, 36, 37, 42, 58)

DBScan\_E4 <- dbscan(OneDdata, eps=4, MinPts = 3, scale=FALSE, showplot=1)

plot(OneDdata,col = DBScan\_E4$cluster+2L)

stripchart(OneDdata)

**RESULTS:**

DBScan\_E4

dbscan Pts=21 MinPts=3 eps=4

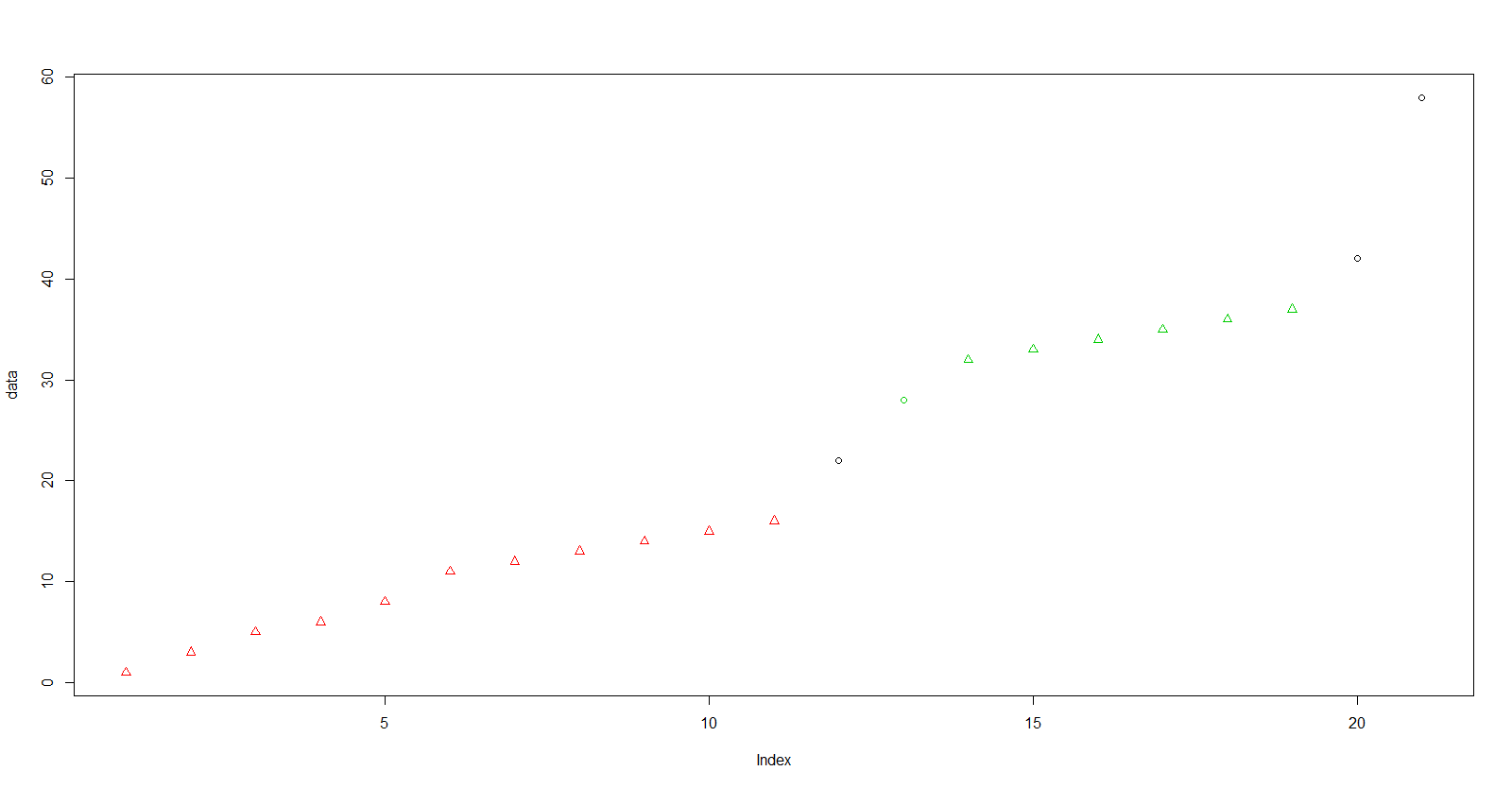
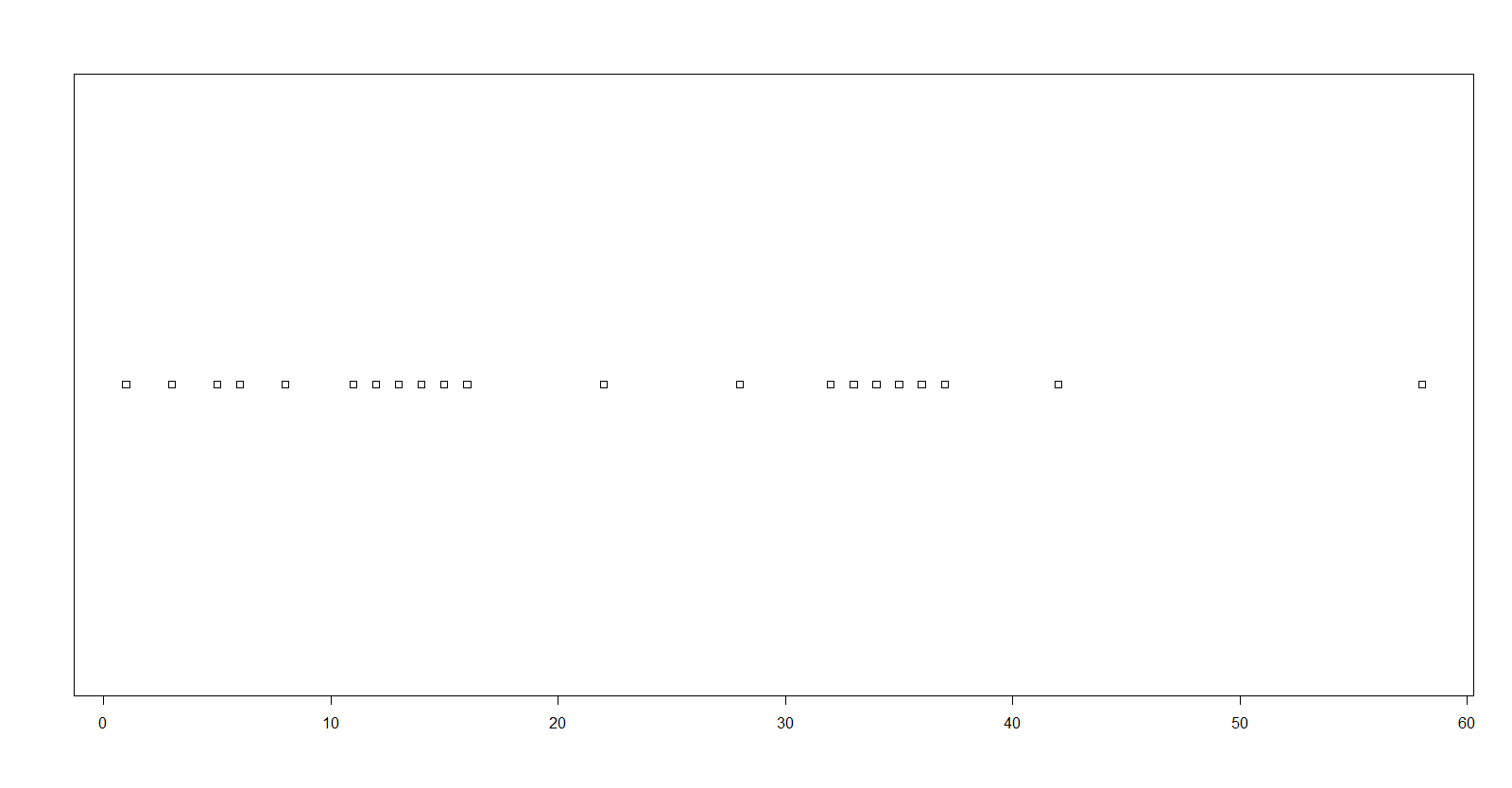
0 1 2

border 3 0 1

seed 0 11 6

total 3 11 7

**SCREEN SHOTS:**

**ANSWER 3 (b):**

*List of steps*

1. Repeat the same above steps with epsilon = 6 and minimum points = 3.
2. Later use stripchart function to show the clusters on a number line
3. Find below the source code

**R SOURCE CODE:**

DBScan\_E6 <- dbscan(OneDdata, eps=6, MinPts = 3, scale=FALSE, showplot=1)

plot(OneDdata,col = DBScan\_E6$cluster+2L)

stripchart(OneDdata)

**RESULTS:**

DBScan\_E6

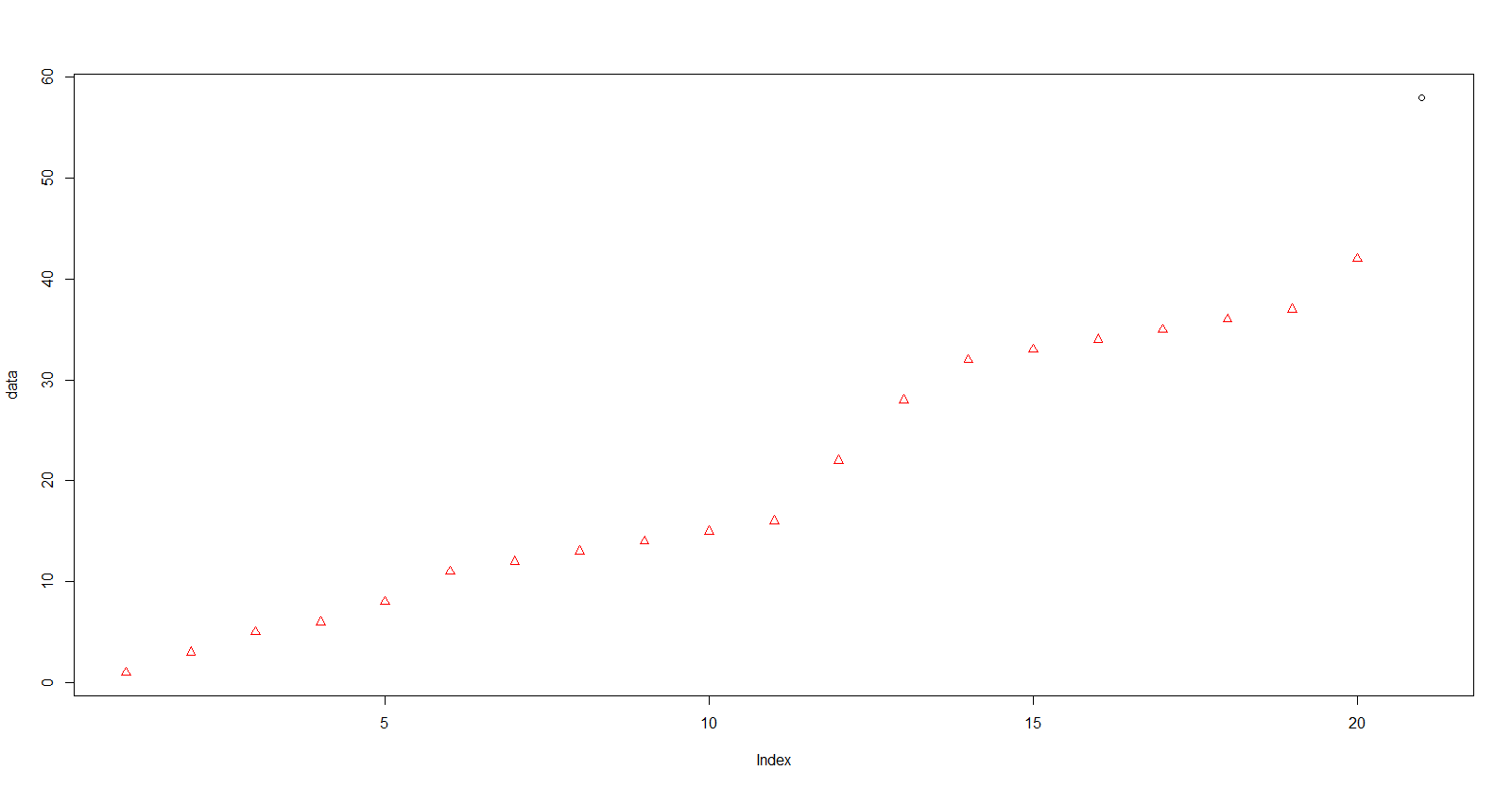
dbscan Pts=21 MinPts=3 eps=6

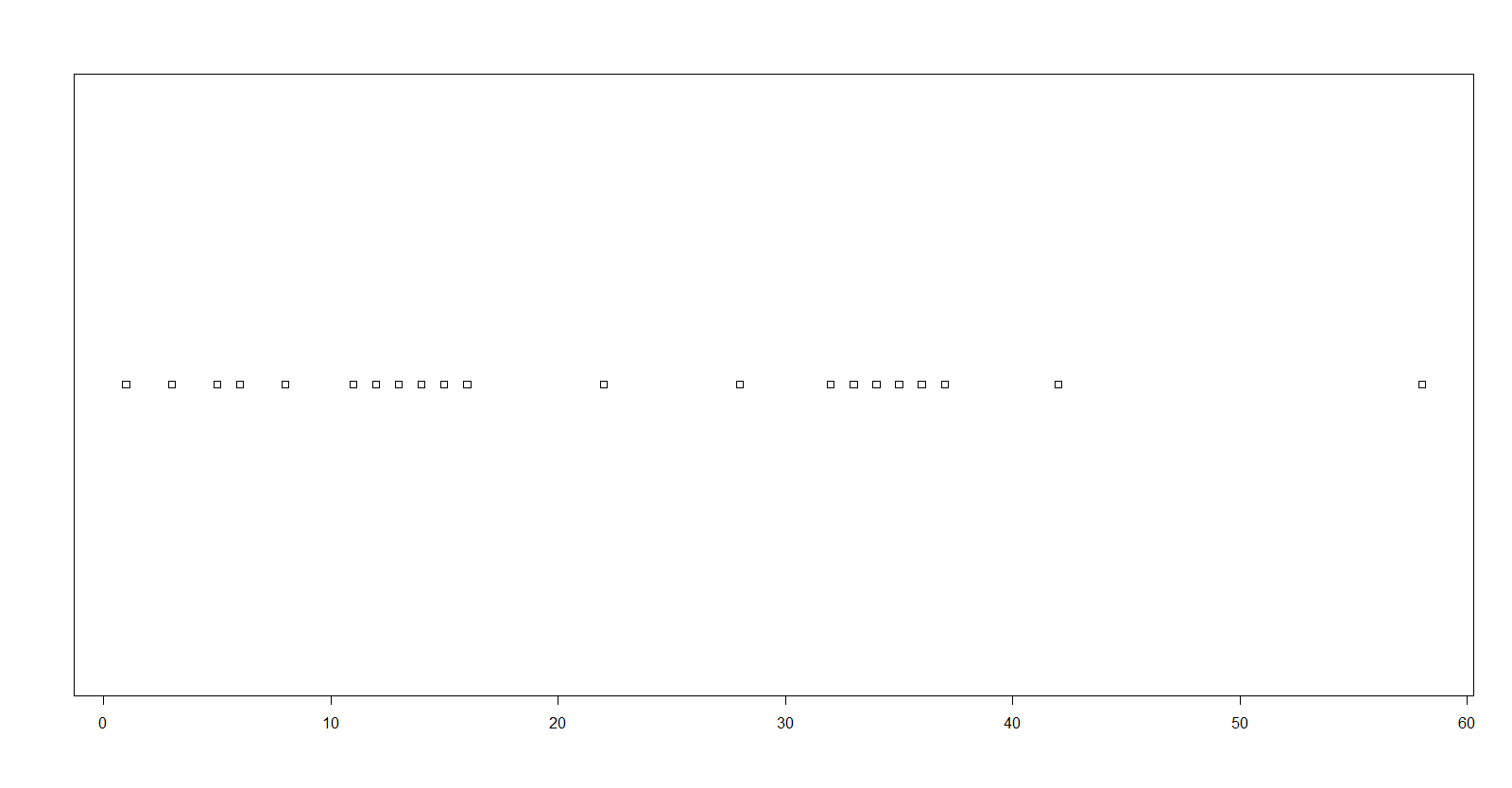
0 1

border 1 0

seed 0 20

total 1 20

**SCREEN SHOTS:** 



**ANSWER 3 (c):**

*List of steps*

1. Using the rand.index function from the package fossil to find the rand index of both the obtained clusters, find the R Code below

**R SOURCE CODE:**

**#USING THE CLUSTER VALUES FROM THE PREVIOUS QUESTIONS WE GET**

libray(fossil)

rand.index(DBScan\_E4$cluster,DBScan\_E6$cluster)

**RESULTS:**

rand.index(DBScan\_E4$cluster,DBScan\_E6$cluster) = [1] 0.452381