**Experiment 9:**

Implement k-means clustering using R.

**Solution:**

**K-Means algorithm**

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

***Algorithm summary***

K-means algorithm can be summarized as follow:

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

***Algorithm implementation***

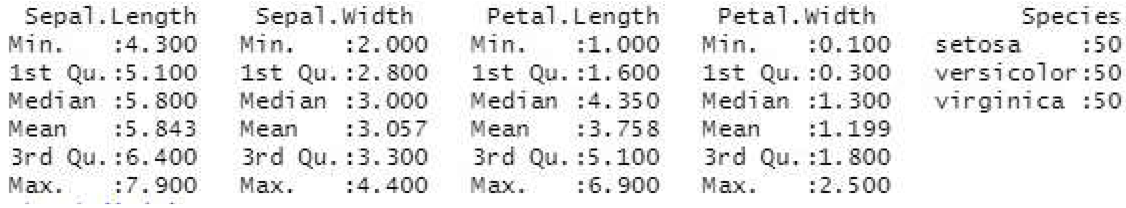
1. Install the relevant packages and call their libraries

```
> install.packages("dplyr")
> install.packages("ggplot2")
> install.packages("ggfortify")
> library("ggplot2")
> library("dplyr")
```

> library("ggfortify")

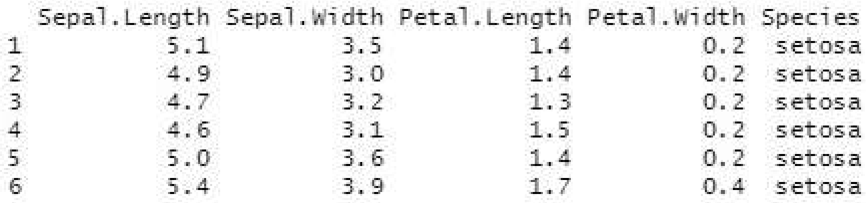
2. Loading and analyzing the dataset

> summary ("iris")



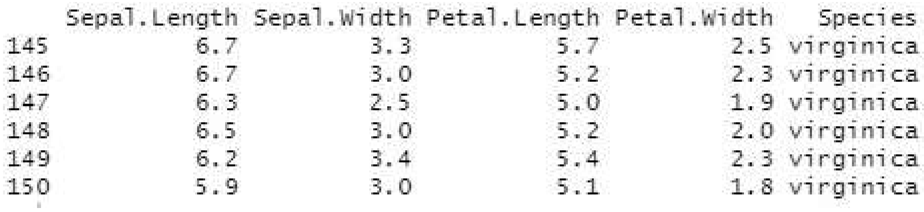
```
 Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species  
Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50  
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50  
Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50  
Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199                  
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800                  
Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500                  
```

> head ("iris")



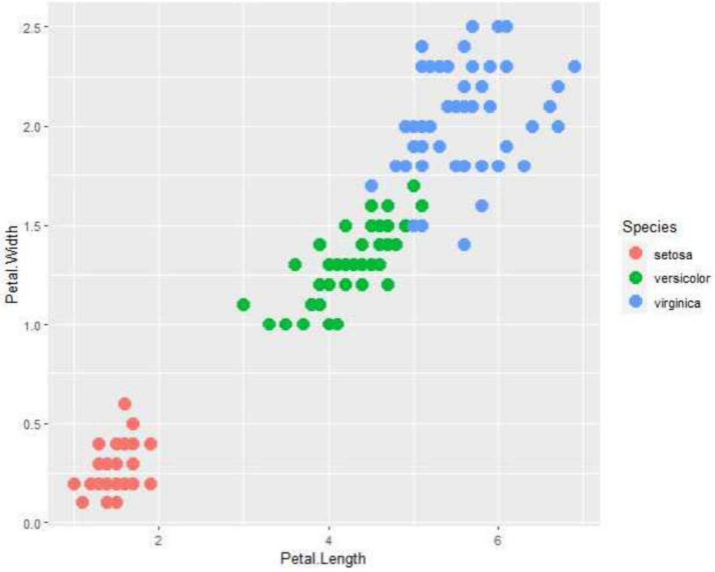
```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

> tail ("iris")



```
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
145          6.7         3.3          5.7         2.5 virginica
146          6.7         3.0          5.2         2.3 virginica
147          6.3         2.5          5.0         1.9 virginica
148          6.5         3.0          5.2         2.0 virginica
149          6.2         3.4          5.4         2.3 virginica
150          5.9         3.0          5.1         1.8 virginica
```

> ggplot(iris)+aes(Petal.Length,Petal.Width)+geom_point(aes(col=Species),size=4)



3. Eliminating the target variable

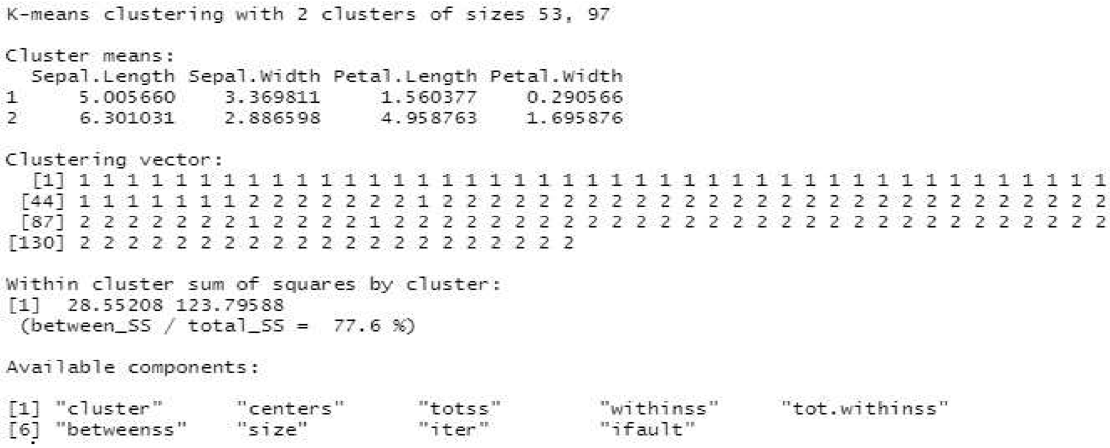
> data <- select (iris, c(1:4))

4. Implement k-means algorithm

K-means algorithm requires dataframe and k value and the command is

> kmean <- kmeans (data, 2)

> kmean

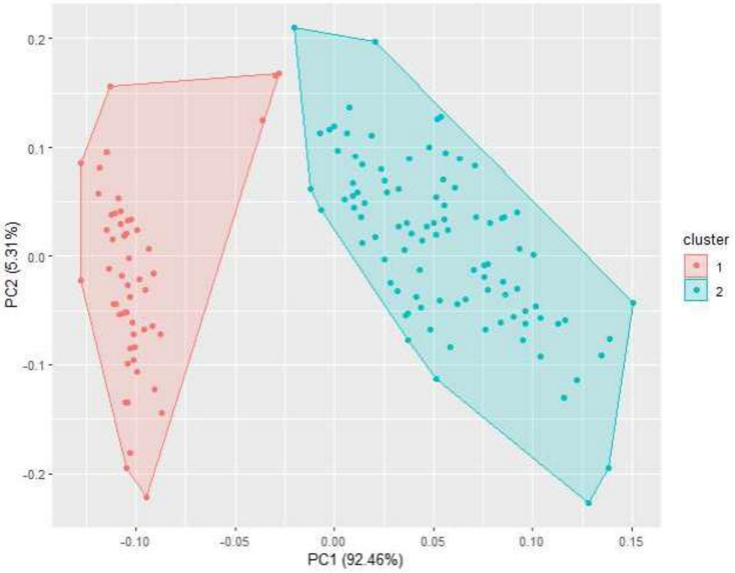


```
K-means clustering with 2 clusters of sizes 53, 97

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.005660    3.369811     1.560377    0.290566
2     6.301031    2.886598     4.958763    1.695876

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [44] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [87] 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[130] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1]  28.55208 123.79588
 (between_SS / total_SS =  77.6 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

***kmean$clusters*** return a vector of numbers ranging from 1 to 2, representing which observations belong to cluster 1 and cluster 2. ***kmean$centers*** return the location of each centroid. For e.g. cluster 1 has a mean value of Sepal.Length = 5.00, Sepal.width = 3.36, Petal.Length = 1.56 and Petal.width = 0.29.

5. Plotting our data-points in clusters

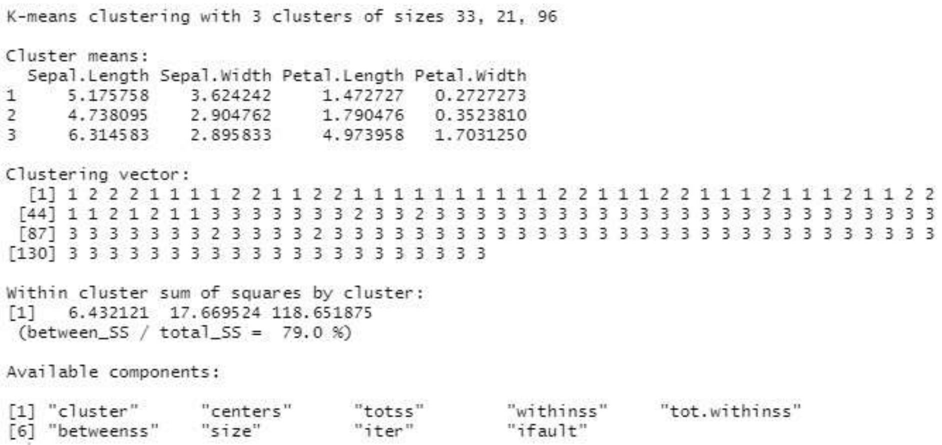
> autoplot (kmean, data, frame=TRUE)



6. K-means with k = 3

> kmean3 <- kmeans (data, 3)

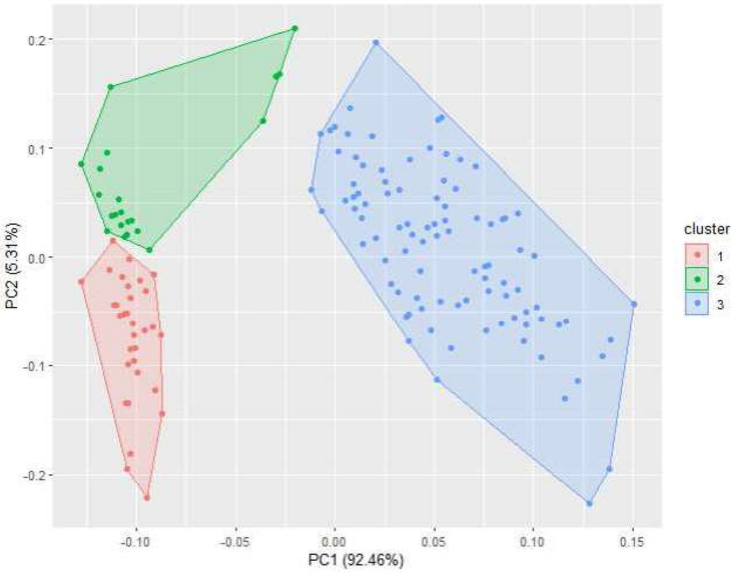
> kmean3



```
K-means clustering with 3 clusters of sizes 33, 21, 96

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.175758    3.624242     1.472727   0.2727273
2     4.738095    2.904762     1.790476   0.3523810
3     6.314583    2.895833     4.973958   1.7031250

Clustering vector:
  [1] 1 2 2 2 1 1 1 1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 2 1 1 1 2 1 1 1 2 1 1 2 2
 [44] 1 1 2 1 2 1 1 3 3 3 3 3 3 3 2 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [87] 3 3 3 3 3 3 3 2 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[130] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

Within cluster sum of squares by cluster:
[1]   6.432121  17.669524 118.651875
 (between_SS / total_SS =  79.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

We see how ***kmean$clusters*** has divided the observations into three clusters now and ***kmean$centers*** has also updated the centroid values as well. The plot below shows the grouping based on 3 clusters.

> autoplot (kmean3, data, frame=TRUE)



K-means is an efficient Machine Learning technique that:

- is easy to implement and apply
- has great interpretability
- produces tighter clusters than Hierarchical clustering
- is computationally fast

n-gl.com