

# k Means Clustering

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

## Procedure

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
4. Compute the sum of the squared distance between data points and all centroids.
5. Assign each data point to the closest cluster (centroid).
6. Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

## Program Execution

1. Load the iris dataset and view the data.

```
#Load and view dataset
require("datasets")
data("iris") # load Iris Dataset
str(iris) #view structure of dataset
```

```
## 'data.frame':   150 obs. of  5 variables:
##  $ Sepal.Length: num   5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num   3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num   1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num   0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

2. Display the Statistical Summary of the dataset

```
summary(iris) #view statistical summary of dataset
```

```
##      Sepal.Length      Sepal.Width      Petal.Length      Petal.Width
##  Min.   :4.300    Min.   :2.000    Min.   :1.000    Min.   :0.100
##  1st Qu.:5.100    1st Qu.:2.800    1st Qu.:1.600    1st Qu.:0.300
##  Median :5.800    Median :3.000    Median :4.350    Median :1.300
##  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
##      Species
##  setosa   :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
head(iris) #view top rows of dataset
```

```
##      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
```

3. Apply the preprocessing to remove the class attribute eg., Species , since clustering is a type of unsupervised learning Preprocess the dataset Since clustering is a type of Unsupervised Learning, we would not require Class Label(output) during #execution of our algorithm. We will, therefore, remove Class Attribute Species and store it in another variable. We would then normalize the attributes

between 0 and 1 using our own function.

```
iris.new<- iris[,c(1,2,3,4)]
iris.class<- iris[, "Species"]
head(iris.new)
```

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

```
head(iris.class)
```

```
## [1] setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

#### 4. Create a function to normalize the data before clustering Normalization

```
normalize <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}
iris.new$Sepal.Length<- normalize(iris.new$Sepal.Length)
iris.new$Sepal.Width<- normalize(iris.new$Sepal.Width)
iris.new$Petal.Length<- normalize(iris.new$Petal.Length)
iris.new$Petal.Width<- normalize(iris.new$Petal.Width)
head(iris.new)
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  0.22222222  0.6250000  0.06779661  0.04166667
## 2  0.16666667  0.4166667  0.06779661  0.04166667
## 3  0.11111111  0.5000000  0.05084746  0.04166667
## 4  0.08333333  0.4583333  0.08474576  0.04166667
## 5  0.19444444  0.6666667  0.06779661  0.04166667
## 6  0.30555556  0.7916667  0.11864407  0.12500000
```

#### 5. Apply k-means clustering algorithm with k = 3

```
result<- kmeans(iris.new,3) #apply k-means algorithm with no. of centroids(k)=3
```

#### 6. Find the number of records in each cluster

```
result$size # gives no. of records in each cluster
```

```
## [1] 61 39 50
```

#### 7. Display the cluster center data point values

```
result$centers # gives value of cluster center datapoint value(3 centers for k=3)
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  0.4412568  0.3073770  0.57571548  0.54918033
## 2  0.7072650  0.4508547  0.79704476  0.82478632
## 3  0.1961111  0.5950000  0.07830508  0.06083333
```

#### 8. Display the cluster vector showing the cluster where each record falls

```
result$cluster #gives cluster vector showing the cluster where each record falls
```

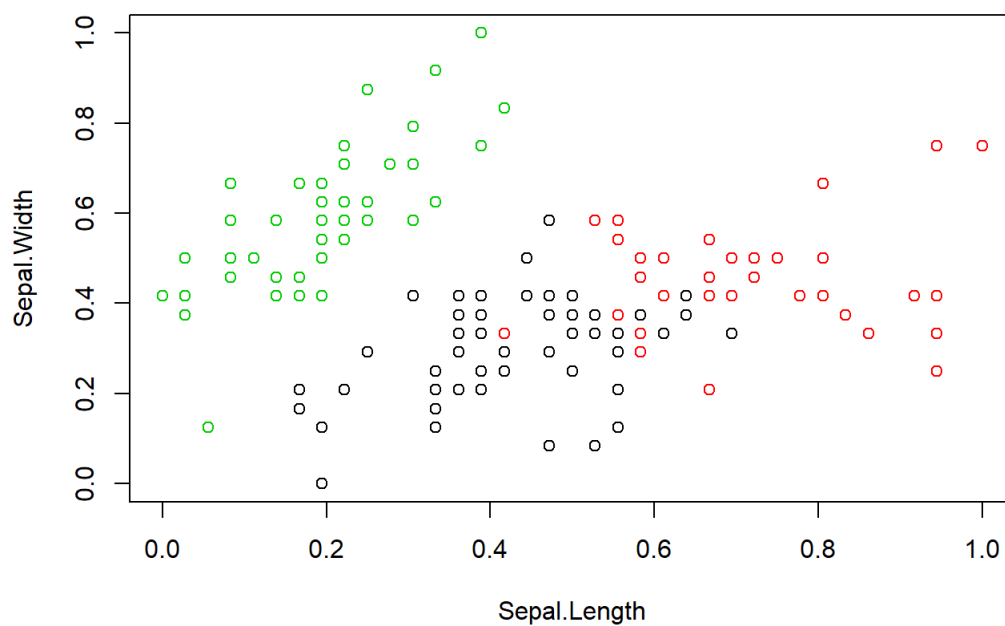
```
##      [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##     [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##     [75] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2
##    [112] 2 2 1 2 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 1 2 2 2 1 2 2 2 1 2
##    [149] 2 1
```

Verify results of clustering

```
par(mfrow=c(2,2), mar=c(5,4,2,2))
```

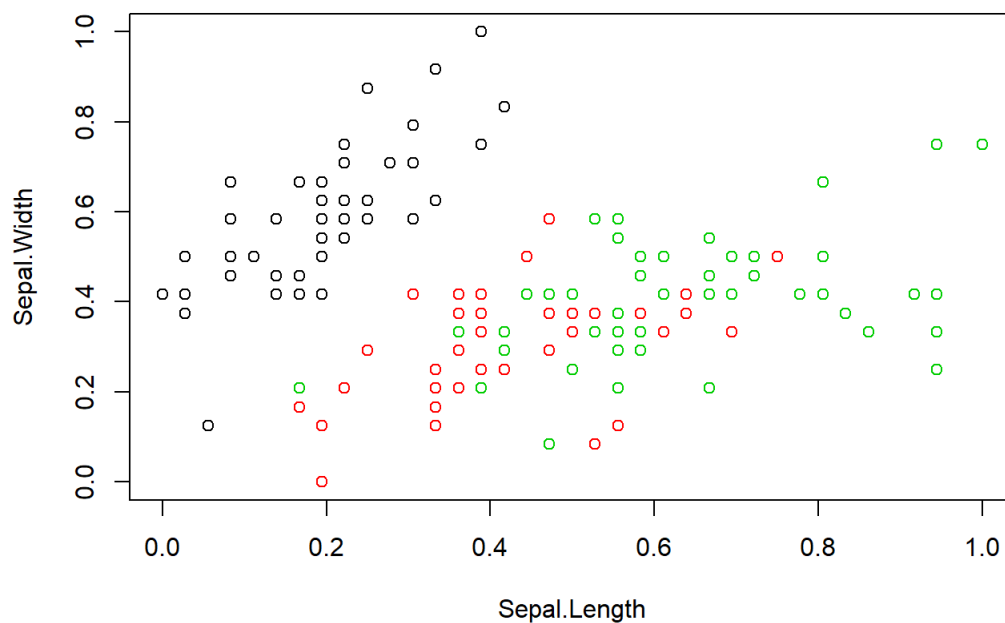
9. Plot to see how Sepal.Length and Sepal.Width data points have been distributed in clusters

```
plot(iris.new[c(1,2)], col=result$cluster)# Plot to see how Sepal.Length and Sepal.Width data points have been distributed in clusters
```



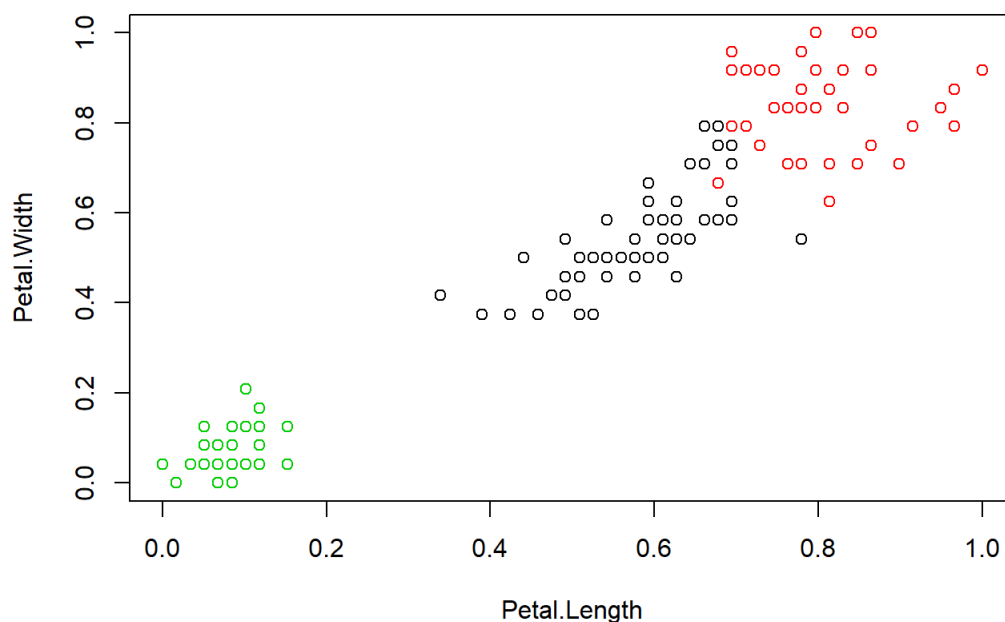
10. Plot to see how Sepal.Length and Sepal.Width data points have been distributed originally as per "class" attribute in dataset

```
plot(iris.new[c(1,2)], col=iris.class)# Plot to see how Sepal.Length and Sepal.Width data points have been distributed originally as per "class" attribute in dataset
```



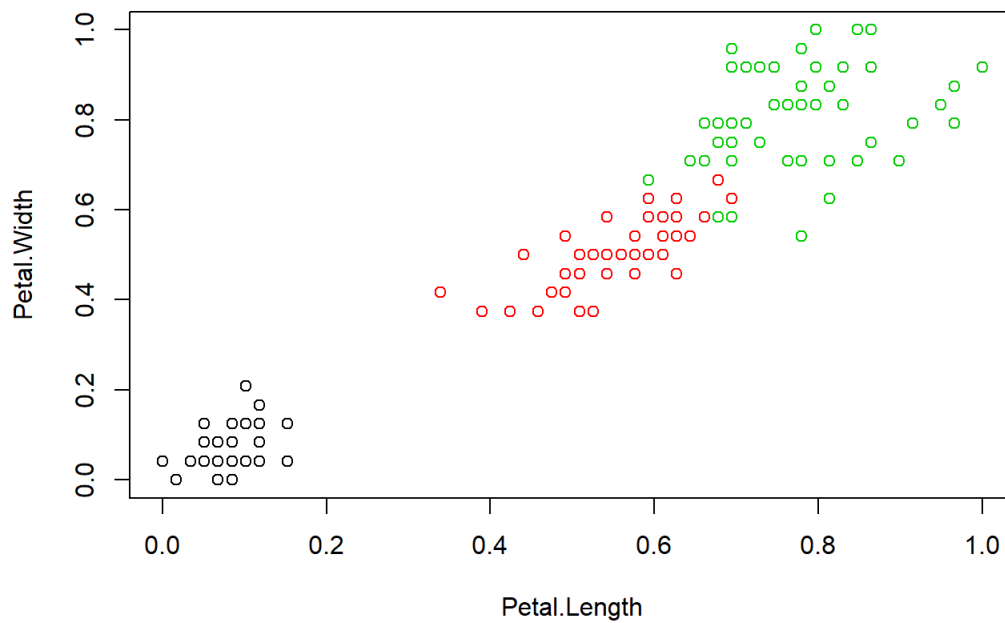
11. Plot to see how Petal.Length and Petal.Width data points have been distributed in clusters

```
plot(iris.new[c(3,4)], col=result$cluster) # Plot to see how Petal.Length and Petal.Width data points have been distributed in clusters
```



12. Plot to see how Petal.Length and Petal.Width data points have been distributed originally as per “class” attribute in dataset

```
plot(iris.new[c(3,4)], col=iris.class)
```



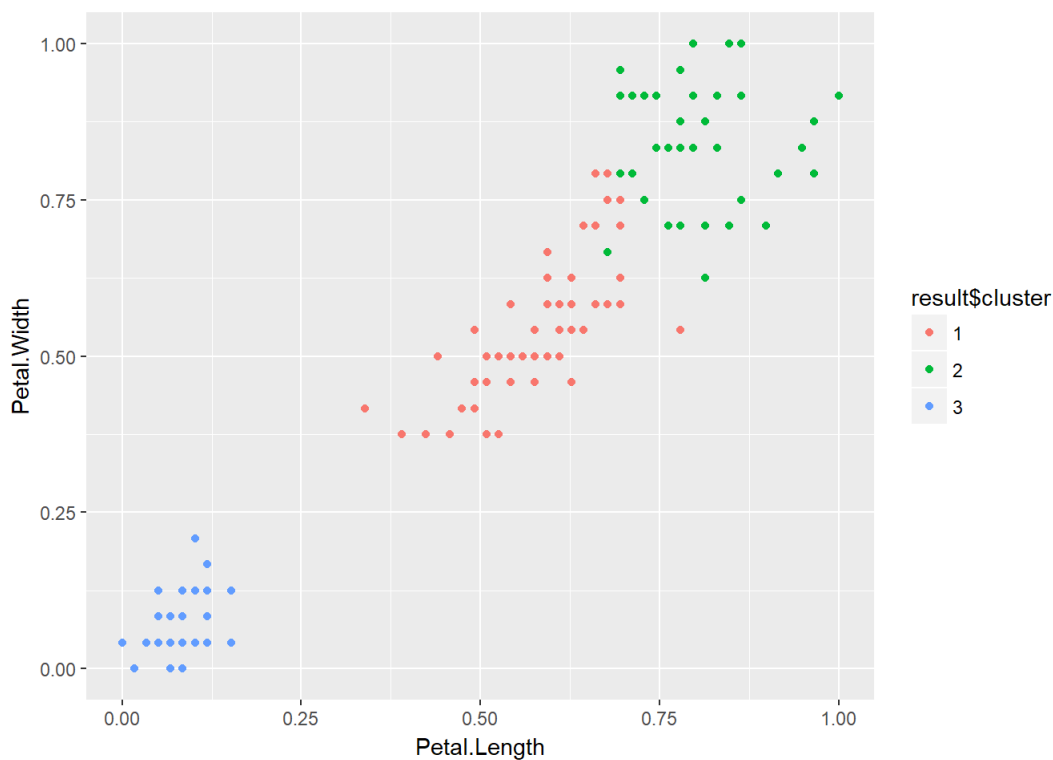
```
result$cluster <- as.factor(result$cluster)
```

13. Install the package ggplot2 and import it and Plot the cluster results using ggplot

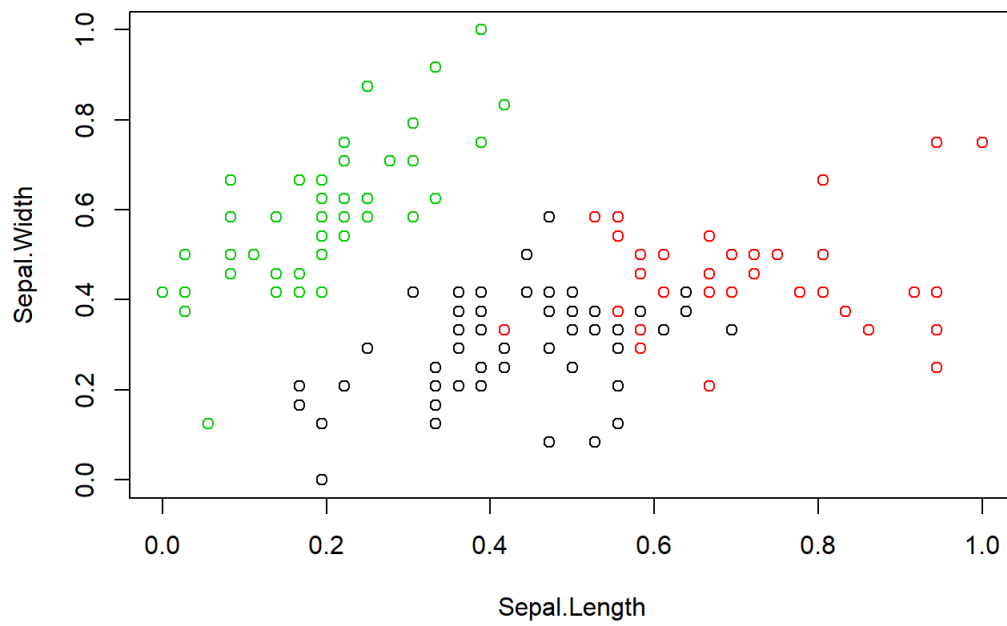
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.3
```

```
ggplot(iris.new, aes(Petal.Length, Petal.Width, color = result$cluster)) + geom_point()
```

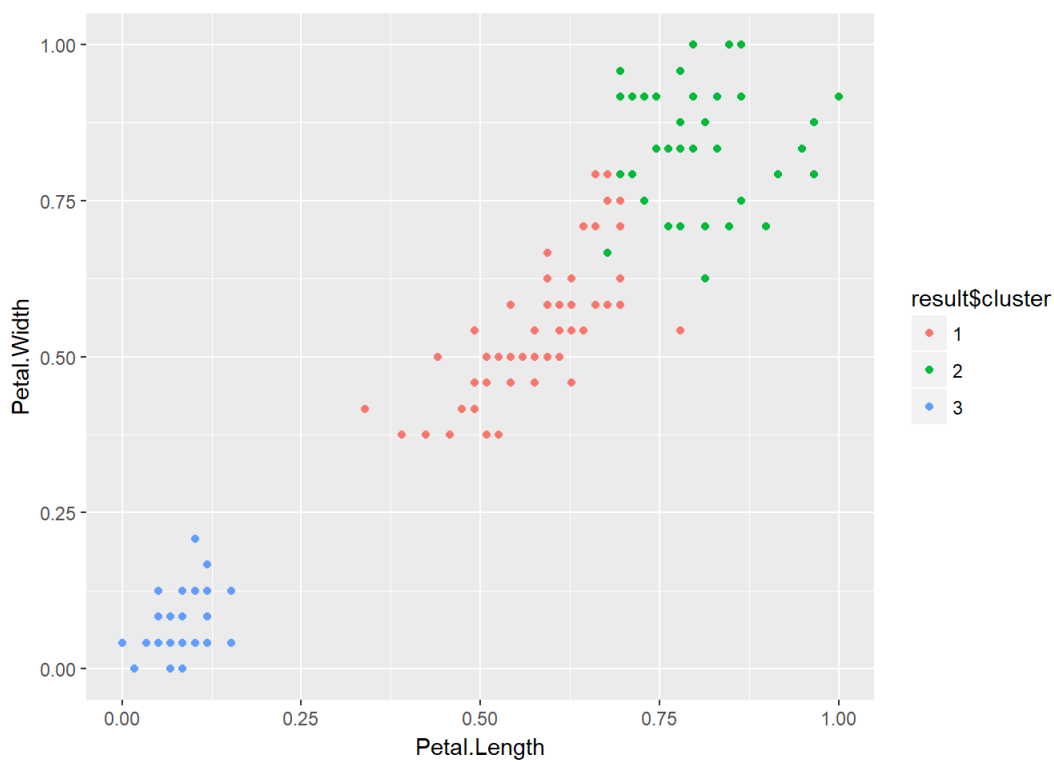


```
plot(iris.new[c("Sepal.Length", "Sepal.Width")], col=result$cluster)
```

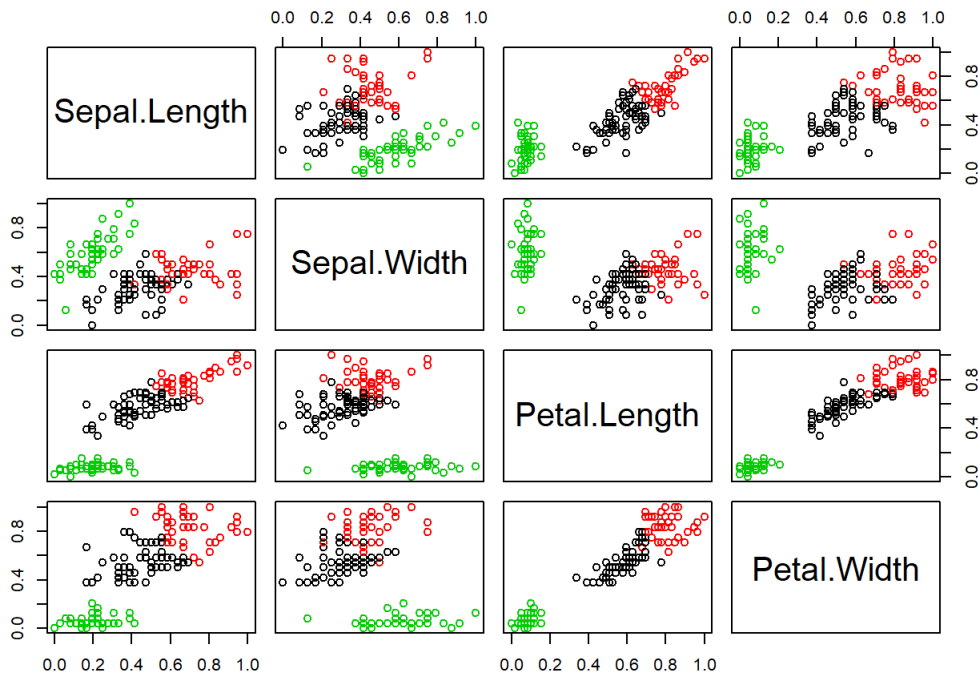


14. Display the clustering results with all parameters

```
ggplot(iris.new, aes(Petal.Length, Petal.Width, color = result$cluster)) + geom_point()
```



```
plot(iris.new[,], col=result$cluster)
```



#### 15. Display the results in table

```
table(result$cluster,iris.class) # Result of table shows that Cluster 1 corresponds to Virginica, Cluster 2
corresponds to Versicolor and Cluster 3 to Setosa.
```

```
##      iris.class
##      setosa versicolor virginica
## 1         0          47         14
## 2         0           3          36
## 3        50           0           0
```

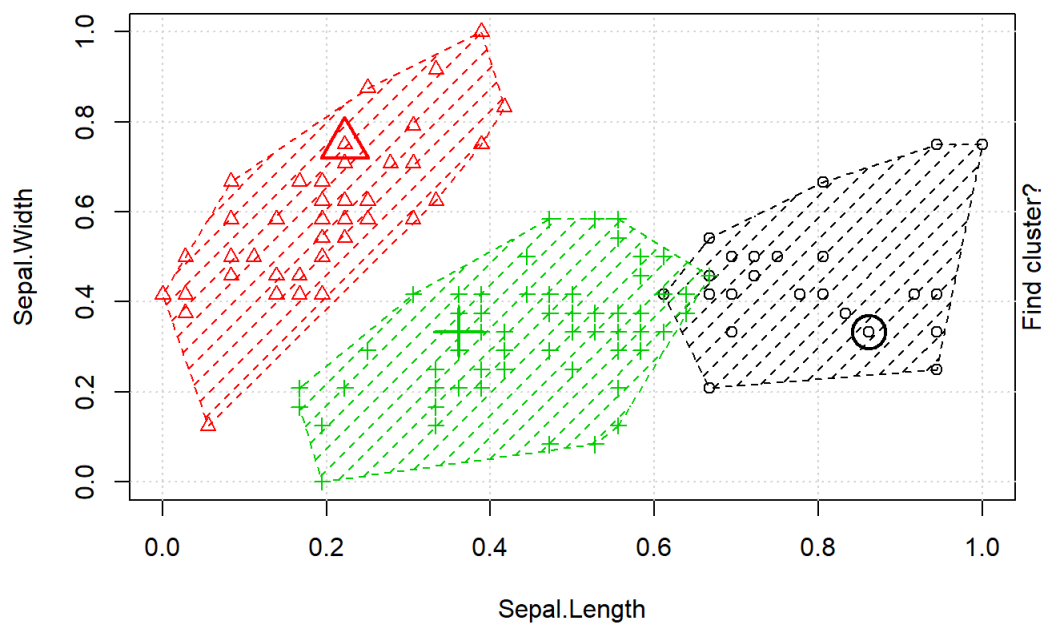
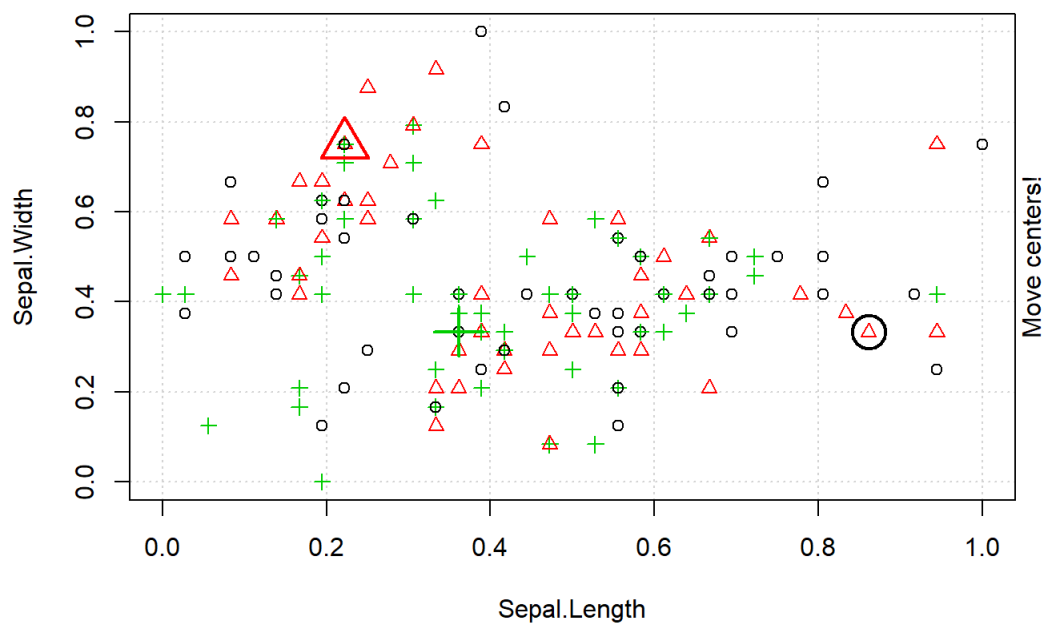
#### Inference

Total number of correctly classified instances are:  $36 + 47 + 50 = 133$  Total number of incorrectly classified instances are:  $3 + 14 = 17$   
 Accuracy =  $133 / (133 + 17) = 0.88$  i.e our model has achieved 88% accuracy! In order to improve this accuracy further, we may try different values of k

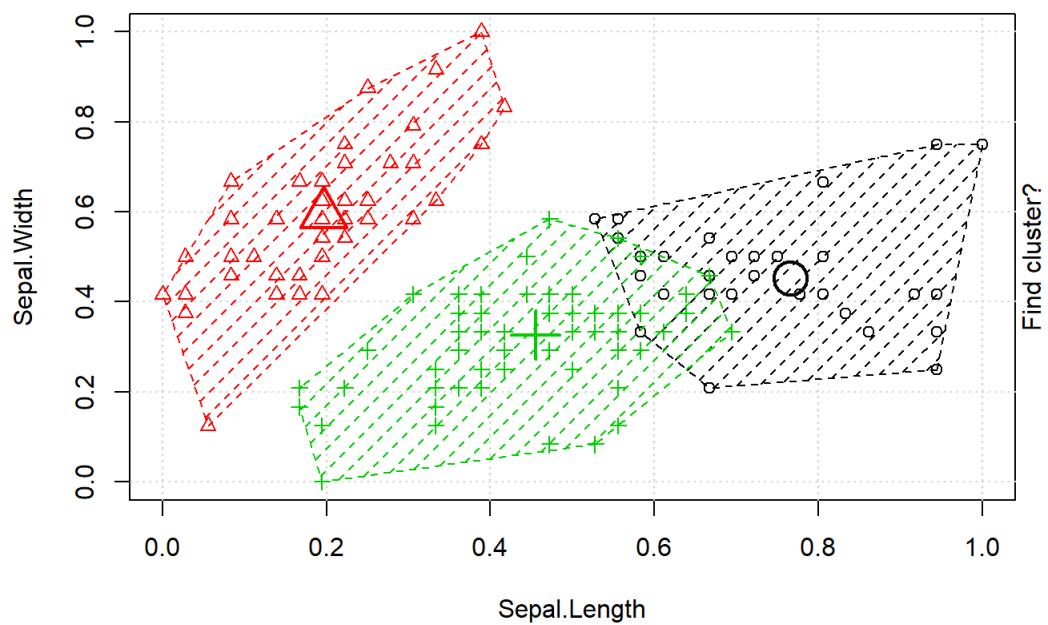
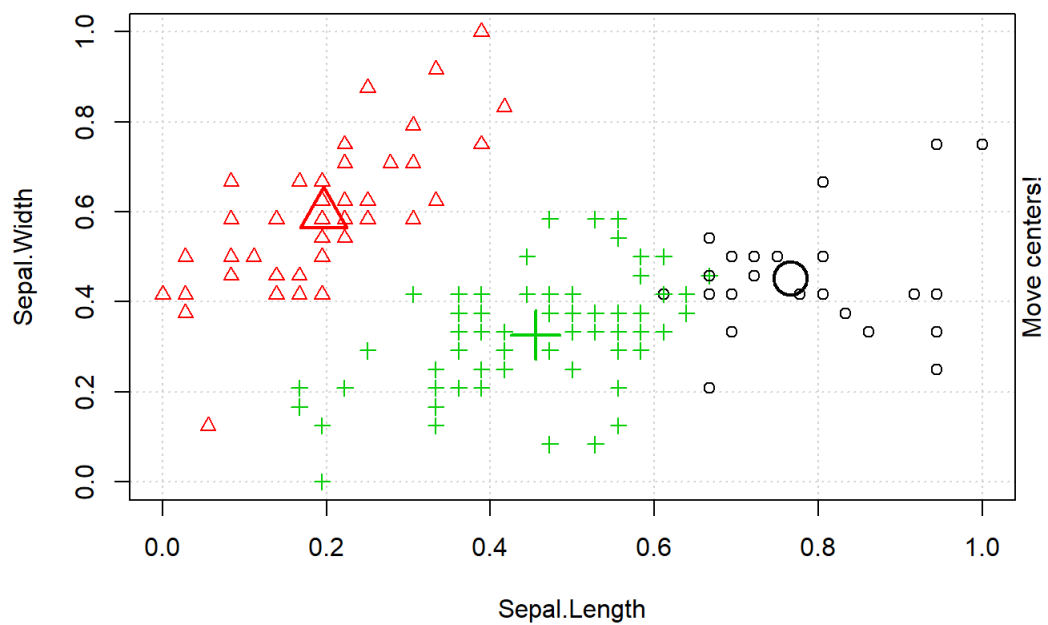
===== K means algorithms with Animation  
 =====

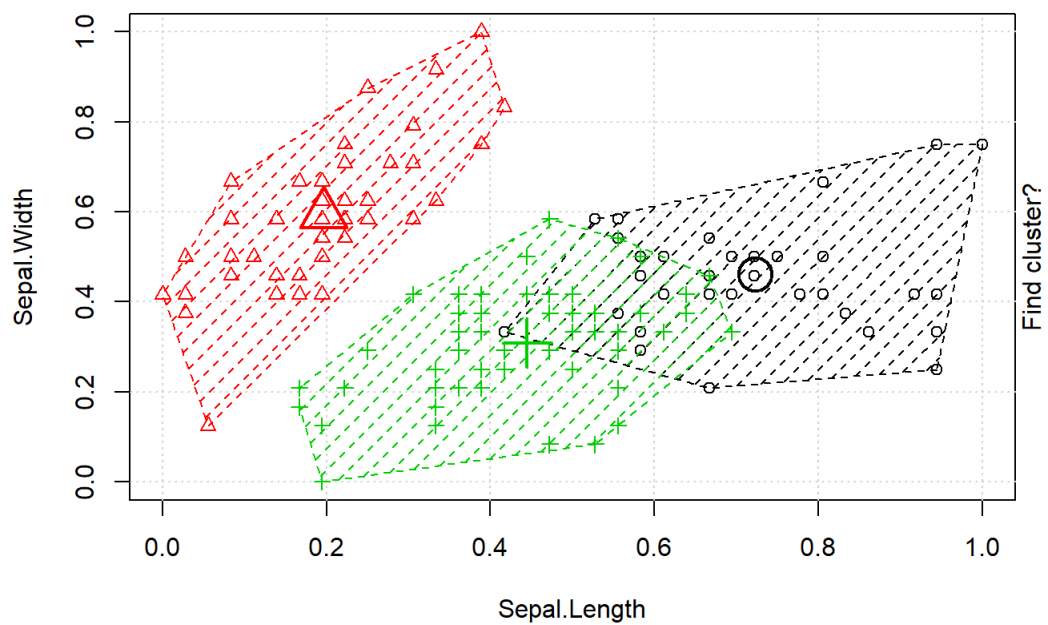
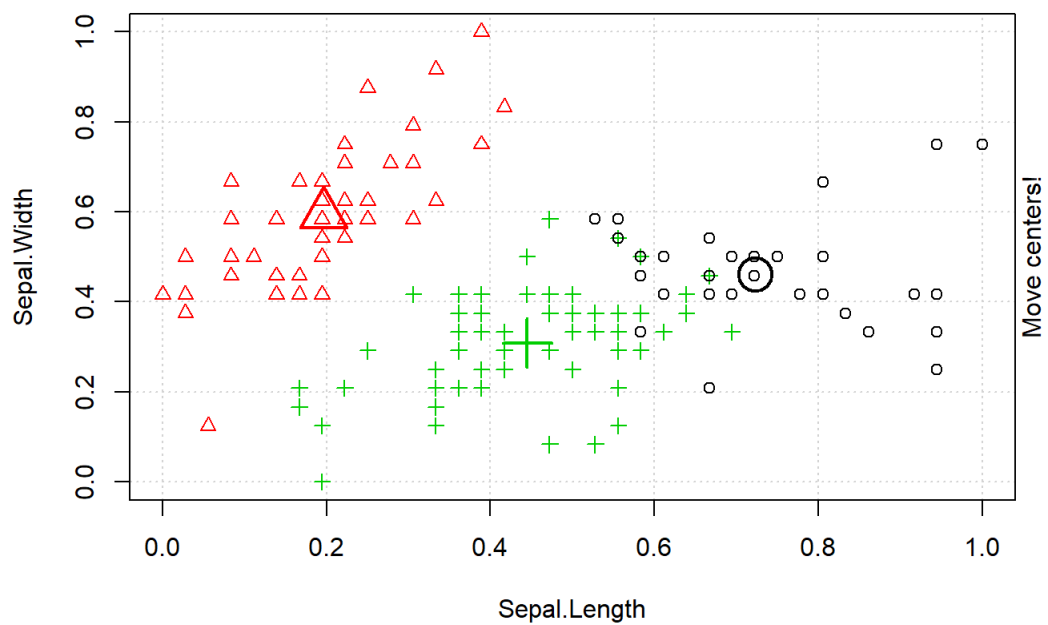
#### 16. Display the K Means Algorithm with Animation and visualize the changes in the cluster center

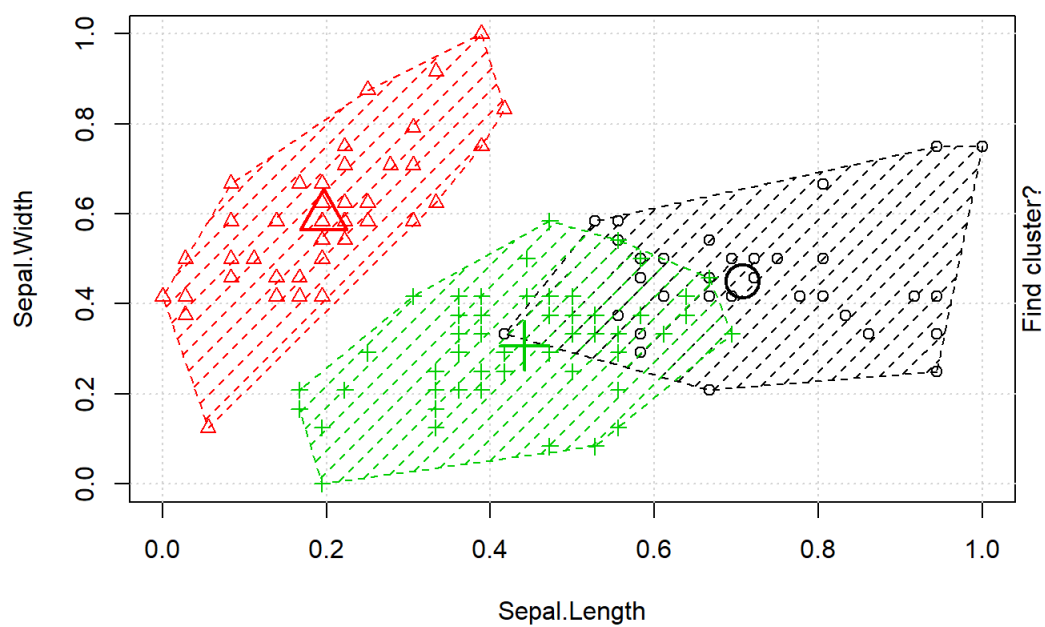
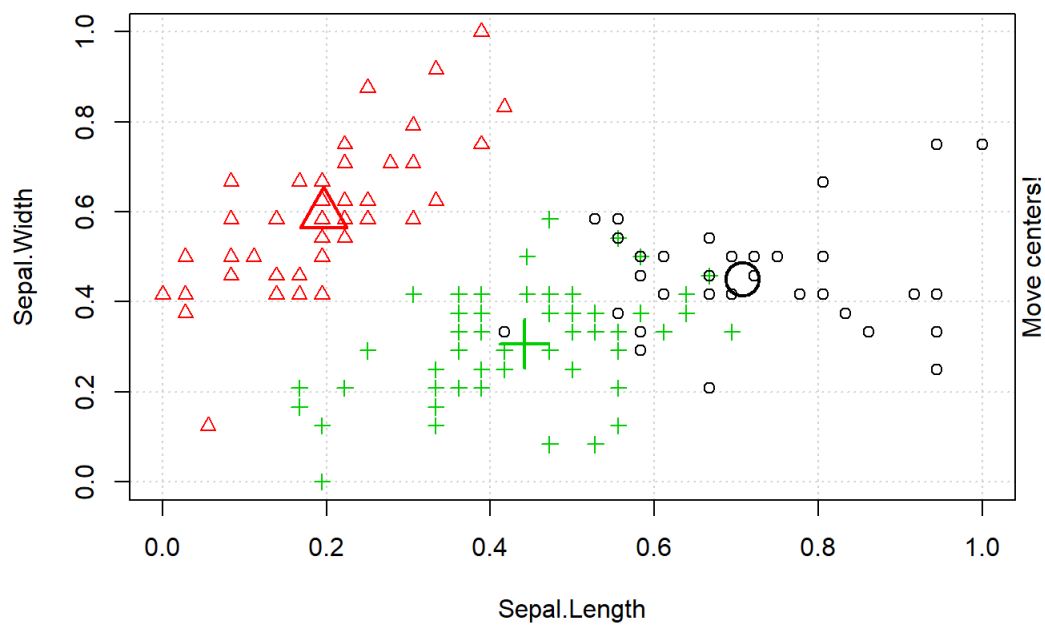
```
library(animation)
kml<-kmeans.ani(iris.new,3)
```









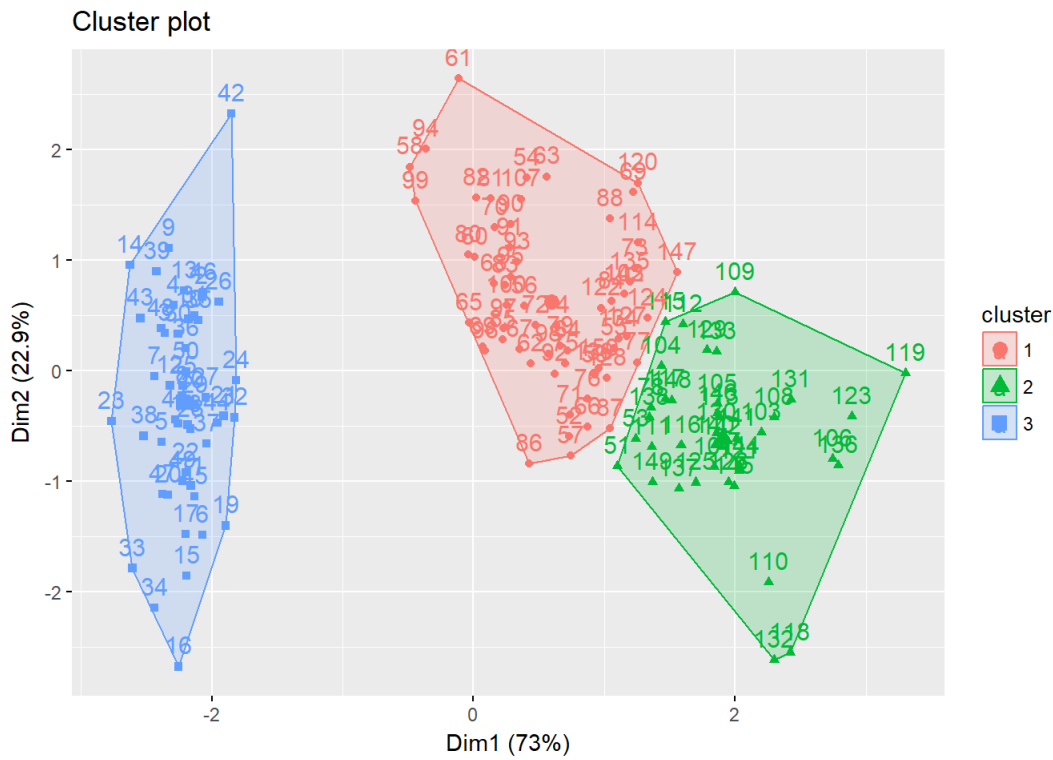


17. Import factoextra package and visualize the cluster result

```
library(factoextra) # clustering algorithms & visualization
```

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

```
fviz_cluster(result, data = iris.new)
```



18. Explore the cluster analysis result with various value of k like 3,4,5

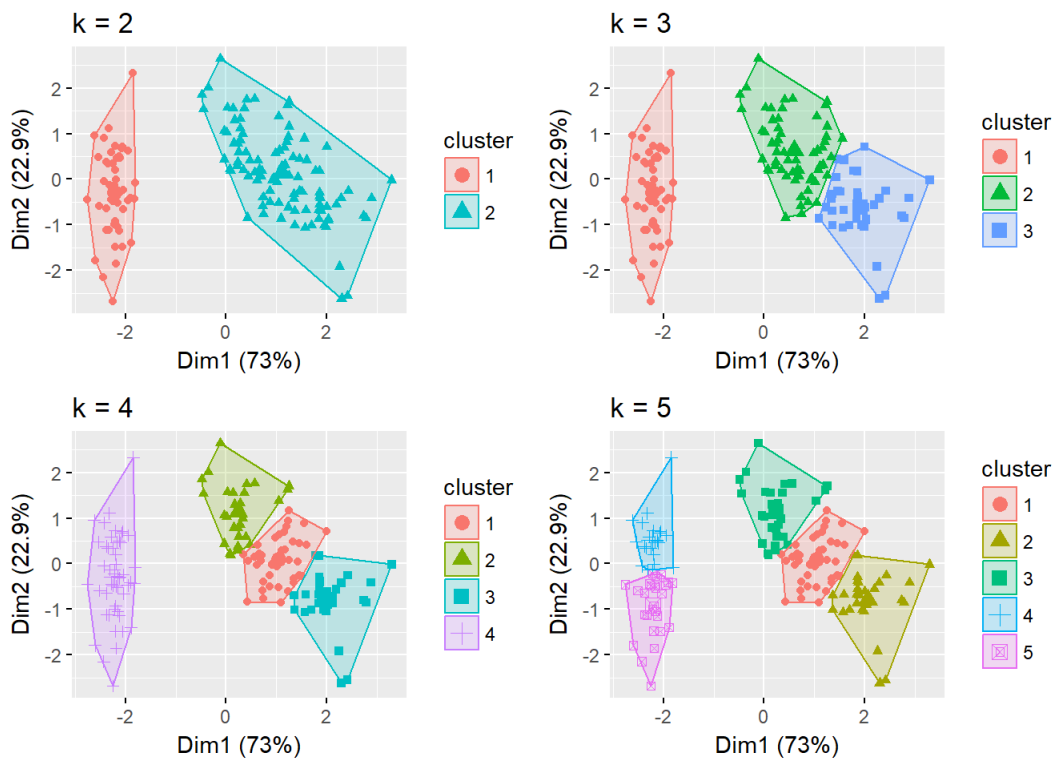
```
k2 <- kmeans(iris.new, centers = 2, nstart = 25)
k3 <- kmeans(iris.new, centers = 3, nstart = 25)
k4 <- kmeans(iris.new, centers = 4, nstart = 25)
k5 <- kmeans(iris.new, centers = 5, nstart = 25)
```

plots the results

```
p1 <- fviz_cluster(k2, geom = "point", data = iris.new) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point", data = iris.new) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point", data = iris.new) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point", data = iris.new) + ggtitle("k = 5")
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.3.3
```

```
grid.arrange(p1, p2, p3, p4, nrow = 2)
```



#### Conclusion

K means clustering algorithm with iris dataset is executed and visualized using various r libraries like ggplot,animation,gridExtra,factoextra