

## ASSIGNMENT NO.3:

**Aim** – Implement K-means using R programming.

### **Theory:**

K-Means Clustering is an unsupervised learning algorithm that is used to solve clustering problems in machine learning or data science. In this topic, we will learn what the K-means clustering algorithm, how it works, along with the Python implementation of k-means clustering.

**K-Means Algorithm:** K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of predefined clusters that need to be created in the process, as if  $K=2$ , there will be two clusters, and for  $K=3$ , there will be three clusters, and so on. It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs to only one group that has similar properties. The k-means clustering algorithm mainly performs two tasks:

Determines the best value for K center points or centroids by an iterative process. Assigns each data point to its closest k-center. Those data points which are near the particular k-center, create a cluster. Hence each cluster has data points with some commonalities, and it is away from other clusters. K-Means algorithm:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other than the input dataset).

Step-3: Assign each data point to its closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster. Step-5: Repeat the third steps, which means reassigning each data point to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step 4 else go to FINISH.

Step-7: The model is ready.

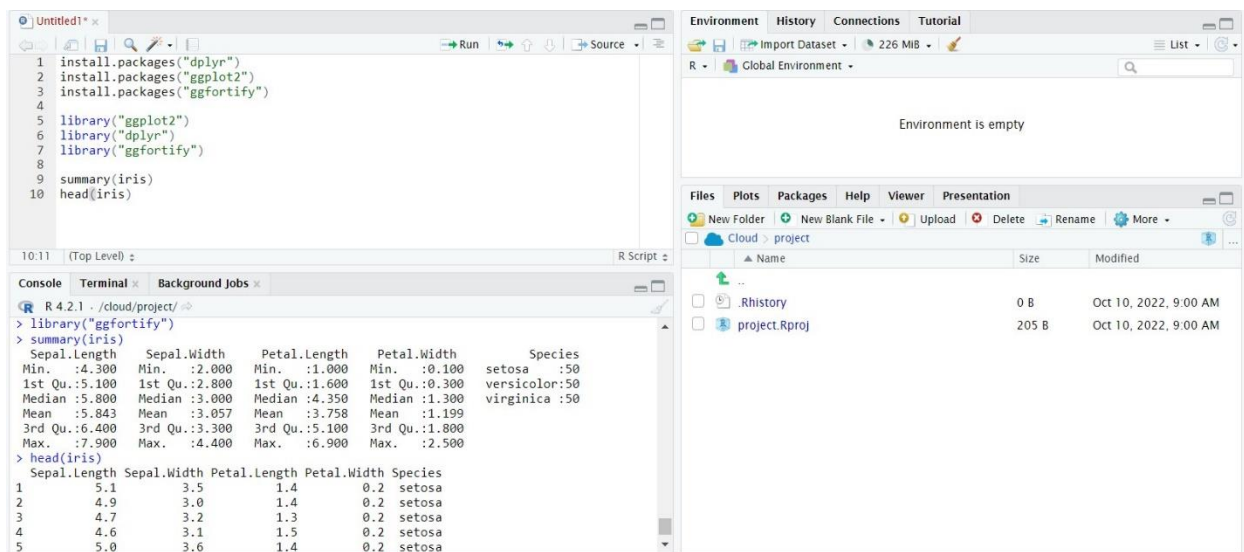
**Elbow Method:** In cluster analysis, the elbow method is a heuristic used in

determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use. Using the "elbow" or "knee of a curve" as a cutoff point is a common heuristic in mathematical optimization to choose a point where diminishing returns are no longer worth the additional cost. In clustering, this means one should choose a number of clusters so that adding another cluster doesn't give a much better modeling of the data.

The Elbow method looks at the total WSS as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't improve much better the total WSS. The optimal number of clusters can be defined Calculate the Within-Cluster-Sum of Squared Errors (WSS) for different values of k, and choose the k for which WSS first starts to diminish. In the plot of WSS-versus-k, this is visible as an elbow. Within-Cluster-Sum of Squared Errors sounds a bit complex. Let's break it down: The Squared Error for each point is the square of the distance of the point from its representation i.e. its predicted cluster center.

## R Program CODE:

### Reading & Summary of data



The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains R code to install and load packages, and to summarize and view the iris dataset.
- Console:** Shows the execution of the code, including the output of `summary(iris)` and `head(iris)`.
- Environment Pane:** Shows the Global Environment with the `iris` dataset loaded.
- Files Pane:** Shows the project structure with files like `Rhistory` and `project.Rproj`.

```
1 install.packages("dplyr")
2 install.packages("ggplot2")
3 install.packages("ggfortify")
4
5 library("ggplot2")
6 library("dplyr")
7 library("ggfortify")
8
9 summary(iris)
10 head(iris)
```

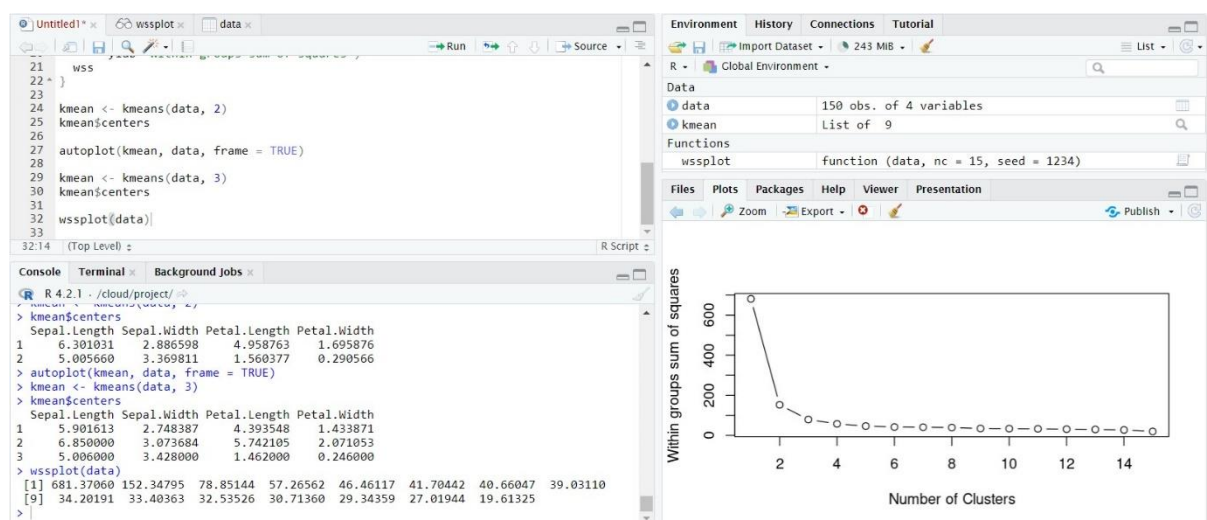
Console Output:

```
R 4.2.1 - /cloud/project/
> library("ggfortify")
> 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
```

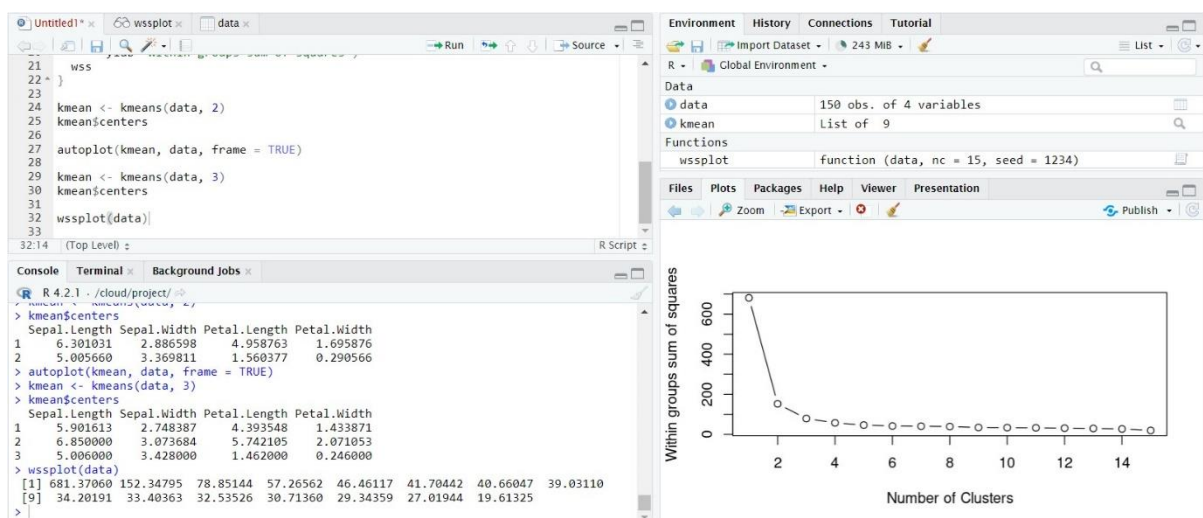
```
> summary(iris)
      Id      Sepal.LengthCm  Sepal.WidthCm  Petal.LengthCm  Petal.WidthCm  Species
Min.   : 1.00      Min.   :4.300      Min.   :2.000      Min.   :1.000      Min.   :0.100      Length:150
1st Qu.: 38.25     1st Qu.:5.100     1st Qu.:2.800     1st Qu.:1.600     1st Qu.:0.300      Class :character
Median : 75.50     Median :5.800     Median :3.000     Median :4.350     Median :1.300      Mode  :character
Mean   : 75.50     Mean   :5.843     Mean   :3.054     Mean   :3.759     Mean   :1.199
3rd Qu.:112.75     3rd Qu.:6.400     3rd Qu.:3.300     3rd Qu.:5.100     3rd Qu.:1.800
Max.   :150.00     Max.   :7.900     Max.   :4.400     Max.   :6.900     Max.   :2.500
```

## Selecting the columns for clustering excluding the classes of dataset

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



## lets find the WCSS score to find the no of clusters



In the above plot WCSS can observe that if we use 2 cluster then it is best choice for us because after k=2 the WCSS is gradual.

## Implementing K-means

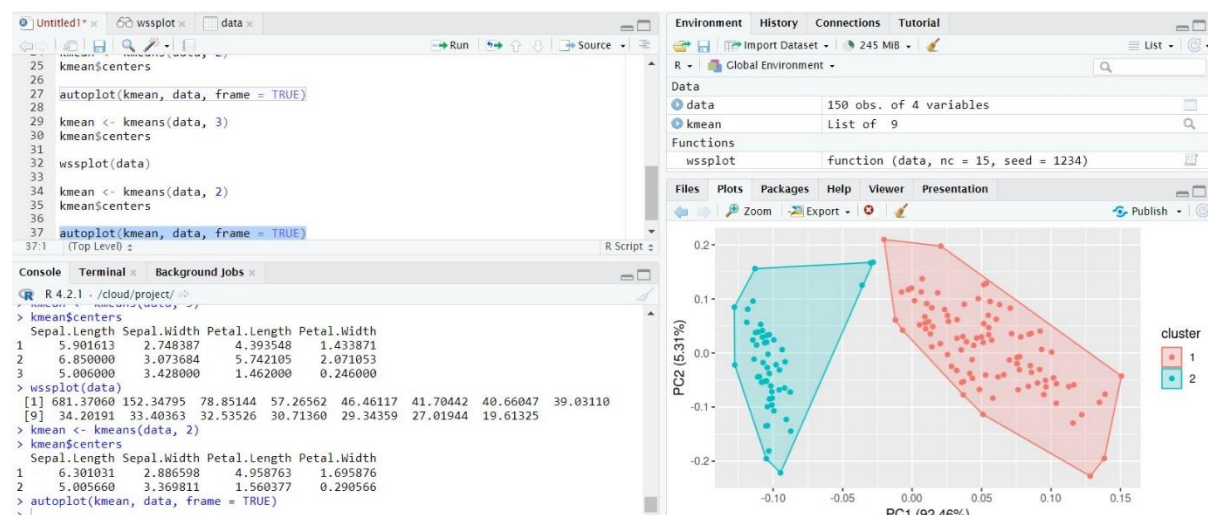
```
kmean <- kmeans(data, 2)
kmean$centers
```

```
23
24 kmean <- kmeans(data, 2)
25 kmean$centers
26
27 autoplot(kmean, data, frame = TRUE)
28
29 kmean <- kmeans(data, 3)
30 kmean$centers
31
32 wssplot(data)
33
34 kmean <- kmeans(data, 2)
35 kmean$centers
35:14 (Top Level) R Script
```

```
R 4.2.1 - /cloud/project/
> autoplot(kmean, data, frame = TRUE)
> kmean <- kmeans(data, 3)
> kmean$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1    5.901613    2.748387    4.393548    1.433871
2    6.850000    3.073684    5.742105    2.071053
3    5.006000    3.428000    1.462000    0.246000
> wssplot(data)
[1] 681.37060 152.34795 78.85144 57.26562 46.46117 41.70442 40.66047 39.03110
[9] 34.20191 33.40363 32.53526 30.71360 29.34359 27.01944 19.61325
> kmean <- kmeans(data, 2)
> kmean$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1    6.301031    2.886598    4.958763    1.695876
2    5.005660    3.369811    1.560377    0.290566
>
```

## Plotting our data-points in clusters

```
autoplot(kmean, data, frame = TRUE)
```



## Conclusion:

WSS helps to identify the K value