

Lab 8: Unsupervised Learning – Clustering

1. Implementation and analysis of Clustering algorithms like:

1. K-Means
2. Hierarchical Clustering - Agglomerative

Theory:

K-Means Clustering:

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined 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. It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters. The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

Working of 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 from the input dataset). **Step-3:**

Assign each data point to their closest centroid, which will form the predefined Kclusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which mean reassign each datapoint to the new closest centroid of each cluster.

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

7: The model is ready.

Hierarchical Clustering-Agglomerative:

In agglomerative clustering, Initially consider every data point as an individual Cluster and at every step, merge the nearest pairs of the cluster. (It is a bottom-up method). At first, every dataset is considered as an individual entity or cluster. At every iteration, the clusters merge with different clusters until one cluster is formed. we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram.

Hierarchical Clustering-Agglomerative Algorithm:

1. Calculate the similarity of one cluster with all the other clusters (calculate proximity matrix)
2. Consider every data point as an individual cluster
3. Merge the clusters which are highly similar or close to each other.
4. Recalculate the proximity matrix for each cluster
5. Repeat Steps 3 and 4 until only a single cluster remains.

1. Implementation of K-Means Clustering:

```
Source

Console Terminal Background Jobs
R 4.2.2 · D:/akashadms/

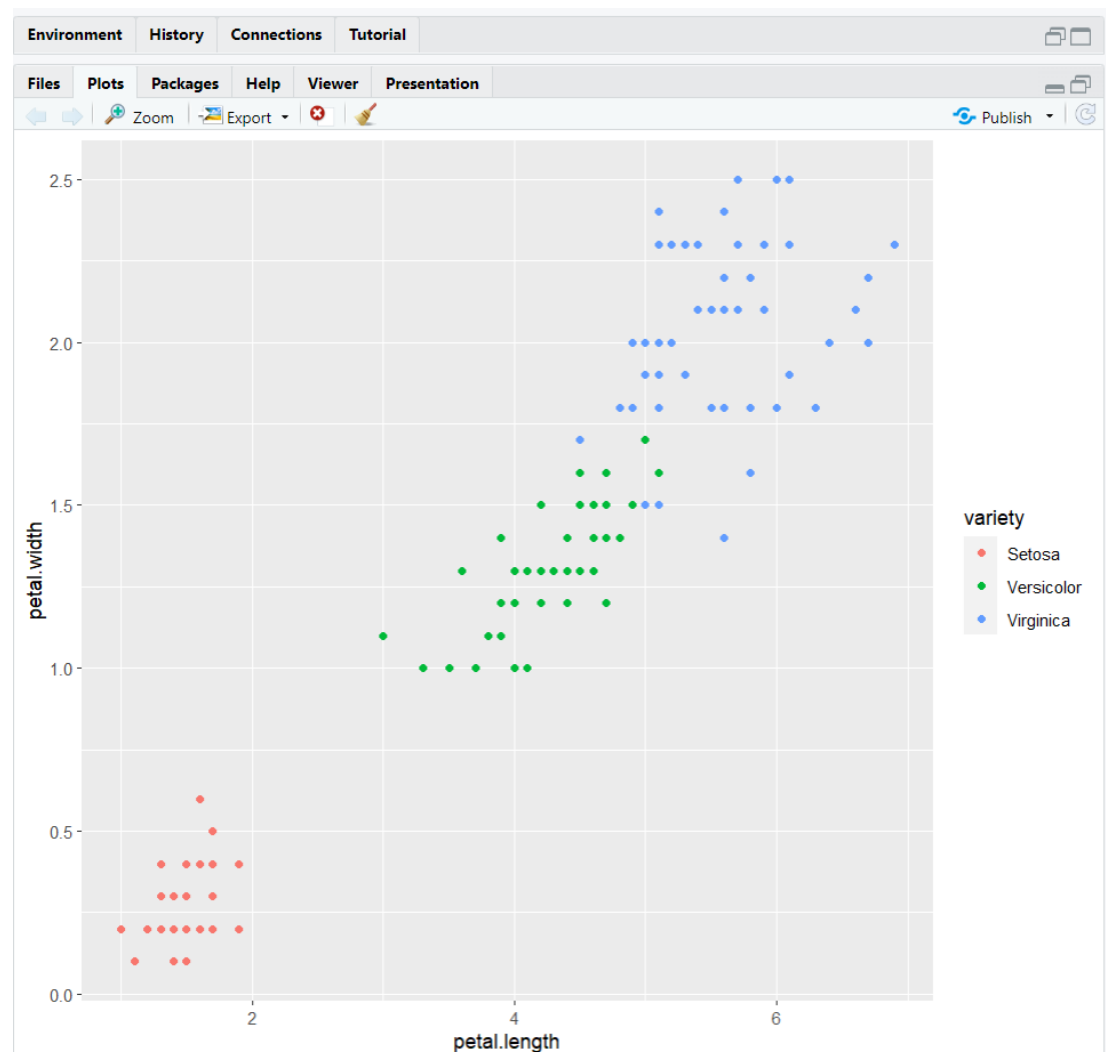
R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> setwd("D:/akashadms")
> getwd()
[1] "D:/akashadms"
> iris<-read.csv('D:/akashadms/iris.csv')
> library(ggplot2)
> ggplot(iris, aes(petal.length, petal.width, color = variety)) + geom_point()
> |
```



```
> iris <- iris[sample(nrow(iris), ), ]
> set.seed(20)
```

```

> irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)
> irisCluster
K-means clustering with 3 clusters of sizes 48, 50, 52

Cluster means:
  petal.length petal.width
1    5.595833    2.037500
2    1.462000    0.246000
3    4.269231    1.342308

Clustering vector:
24 124 8 16 33 15 142 31 18 59 105 98 46 74 97 56 53 29 47 80 55
2 1 2 2 2 2 1 2 2 3 1 3 2 3 3 3 3 2 2 3 3
72 126 99 81 78 52 132 90 62 109 34 135 32 22 73 61 21 13 51 125 100
3 1 3 3 1 3 1 3 3 1 2 1 2 2 3 3 2 2 3 1 3
26 150 6 11 28 43 111 128 145 122 148 19 91 138 2 35 141 14 123 12 30
2 1 2 2 2 2 1 1 1 1 1 2 3 1 2 2 1 2 1 2 2
40 144 107 3 114 119 118 37 106 133 149 25 136 143 69 48 7 85 113 131 41
2 1 3 2 1 1 1 2 1 1 1 2 1 1 3 2 2 3 1 1 2
58 44 4 57 96 102 42 36 27 147 112 101 139 110 9 108 50 117 95 70 79
3 2 2 3 3 1 2 2 2 1 1 1 3 1 2 1 2 1 3 3 3
137 68 82 84 121 38 140 92 54 10 129 94 76 89 134 120 104 39 23 20 5
1 3 3 1 1 2 1 3 3 2 1 3 3 3 1 3 1 2 2 2 2
49 130 63 83 88 116 87 77 86 127 17 66 115 67 60 146 64 93 1 45 103
2 1 3 3 3 1 3 3 3 3 2 3 1 3 3 1 3 3 2 2 1
75 71 65
3 3 3

Within cluster sum of squares by cluster:
[1] 16.29167 2.02200 13.05769
(between_SS / total_SS = 94.3 %)

Available components:

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

> table(irisCluster$cluster, iris$variety)

      Setosa Versicolor Virginica
1         0             2         46
2        50             0          0
3         0            48          4

~
> irisCluster$iter # total iteration
[1] 2

> irisCluster$centers # three centroid for three cluster
  petal.length petal.width
1    5.595833    2.037500
2    1.462000    0.246000
3    4.269231    1.342308

> irisCluster$size # no of record in each cluster
[1] 48 50 52

> irisCluster$ifault # fault or wrong interpreted record
[1] 0

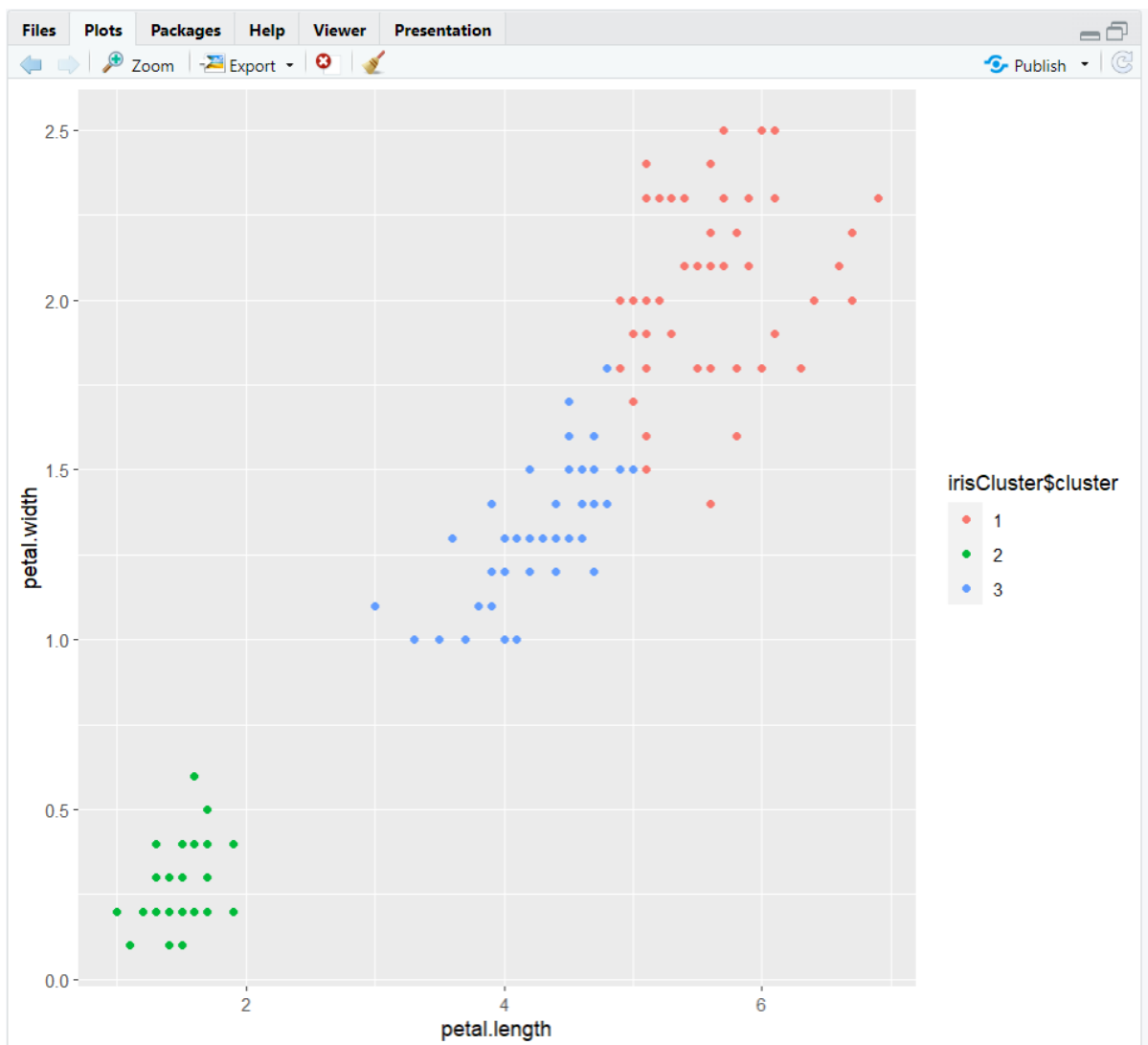
```

```
> irisCluster$cluster # final cluster assign to each record
```

```
24 124 8 16 33 15 142 31 18 59 105 98 46 74 97 56 53 29 47 80 55
2 1 2 2 2 2 1 2 2 3 1 3 2 3 3 3 3 2 2 3 3
72 126 99 81 78 52 132 90 62 109 34 135 32 22 73 61 21 13 51 125 100
3 1 3 3 1 3 1 3 3 1 2 1 2 2 3 3 2 2 3 1 3
26 150 6 11 28 43 111 128 145 122 148 19 91 138 2 35 141 14 123 12 30
2 1 2 2 2 2 1 1 1 1 1 2 3 1 2 2 1 2 1 2 2
40 144 107 3 114 119 118 37 106 133 149 25 136 143 69 48 7 85 113 131 41
2 1 3 2 1 1 1 2 1 1 1 2 1 1 3 2 2 3 1 1 2
58 44 4 57 96 102 42 36 27 147 112 101 139 110 9 108 50 117 95 70 79
3 2 2 3 3 1 2 2 2 1 1 1 3 1 2 1 2 1 3 3 3
137 68 82 84 121 38 140 92 54 10 129 94 76 89 134 120 104 39 23 20 5
1 3 3 1 1 2 1 3 3 2 1 3 3 3 1 3 1 2 2 2 2
49 130 63 83 88 116 87 77 86 127 17 66 115 67 60 146 64 93 1 45 103
2 1 3 3 3 1 3 3 3 3 2 3 1 3 3 1 3 3 2 2 1
75 71 65
3 3 3
```

```
> irisCluster$cluster<- as.factor(irisCluster$cluster)
```

```
> ggplot(iris, aes(petal.length, petal.width, color = irisCluster$cluster)) +geom_point()
```



2.Implementation of Hierarchical Clustering-Agglomerative

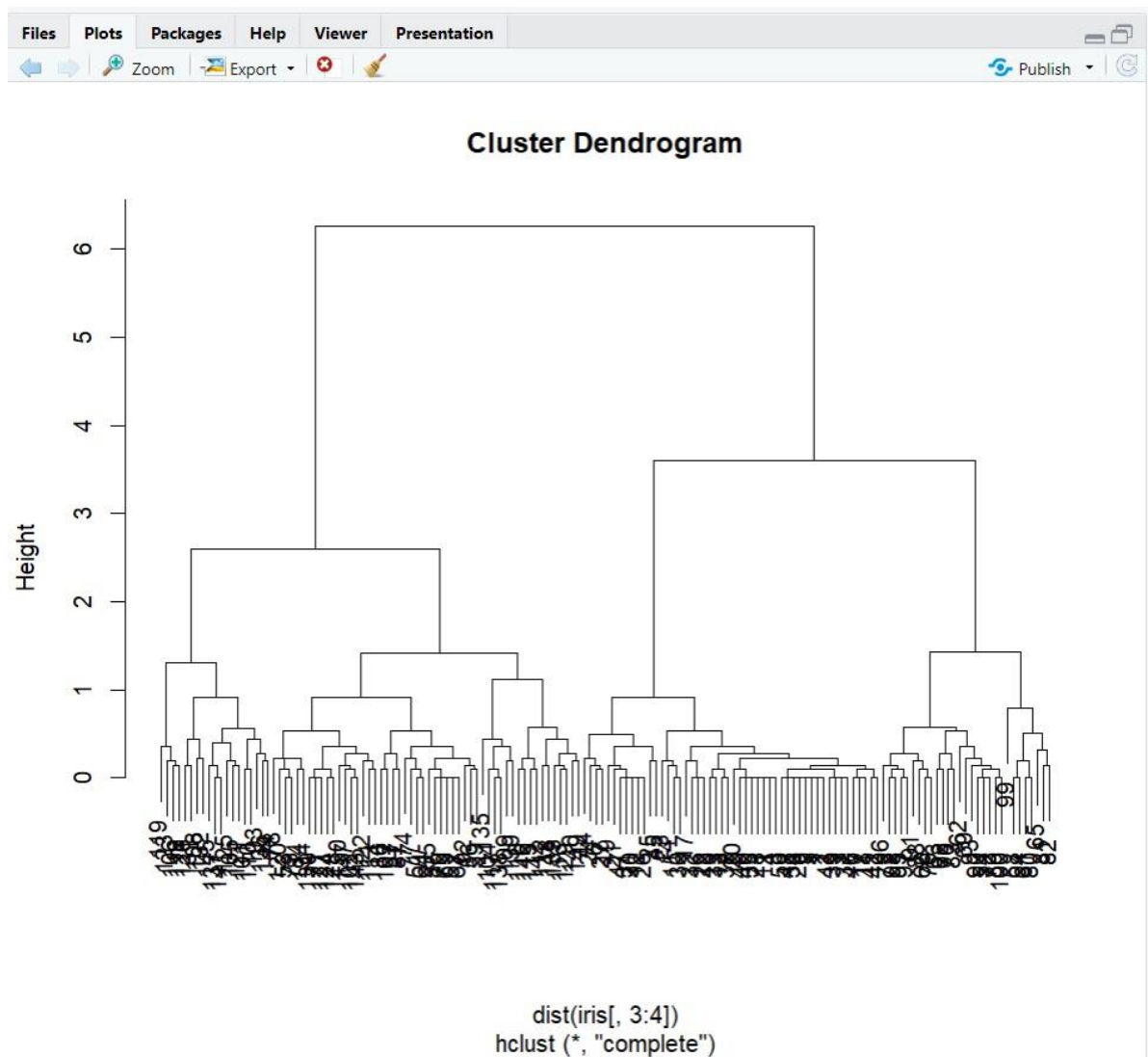
```
> iris<-read.csv('D:/akashadms/iris.csv')
> head(iris)
  sepal.length sepal.width petal.length petal.width variety
1          5.1         3.5         1.4         0.2   Setosa
2          4.9         3.0         1.4         0.2   Setosa
3          4.7         3.2         1.3         0.2   Setosa
4          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
```

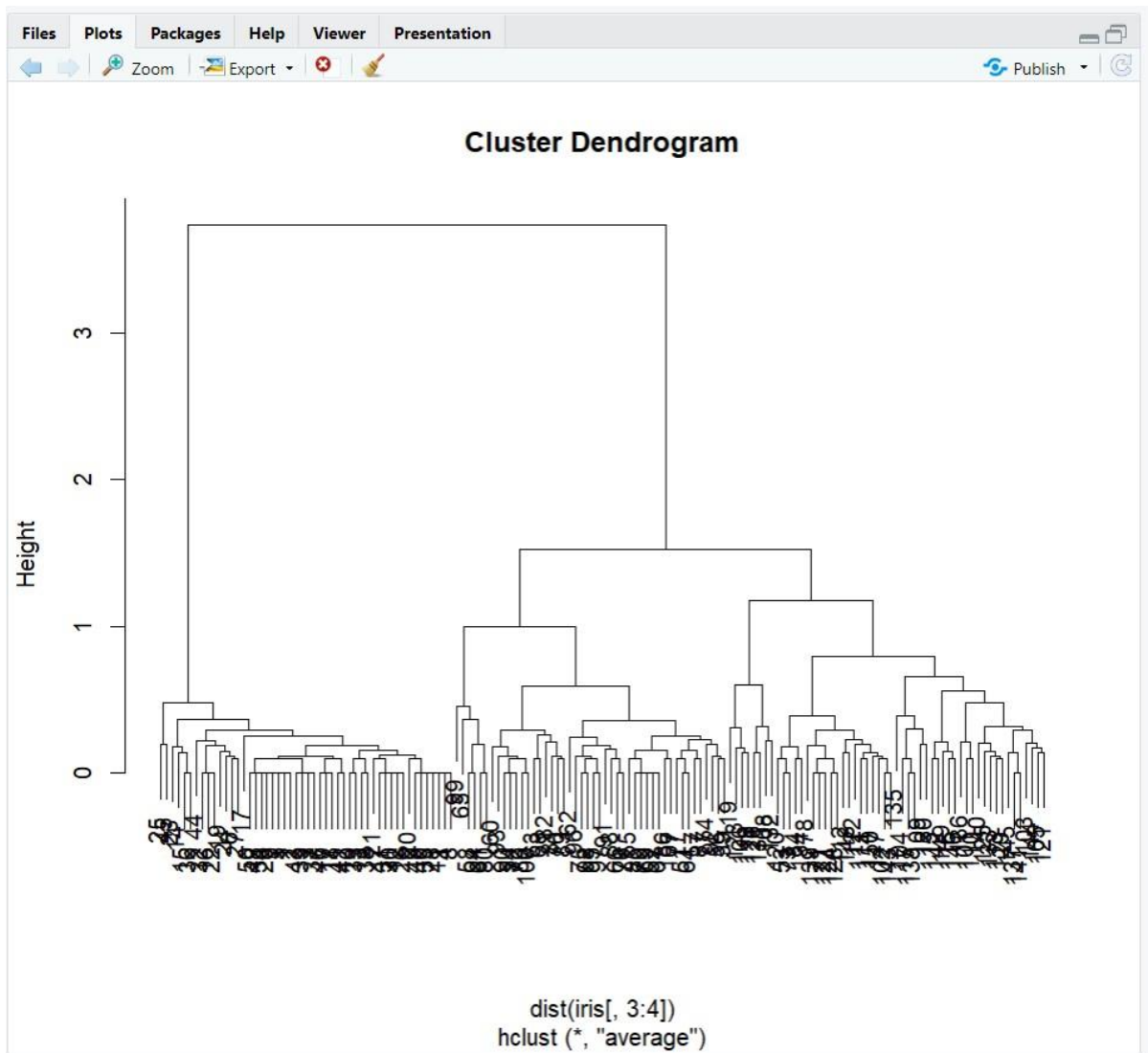
```
> clusters <- hclust(dist(iris[, 3:4]), method="complete")
> clusters
```

Call:
hclust(d = dist(iris[, 3:4]), method = "complete")

Cluster method : complete
Distance : euclidean
Number of objects: 150

```
> plot(clusters)
```





```
> clusterCut <- cutree(clusters, 3)
> table(clusterCut, iris$variety)

clusterCut Setosa Versicolor Virginica
1          50           0           0
2           0          45           1
3           0           5          49
```