CDA532 Homework 3

UB ID: 50413349

Name: Shreeram GS

1) ISLR2 Ch 12 # 4:

a)

Soln: We say that there is not enough information to tell because:

In ***Single Linkage***, the distance between two clusters is the minimum distance between members of the two clusters.

In ***complete-linkage clustering***, the link between two clusters contains all element pairs, and the distance between clusters equals the distance between those two elements (one in each cluster) that are *farthest away from each other*.

In the case of clusters {1,2,3} and {4,5}, suppose the Euclidean distances between the two points of the cluster are as follows:

clusters	4	5
1	1.5	2.5
2	2.5	3.5
3	3.5	4.5

So here we see dist(1,4) = 1.5 , dist(1,5) = 2.5, dist(2,4) = 2.5 and so on.

From the definition above, single linkage dissimilarity would be equal to 1.5. Similarly, based on the above complete linkage clustering definition, we would observe that the complete linkage would fuse at 4.5.

Now let us consider a scenario where the inter-observation's distances are equal, that is each pair are spaced at equidistance. Here we see that all pairs have a Euclidean distance of 1.5, hence here both single linkage and complete linkage would fuse at 1.5

clusters	4	5
1	1.5	1.5
2	1.5	1.5
3	1.5	1.5

b) The definition of *single linkage* states that the distance between 2 clusters is the minimum distance between its members, and incomplete *linkage* states that distance between clusters equals the distance between those two elements that are *farthest away from each other*. So, in

the case of the clusters {5} and {6} fuse, they would fuse at the same height in both scenarios, as there is only one Euclidean distance between the two 2 clusters.

2) ISLR2 Ch 12 # 10:

Soln:

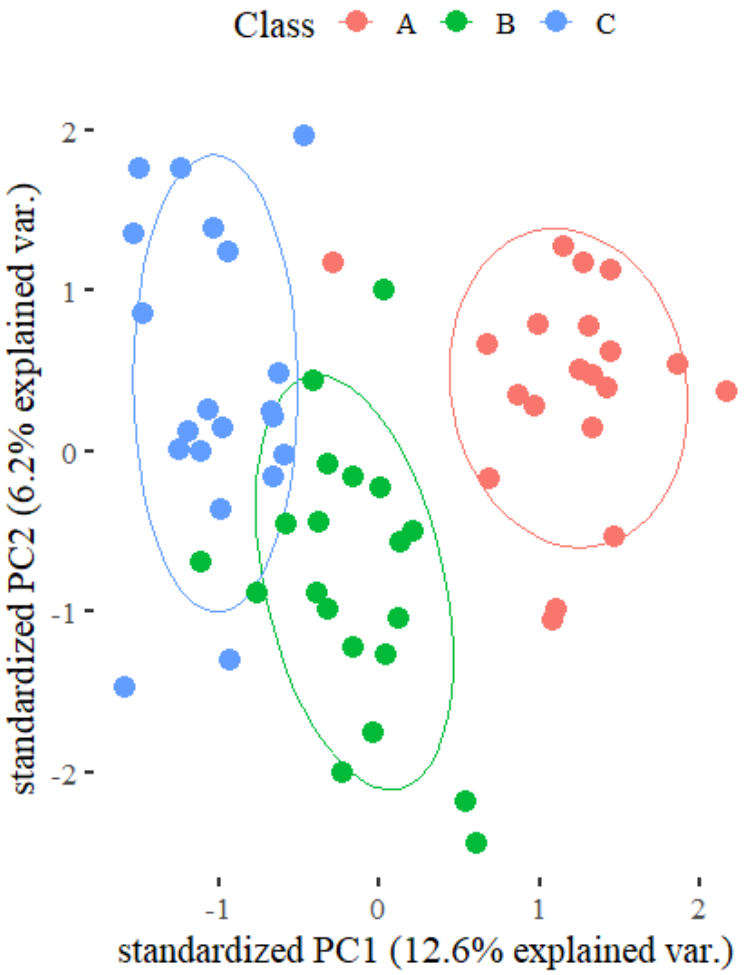
a) First, we create 3 classes with 20 observations each and 50 variables

```
ds <- data.frame(replicate(50, rnorm(20, mean = rnorm(1, mean = 0),  sd = 3))) %>%
  rbind(data.frame(replicate(50, rnorm(20, mean = rnorm(1, mean = 1), sd = 3)))) %>%
  rbind(data.frame(replicate(50, rnorm(20, mean = rnorm(1, mean = 2), sd = 3)))) %>%
  as.tibble %>%
  mutate(id = row_number(),
         class = ifelse(id <= 20, 'A',
                        ifelse(id <= 40, 'B',
                               'C'))) %>%
  select(-id)
```

b)

Next, we perform PCA as shown below using the prcomp() function on the 60 observations of the 3 classes, and we shall also move forward to apply various colors, to indicate the observations in each of the three classes.

```
#b)
require(ggbiplot); require(ggthemes)
pca <- prcomp(ds %>% select(-class), scale = TRUE)
x11()
ggbiplot(pca, groups = ds$class, var.axes = FALSE,
         ellipse = TRUE) +
  geom_point(aes(col = ds$class), size = 4) +
  theme_tufte(base_size = 16) +
  theme(legend.position = 'top') +
  guides(name = 'Groups') +
  scale_color_discrete(name = 'Class')
```

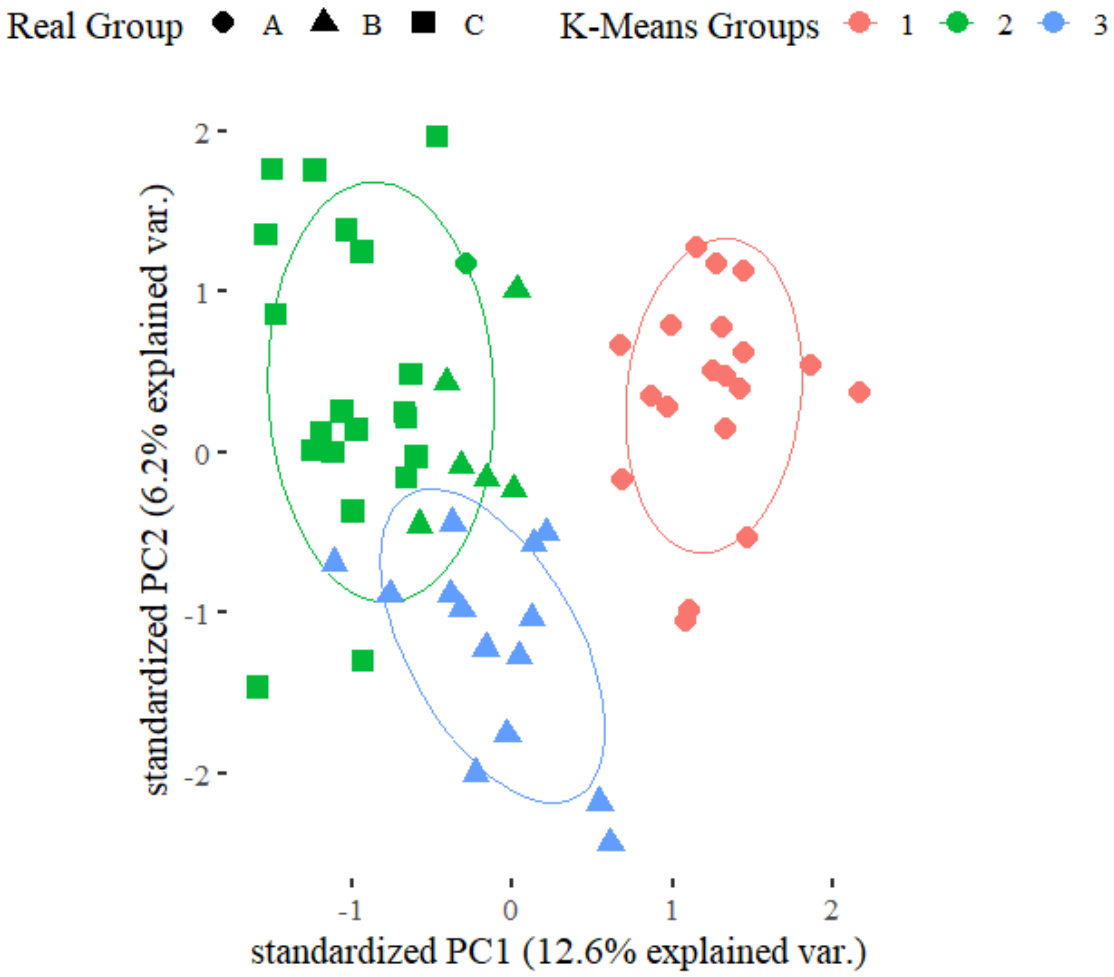


c)Next we shall see kmeans clustering of observations with k = 3. Here we do that below by specifying that centers = 3.

```
> scaling <- function(x) (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
> kmeans.clust <- ds %>%
+   select(-class) %>%
+   kmeans(centers = 3)
> x11()
> ggbiplot(pca, groups = factor(kmeans.clust$cluster), var.axes = FALSE,
+          ellipse = TRUE) +
+   geom_point(aes(shape = ds$class,
+                  col = factor(kmeans.clust$cluster)), size = 4) +
+   theme_tufte(base_size = 16) +
+   theme(legend.position = 'top') +
+   guides(name = 'Groups') +
+   theme(legend.position = 'top') +
+   scale_color_discrete(name = 'K-Means Groups') +
+   scale_shape_discrete(name = 'Real Group')

> true_class = c(rep(1,20), rep(2,20), rep(3,20))
> table(kmeans.clust$cluster, true_class)
   true_class
     1  2  3
  1  0 16  0
  2  1  4 20
  3 19  0  0
```

Above we can see that after kmeans cluster = 3, we get the above true class labels. We see how each of the 3 groups correlates with each other. Groups 1 and 3 are very closely related whereas there is a clear and more sparse separation between group 1,2 and group 2. A



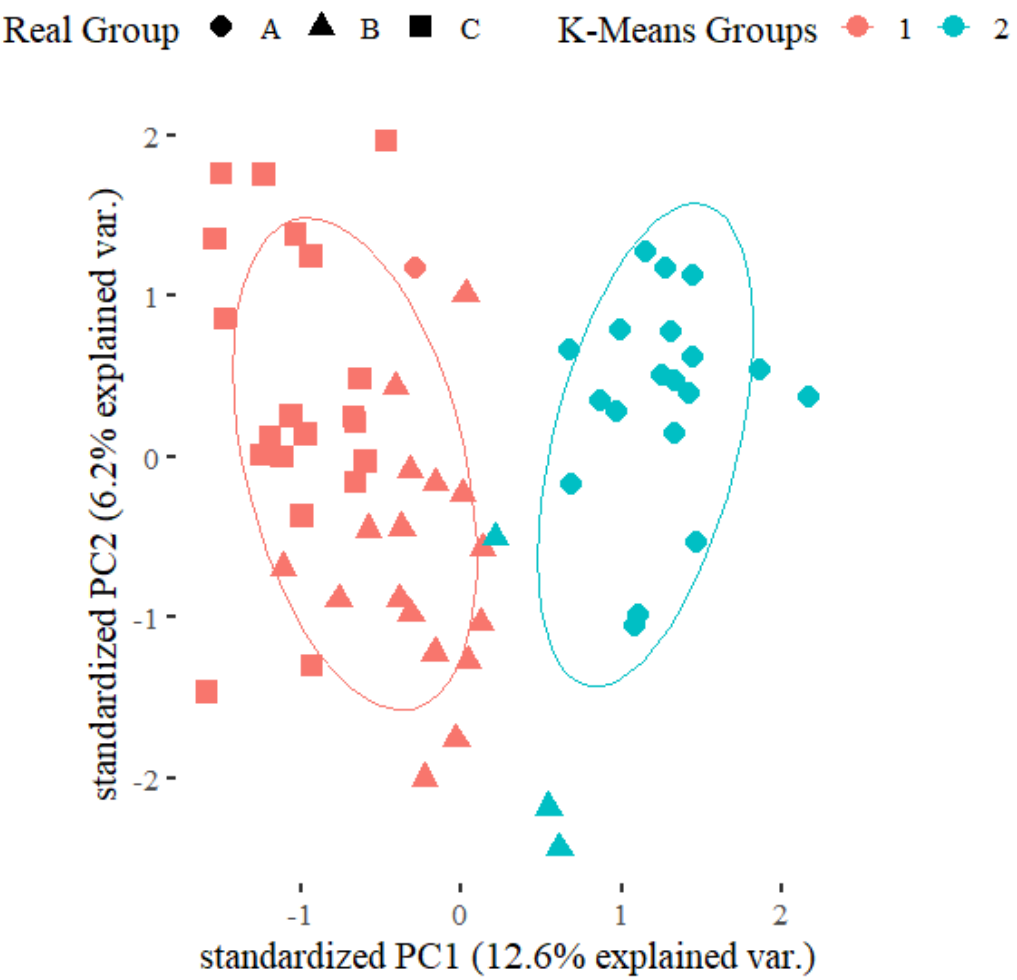
d)Below we do kmeans with k = 2 by specifying centers = 2.

```
> #d)
> kmeans.clust <- ds %>%
+   select(-class) %>%
+   kmeans(centers = 2)
> x11()
> ggbiplot(pca, groups = factor(kmeans.clust$cluster), var.axes = FALSE,
+          ellipse = TRUE) +
+   geom_point(aes(shape = ds$class,
+                  col = factor(kmeans.clust$cluster)), size = 4) +
+   theme_tufte(base_size = 16) +
+   theme(legend.position = 'top') +
+   guides(name = 'Groups') +
+   theme(legend.position = 'top') +
+   scale_color_discrete(name = 'K-Means Groups') +
+   scale_shape_discrete(name = 'Real Group')
```

Below we can see that after kmeans with k = 2, we get the above true class labels. We see how each of the 2 groups correlates with each other.

```
> true_class = c(rep(1,20), rep(2,20), rep(3,20))
> table(kmeans.clust$cluster, true_class)
   true_class
     1  2  3
  1 19  3  0
  2  1 17 20
```

The kmeans with k = 2 we get the above group to class correlation. We see that there is a fine even separation between both the groups. The two groups are almost completely separated and easy to differentiate.



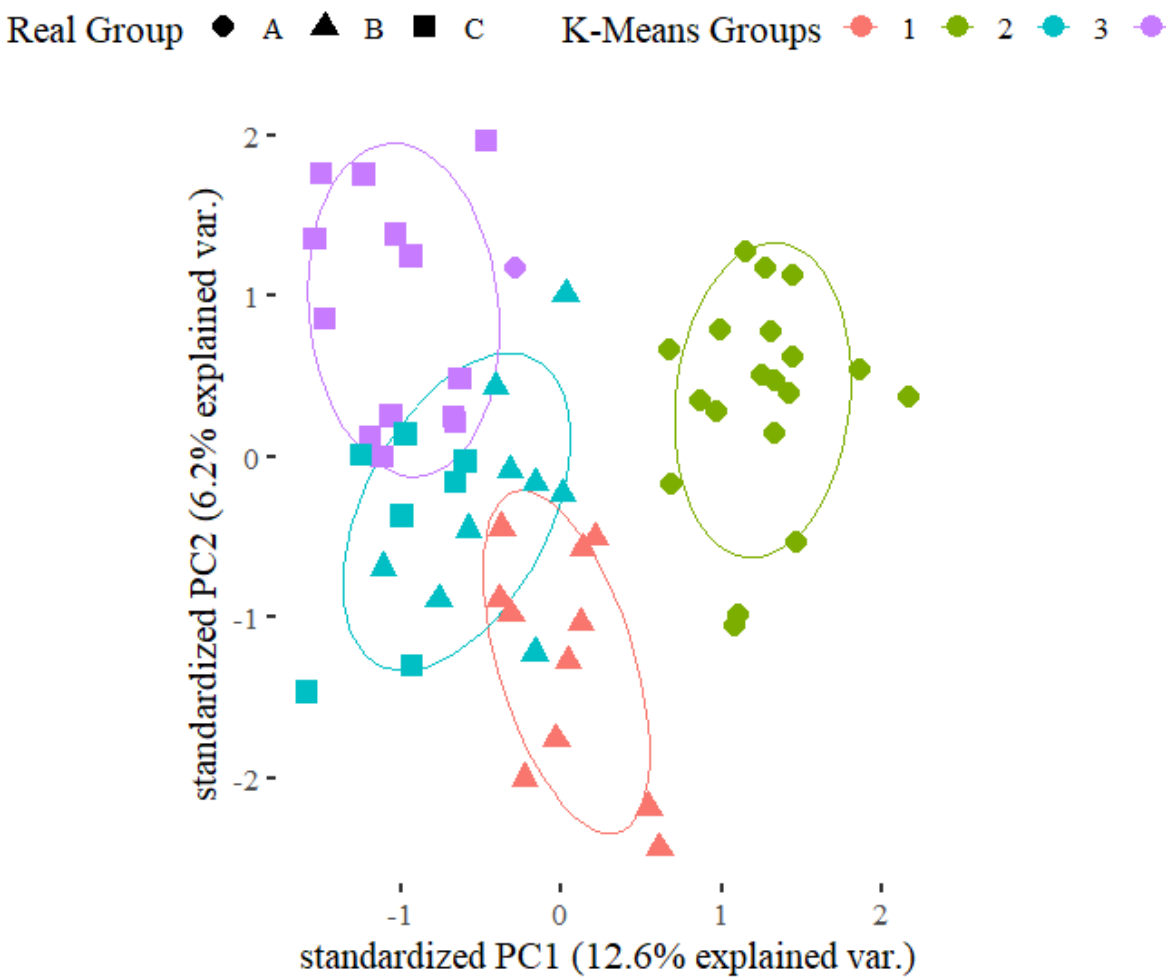
e) Now by setting centers = 4, that is kmeans of k= 4 neighbors.

```
#e)
kmeans.clust <- ds %>%
  select(-class) %>%
  kmeans(centers = 4)
X11()
ggbiplot(pca, groups = factor(kmeans.clust$cluster), var.axes = FALSE,
         ellipse = TRUE) +
  geom_point(aes(shape = ds$class,
                 col = factor(kmeans.clust$cluster)), size = 4) +
  theme_tufte(base_size = 16) +
  theme(legend.position = 'top') +
  guides(name = 'Groups') +
  theme(legend.position = 'top') +
  scale_color_discrete(name = 'K-Means Groups') +
  scale_shape_discrete(name = 'Real Group')
```

Below we see that cluster 1, 3, and 4 are near to each other, whereas they are a much clear separation between these group and group 2. We also see from the true label table the divide between each node.

```
true_class = c(rep(1,20), rep(2,20), rep(3,20))
table(kmeans.clust$cluster, true_class)
  true_class
    1  2  3
1 19  0  0
2  1  9  6
3  0 11  0
4  0  0 14
```

Above we see the 4 groups' true label with respect to the 3 classes.



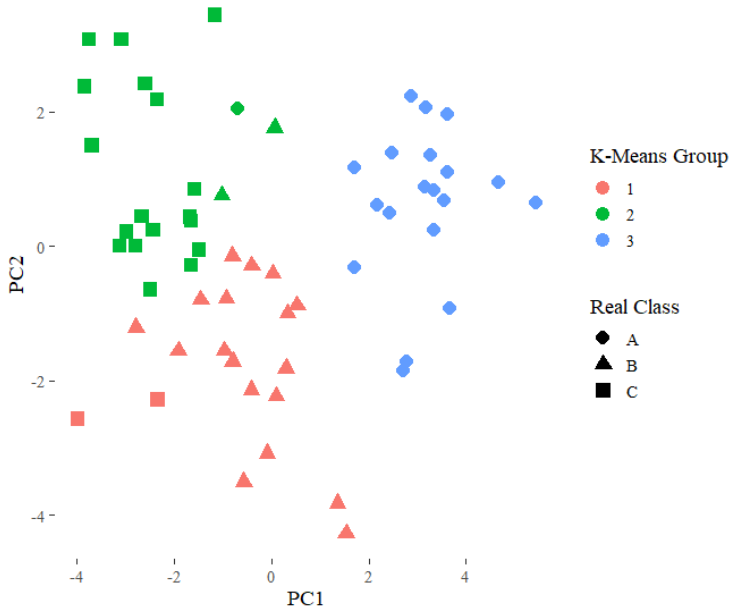
f) Now with kmeans k = 3, on the first two principal component score vectors

```
#f)
pca_2 <- pca$x %>%
  as.tibble %>%
  select(PC1, PC2)

pca_kmeans <- pca_2 %>%
  kmeans(centers = 3)
X11()
ggplot(pca_2, aes(PC1, PC2,
                  col = factor(pca_kmeans$cluster),
                  shape = ds$class)) +
  geom_point(size = 4) +
  theme_tufte(base_size = 14) +
  scale_color_discrete(name = 'K-Means Group') +
  scale_shape_discrete(name = 'Real Class')
```

```
> true_class = c(rep(1,20), rep(2,20), rep(3,20))
> table(pca_kmeans$cluster, true_class)
   true_class
     1  2  3
  1  0 18  2
  2  1  2 18
  3 19  0  0
```

Above the true labels table after the PCA and kmeans shows the correlation between the 3 kmeans groups. Here we see a much clear divide between group 1 and group 3 as shown by the true labels, and even after visualization we that they are comparatively more sparsely spaced. Whereas group 1 and group 2 are much nearer to each other.



g)

Next we shall that after scaling and performing kmeans with k = 3, we see a much more clean plot with all the groups being much more evenly spaced, with group 3 being the most

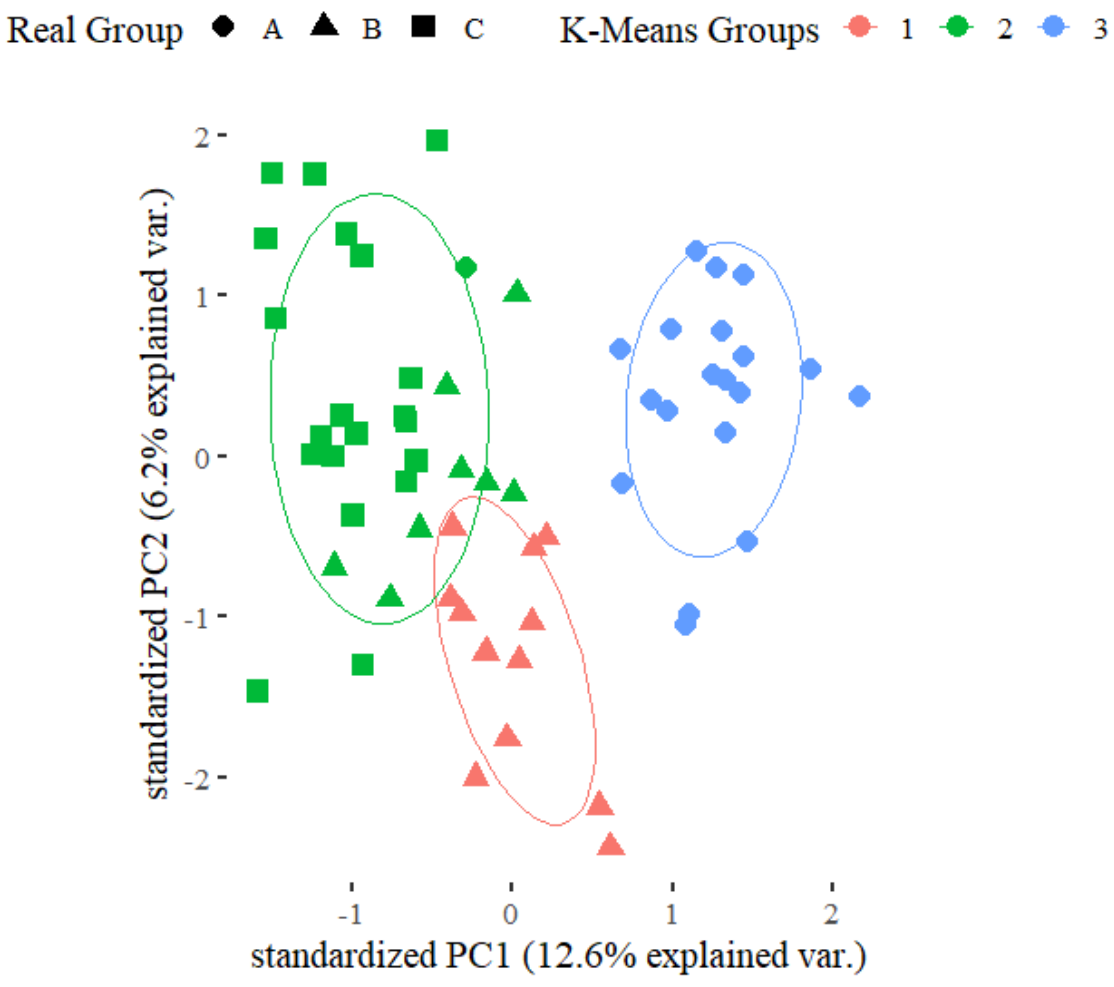
separated. Even between group 1 and group 2 there is a better and more clear divide and the nodes now can be more clearly separated.

```
#g)
scaling <- function(x) (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)

scaled_kmeans <- ds %>%
  select(-class) %>%
  map_df(scaling) %>%
  kmeans(centers = 3)

X11()
ggbiplot(pca, groups = factor(scaled_kmeans$cluster), var.axes = FALSE,
         ellipse = TRUE) +
  geom_point(aes(shape = ds$class,
                 col = factor(scaled_kmeans$cluster)), size = 4) +
  theme_tufte(base_size = 16) +
  theme(legend.position = 'top') +
  guides(name = 'Groups') +
  theme(legend.position = 'top') +
  scale_color_discrete(name = 'K-Means Groups') +
  scale_shape_discrete(name = 'Real Group')

> true_class = c(rep(1,20), rep(2,20), rep(3,20))
> table(scaled_kmeans$cluster, true_class)
   true_class
     1  2  3
  1  0 19  0
  2  1  1 20
  3 19  0  0
```



3) ISLR2 Ch 12 # 13:

Soln: a) First we shall load all the required libraries, download the dataset from the 2nd edition resources, and set the working directory to the location of the stored CSV file. Next, we shall use the read.csv() function, apply header = F(False), and store the CSV file in a data frame shown below.

```
> library(dendextend)
> library("ape")
> require(ape)
> require(RColorBrewer)
> require(corrplot)
> setwd("C:/Users/Sriram/Desktop/R projects/Sem 2/HW3")
> #a)
> #Read the csv file with header = F
> dataset_q13 = data <- read.csv("Ch12Ex13.csv",header = F)
```

b) Now we shall use the Pearson method to calculate the correlation distance and store the result in a variable.

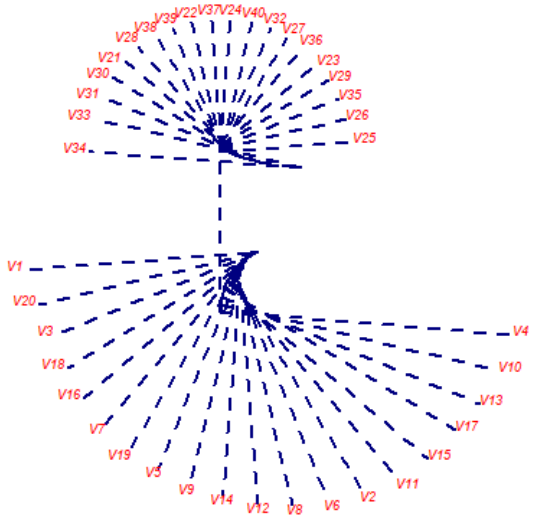
```
#b)
#finding correlation based distance
cdist <- dist(cor(dataset_q13, method = "pearson", use = "everything"))
```

Below we create a list of clustering methods, stored as follows, and then loop the methods accordingly. We also move forward to create an unrooted dendrogram to view the cluster.

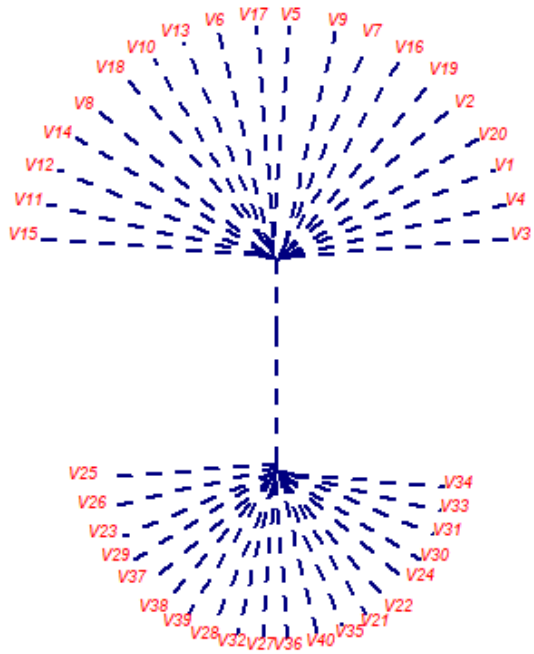
```
> methods <- c('centroid', 'average', 'single', 'complete')
> #Dendograms
>
> for (method in methods) {
+   clusts <- hclust(cdist, method = method)
+   x11()
+   plot(as.phylo(clusts), type = "unrooted",
+        cex = 0.6,
+        col = "#487AA1", col.main = "#45ADA8",
+        col.lab = "#7C8071", col.axis = "#F38630",
+        edge.color = "navyblue",
+        edge.width = 2, edge.lty = 2,
+        tip.color = "red",
+        main = paste0('Cluster Dendrogram using ', method, ' metric'))
+ }
```

Below are the results of Various plots with their respective clustering techniques.

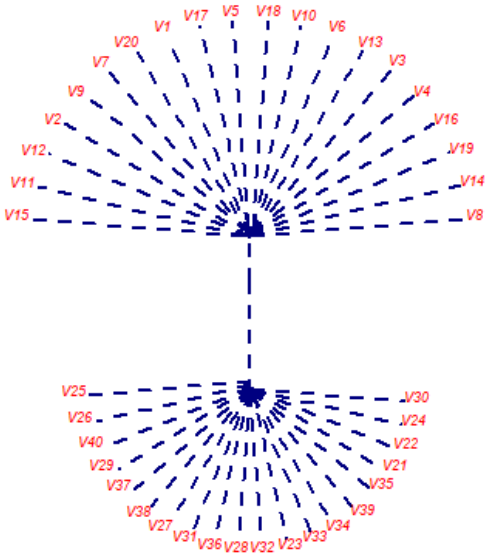
Cluster Dendrogram using centroid metric



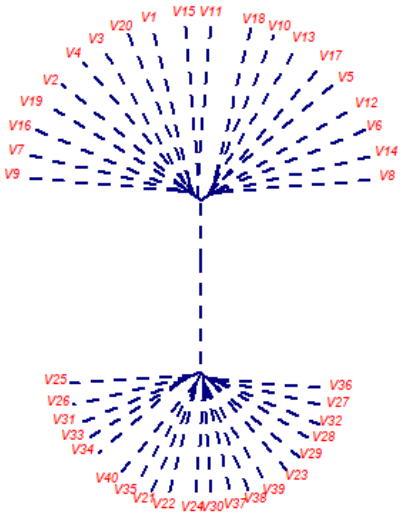
Cluster Dendrogram using average metric



Cluster Dendrogram using single metric



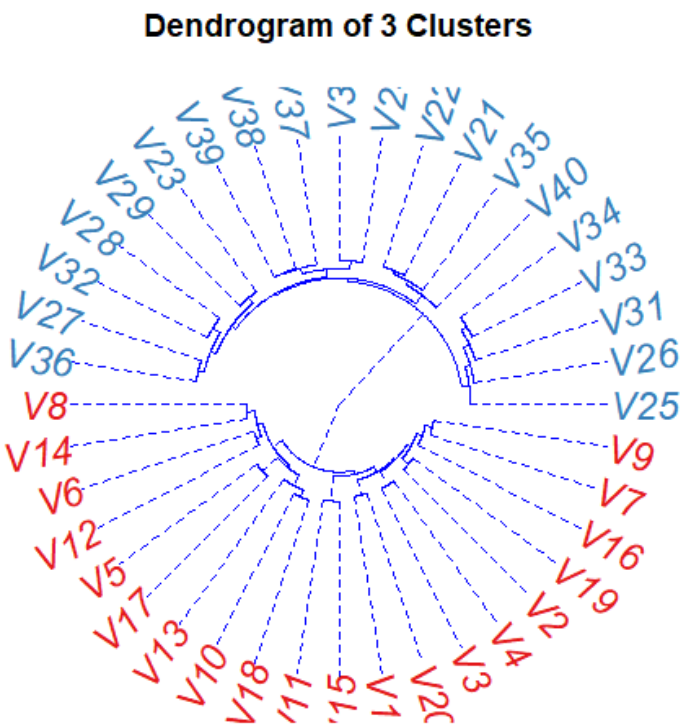
Cluster Dendrogram using complete metric



All the clustering techniques above hint that the variables are being separated into 2 parts.

Next, to get a further better understanding of the clustering we shall move forward to cut the dendrogram into two clusters and plot a fan plot to visualize the results.

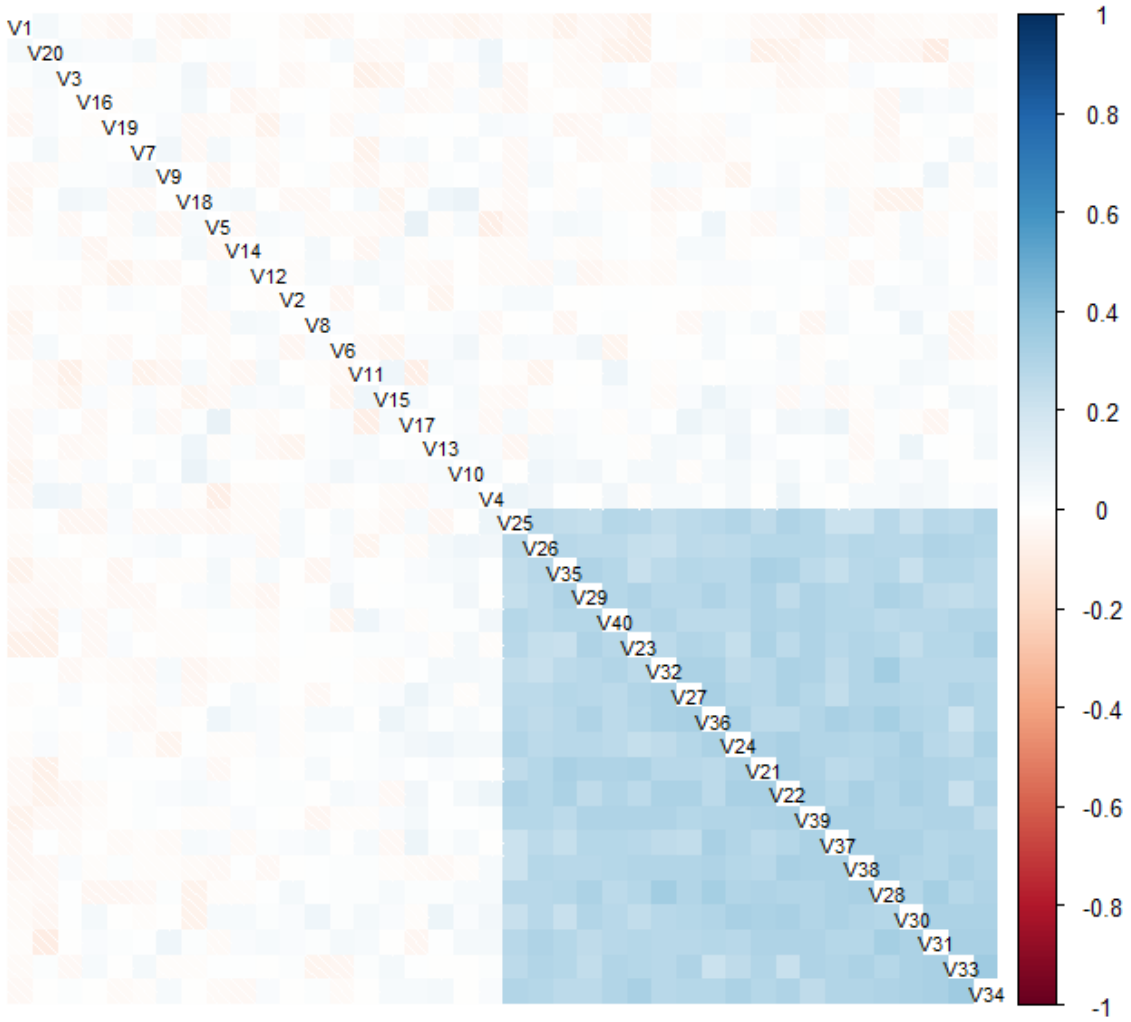
```
> #cut into two clusters as mainly all the above graphs are cut into 2 clusters
>
> x11()
>
> plot(as.phylo(clusts), type = 'fan',
+      tip.color = brewer.pal(2, 'Set1')[cutree(clusts, 2)],
+      edge.color = 'blue', edge.lty = 2,
+      cex = 1.5,
+      main = 'Dendrogram of cut Clusters')
Warning message:
In brewer.pal(2, "Set1") :
  minimal value for n is 3, returning requested palette with 3 different levels
```



As shown below, shall perform a correlation plot and use centroid clustering to show the correlation between the variables.

```
> #corplot
> x11()
> corrplot(cor(dataset_q13), method = 'shade',
+          diag = FALSE, order = 'hclust',
+          rect.col = 'red',
+          rect.lwd = 3, tl.pos = 'd',
+          tl.col = 'black', tl.cex = 0.7,
+          hclust.method = 'centroid',
+          col = COL2('RdBu', 200))
>
```

Again, below we see that there is an existence of two different groups.



Now we see that the genes divide the sample into two separate groups. We see that using any of the linkage techniques, we get the same results of the separation.

#c) Next, we shall import the essential plotting packages and move forward to perform PCA Analysis. This method reduces the dimensionality of the dataset and increases interpretability.

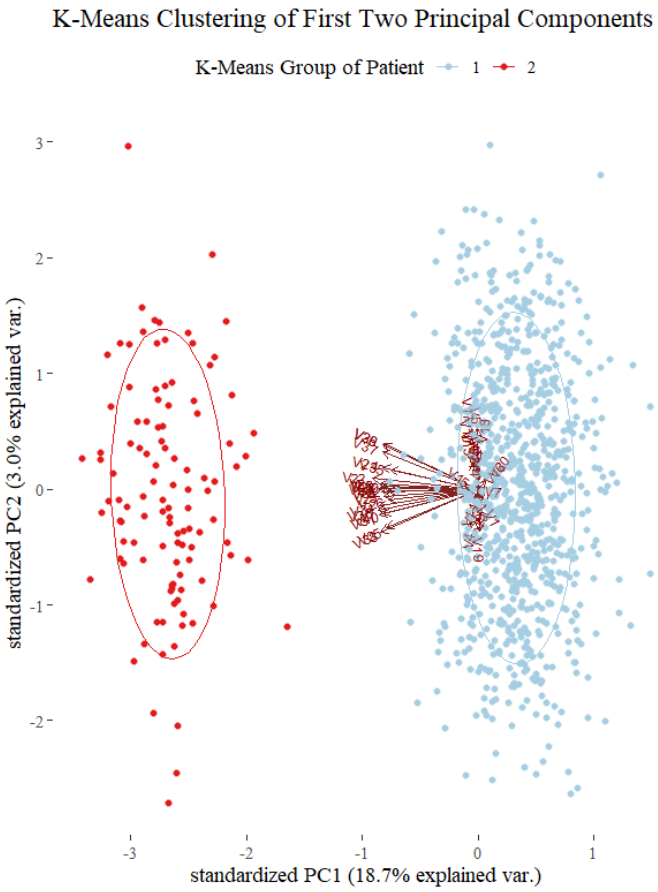
Further, we also perform K-Means clustering on the first two dimensions, and consequently, we shall make plots to see if there is a good separation between those components.

```
> require(ggbiplot)
> require(ggthemes)
> pca_ds <- prcomp(dataset_q13)
>
> pca_1 <- pca_ds$x %>%
+   as.tibble %>%
+   select(PC1, PC2)
```

```
kmeans_ds <- pca_1 %>%
  kmeans(centers = 2)
```

Below is the plot of the K-means clustering of the first two principal components using the ggbiplot visualization function.

```
> x11()
> ggbiplot(pca_ds, groups = factor(kmeans_ds$cluster),
+          ellipse = TRUE) +
+   theme_tufte(base_size = 14) +
+   geom_point(aes(col = factor(kmeans_ds$cluster)),
+              size = 2, alpha = 0.2) +
+   theme(legend.position = 'top') +
+   scale_color_manual(name = 'K-Means Group of Patient',
+                      values = c('#a6cee3', '#e31a1c')) +
+   ggtitle('K-Means Clustering of First Two Principal Components')
```



After the PCA analysis followed by kmeans, from the plots above we can confirm our assumption regarding the existence of the divide between the variables, that load heavily onto the first factor and variables that load heavily onto the second.

```
> sep_genes <- dataset_q13 %>%
+   dplyr::mutate(variable = paste0('Gene', row_number())) %>%
+   filter(kmeans_ds$cluster == 1) %>%
+   select(variable)
> my_tab <- table(sep_genes)
> View(my_tab)
> nrow(sep_genes)
[1] 110
```

	sep_genes	Freq
1	Gene11	1
2	Gene12	1
3	Gene13	1
4	Gene14	1
5	Gene15	1
6	Gene16	1
7	Gene17	1
8	Gene18	1
9	Gene19	1
10	Gene20	1
11	Gene501	1
12	Gene502	1
13	Gene503	1
14	Gene504	1
15	Gene505	1
16	Gene506	1
17	Gene507	1
18	Gene508	1

In addition to the first factor, there is a clear separation of patients. I will now see which variables most strongly correlate with it.