***k*-Means: A Centroid-Based Technique**

The *k*-means algorithm defines the centroid of a cluster as the mean value of the points within the cluster. It proceeds as follows. First,it randomly selects *k* of the objects in *D*, each of which initially represents a cluster meanor center. For each of the remaining objects, an object is assigned to the cluster to whichit is the most similar, based on the Euclidean distance between the object and the clustermean.

The *k*-means algorithm then iteratively improves the within-cluster variation.

For each cluster, it computes the new mean using the objects assigned to the cluster in

the previous iteration. All the objects are then reassigned using the updated means as

the new cluster centers. The iterations continue until the assignment is stable, that is,

the clusters formed in the current round are the same as those formed in the previous

round.

**Algorithm: *k***-**means.** The *k*-means algorithm for partitioning, where each cluster’s center

is represented by the mean value of the objects in the cluster.

**Input:**

*k*: the number of clusters,

*D*: a data set containing *n* objects.

**Output:** A set of *k* clusters.

**Method:**

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;

(2) **repeat**

(3) (re)assign each object to the cluster to which the object is the most similar,

based on the mean value of the objects in the cluster;

(4) update the cluster means, that is, calculate the mean value of the objects for

each cluster;

(5) **until** no change;

**Program:**

kmean <- function(data,k){

data <- as.data.frame(data)

#vector to store cluster number for each object

clustring\_vector <- vector(length=nrow(data))

#selecting first k object

centroid <- data[sample(1:nrow(data),size=k),]

centroid\_Old <- data[sample(1:nrow(data),size=k),]

#repeat for iteration

repeat{

#if old cluster mean and new are same stop algo

if(all(centroid==centroid\_Old)){break()}

#loop for finding clustring\_vector for each object

for(i in 1:nrow(data)){

#vector to store distance from each centroid

distance <- c()

for(j in 1:nrow(centroid)){

distance <- c(distance,dist(rbind(data[i,],centroid[j,]),method="euclidean"))

}

#stroring minimum distance cluster index

clustring\_vector[i] <- order(distance)[1]

}

#loops for finding new cluster mean

centroid\_Old <- centroid

for(j in 1:k){

#temporary data frame to store object of particular cluster to find their mean

temp <- data.frame()

#vector fot storing mean for each column

mn <- c()

#loop for storing object of particular cluster in `temp`

for(i in 1:nrow(data)){

if(clustring\_vector[i] == j){

temp <- rbind(temp,data[i,])

}

}

#using apply

mn <- apply(temp[,1:ncol(temp)],2,mean)

#print(temp)

tdf <- data.frame()

tdf<- rbind(tdf,mn)

centroid[j,] <- tdf

}

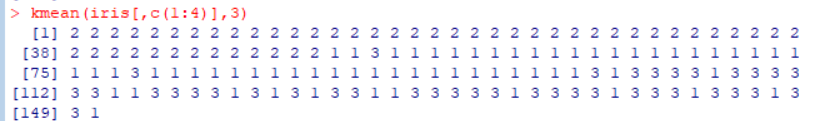
}

plot(data,col=clustring\_vector,main="K - Mean", pch=20, cex=2)

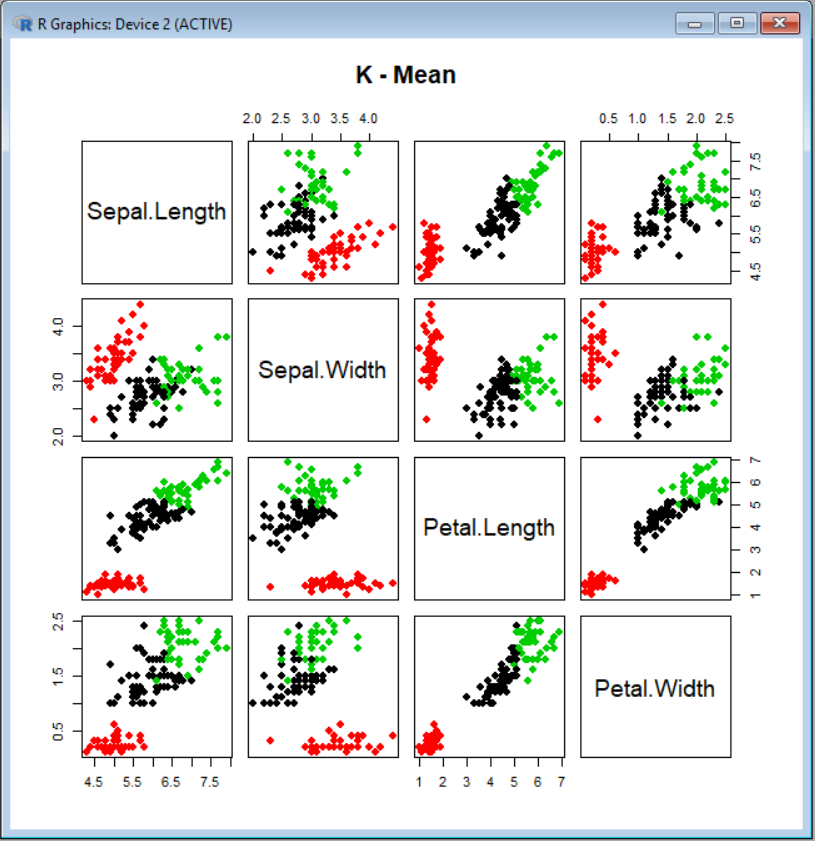
return (clustring\_vector)

}

**Output:**

****

**Output Graph:**

****

**K-Means Advantages :**1) If variables are huge, then  K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls.  
  
2) K-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.  
  
**K-Means Disadvantages :**1) Initially selected k-values should not belong to same cluster.  
2) K-mean only detect spherical cluster.  
3) K-mean does not detect noise or outliers.

**Complexity of k-mean:**

**Time complexity**: **O(kn)**

where k = number of cluster

n = number of data items

**Space complexity: O(n)**

Where n = number of data items