Assignment No. 2

Title:

Consider a suitable dataset. For clustering of data instances in different groups, apply different clustering techniques (minimum 2). Visualize the clusters using suitable tool.

Problem Definition:

Visulize the Cluster using Suitable tool

Prerequisite:

- Basic concepts of ETL.
- Knowledge about R tool/ RapidMiner/Weka

Software Requirements:

- R tool/ RapidMiner/Weka

Hardware Requirement:

- Dual Core Processor PC, 2GB RAM, 500 GB HDD,

Learning Objectives:

Use R functions to create K-means Clustering models and hierarchical clustering models/ Study the use of RapidMiner and Weka for Clustering

Outcomes:

Visualize the effectiveness of the K-means Clustering algorithm and hierarchical clustering using graphic capabilities in R, RapidMiner and Weka

Theory Concepts:

Technique 1:K-means Clustering

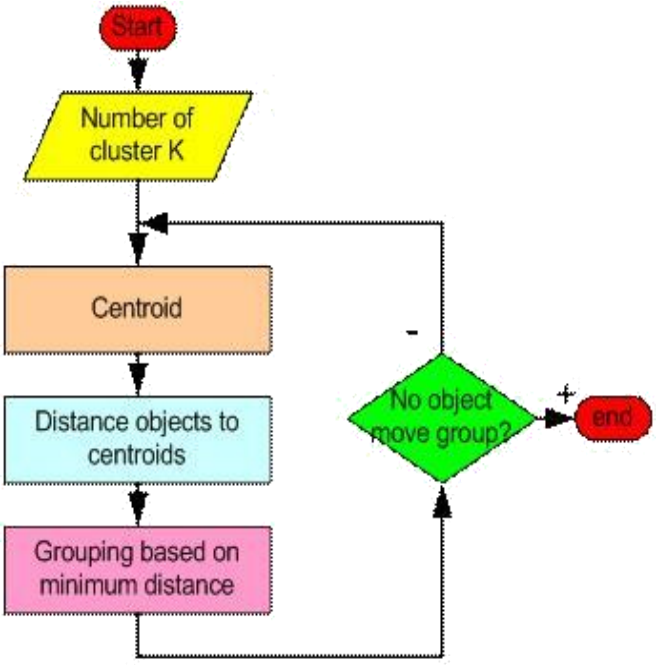
What is K-means clustering?

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K*. The algorithm works iteratively to assign each data point to one of *K* groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the *K*-means clustering algorithm are:

1. The centroids of the *K* clusters, which can be used to label new data
2. Labels for the training data (each data point is assigned to a single cluster)

Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have formed organically. The "Choosing K" section below describes how the number of groups can be determined. Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

Steps to Perform K-Means Clustering



As a simple illustration of a k-means algorithm, consider the following data set consisting of the scores of two variables on each of seven individuals:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

This data set is to be grouped into two clusters. As a first step in finding a sensible initial partition, let the A & B values of the two individuals furthest apart (using the Euclidean distance measure), define the initial cluster means, giving:

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

The remaining individuals are now examined in sequence and allocated to the cluster to which they are closest, in terms of Euclidean distance to the cluster mean. The mean vector is recalculated each time a new member is added. This leads to the following series of steps:

	Cluster 1		Cluster 2	
Step	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Now the initial partition has changed, and the two clusters at this stage having the following characteristics:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)

But we cannot yet be sure that each individual has been assigned to the right cluster. So, we compare each individual's distance to its own cluster mean and to that of the opposite cluster. And we find:

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Only individual 3 is nearer to the mean of the opposite cluster (Cluster 2) than its own (Cluster 1). In other words, each individual's distance to its own cluster mean should be smaller that the distance to the other cluster's mean (which is not the case with individual 3). Thus, individual 3 is relocated to Cluster 2 resulting in the new partition:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

The iterative relocation would now continue from this new partition until no more relocations occur. However, in this example each individual is now nearer its own cluster mean than that of the other cluster and the iteration stops, choosing the latest partitioning as the final cluster solution.

R implementation

PROGRAN:

```
df <- read.csv("iris.csv")
summary(iris)
head(iris)
names(iris)
x = iris[,-5]
y = iris$Species
kc <- kmeans(x,3)
kc
table(y,kc$cluster)
plot(x[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)
points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3, pch=23, cex=3)
```

OUTPUT:

```
> df <- read.csv("iris.csv")
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width             Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Iris-setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Iris-versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Iris-virginica :50
 Mean   :5.843   Mean   :3.054   Mean   :3.759   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 Iris-setosa
2          4.9         3.0          1.4         0.2 Iris-setosa
3          4.7         3.2          1.3         0.2 Iris-setosa
4          4.6         3.1          1.5         0.2 Iris-setosa
5          5.0         3.6          1.4         0.2 Iris-setosa
6          5.4         3.9          1.7         0.4 Iris-setosa
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
> x = iris[,-5]
> y = iris$Species
> kc <- kmeans(x,3)
> kc
K-means clustering with 3 clusters of sizes 97, 31, 22

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.301031    2.886598     4.958763   1.6958763
2     5.216129    3.538710     1.680645   0.3580645
3     4.709091    3.109091     1.395455   0.1909091
```

```
Clustering vector:
  [1] 2 3 3 3 2 2 3 2 3 3 2 3 3 3 2 2 2 2 2 2 2 3 2 2 3 2 2 2 3 3 2 2 2 3 3 2 3 3 2 2 3 3 2
 [45] 2 3 2 3 2 3 1 1 1 1 1 1 1 2 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
 [89] 1 1 1 1 1 2 1 1 1 1 2 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
[133] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 123.795876  18.639355   2.844091
 (between_SS / total_SS =  78.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> table(y,kc$cluster)

y                 1  2  3
  Iris-setosa     0 28 22
  Iris-versicolor 47  3  0
  Iris-virginica  50  0  0
> plot(x[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)
> points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3, pch=23, cex=3)
```

The K-Means function, provided by the *cluster* package, is used as follows:

kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"))

where the arguments are:

x: A numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).

centers: Either the number of clusters or a set of initial (distinct) cluster centers. If a number, a random set of (distinct) rows in x is chosen as the initial centers.

iter.max: The maximum number of iterations allowed.

nstart: If *centers* is a number, *nstart* gives the number of random sets that should be chosen.

algorithm: The algorithm to be used. It should be one of the following "Hartigan-Wong", "Lloyd", "Forgy" or "MacQueen". If no algorithm is specified, the algorithm of Hartigan and Wong is used by default

IRIS dataset

This is perhaps the best known database to be found in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.

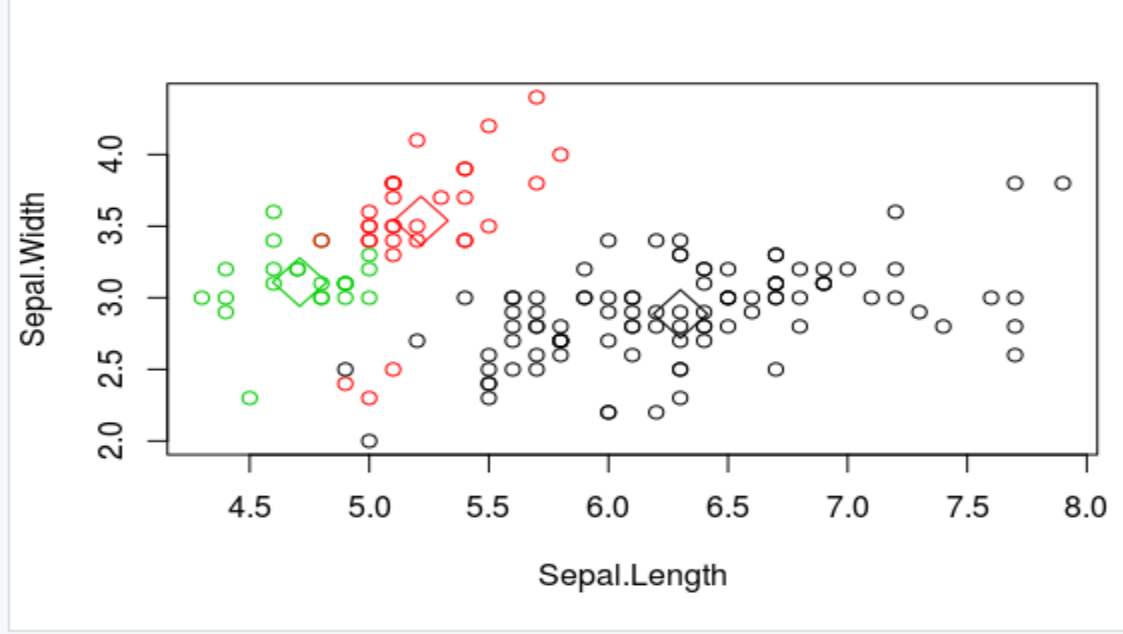
Attribute Information:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
    1. Iris Setosa
    2. Iris Versicolour
    3. Iris Virginica

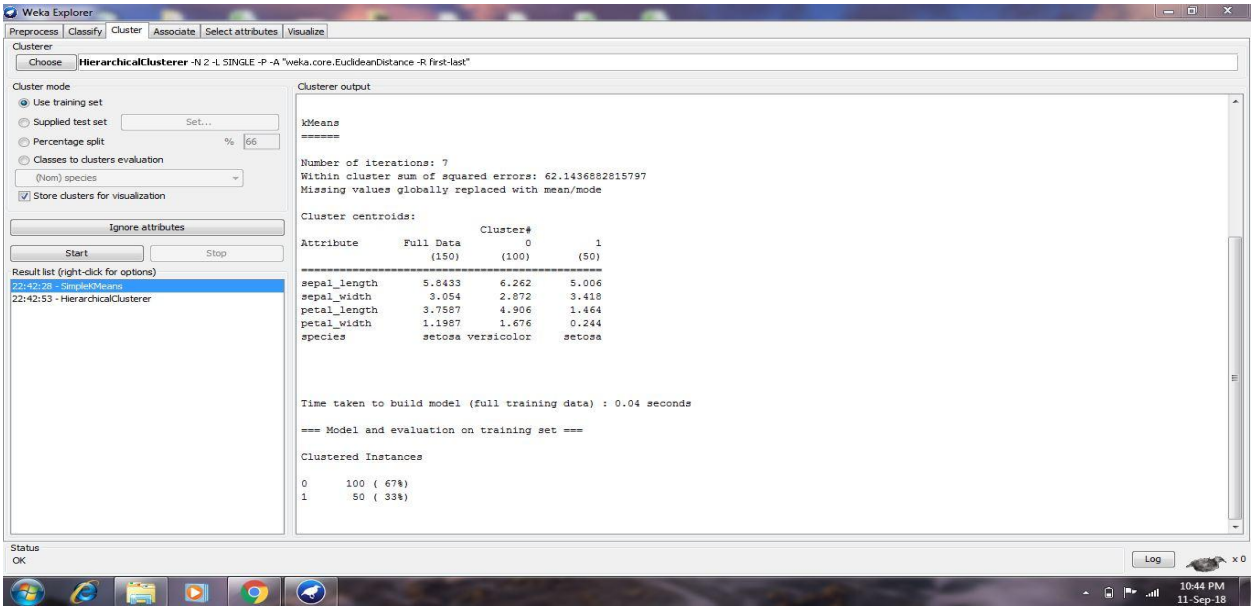
Steps

1. Set working directory
2. Get data from datasets
3. Execute the model
4. View the output
5. Plot the results

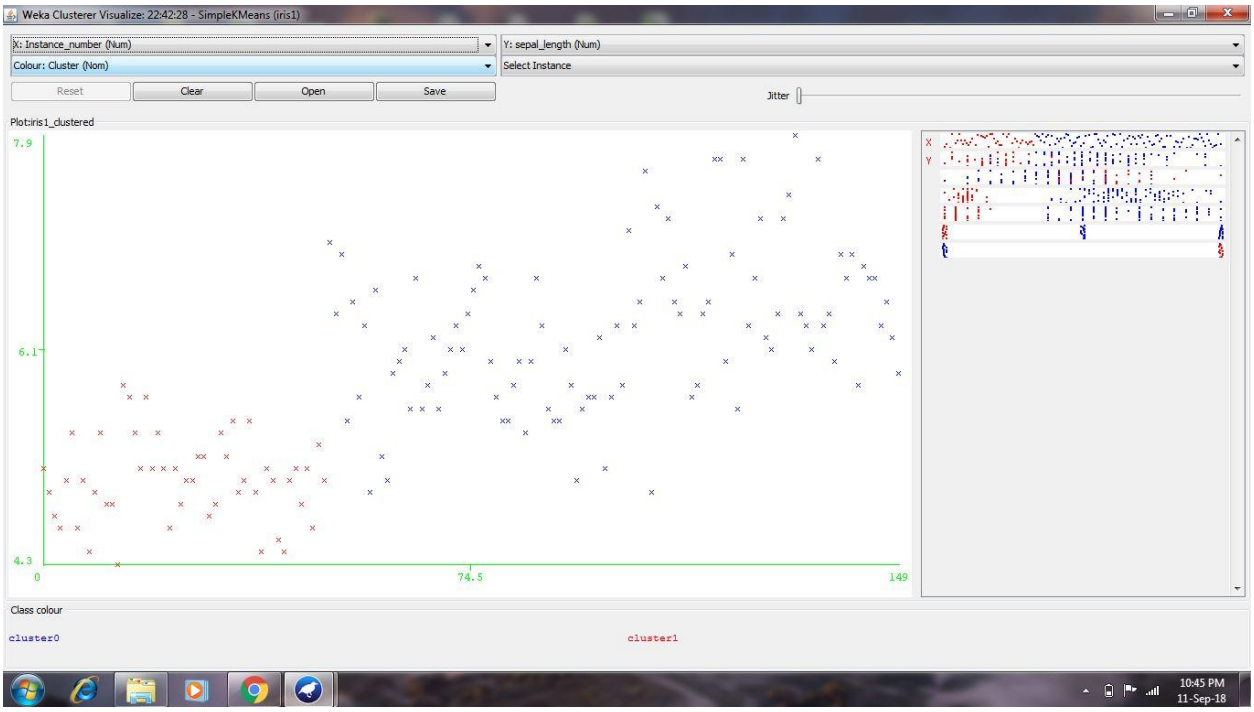
ScreenShots of R implementation



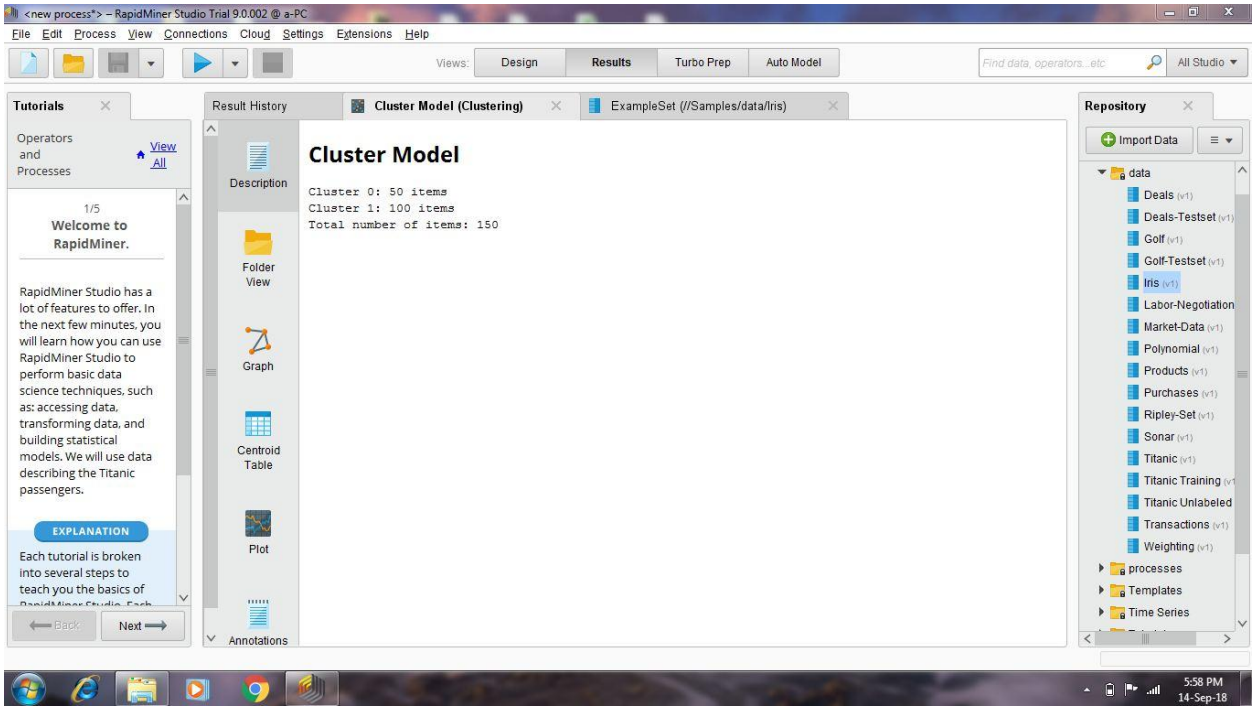
Weka Implementation of K-Means

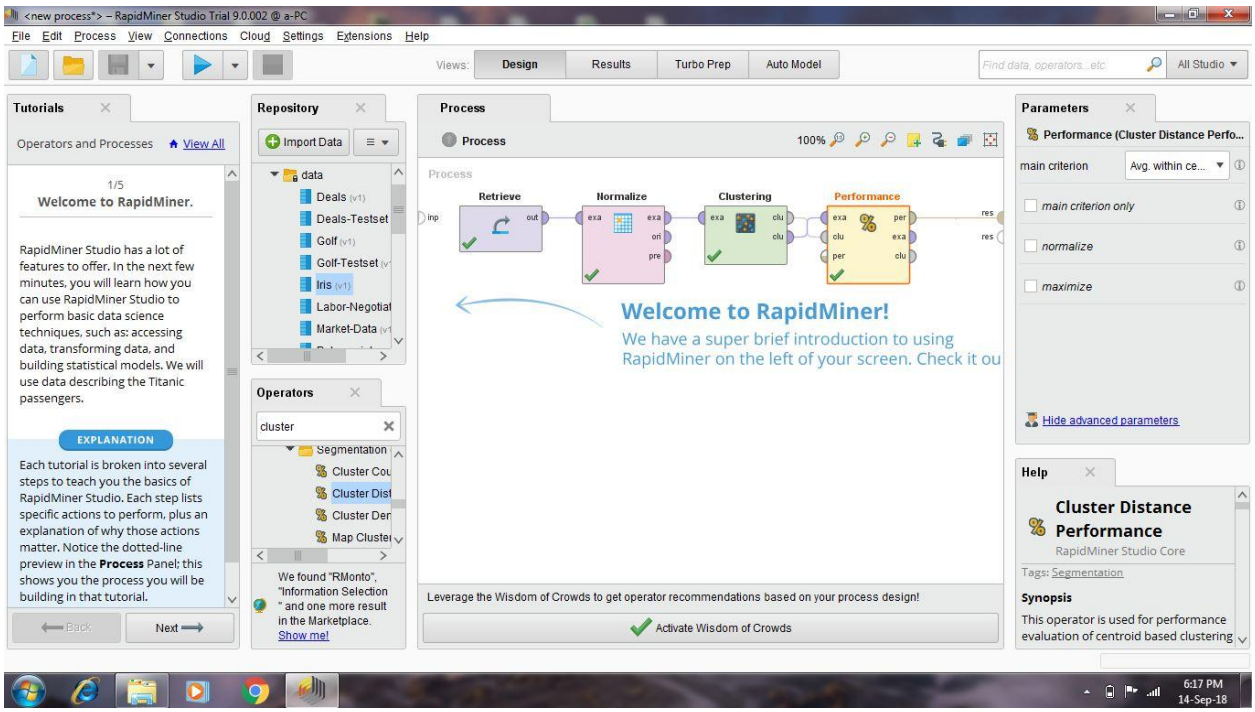


Screenshots of Weka Implementation of K-Means

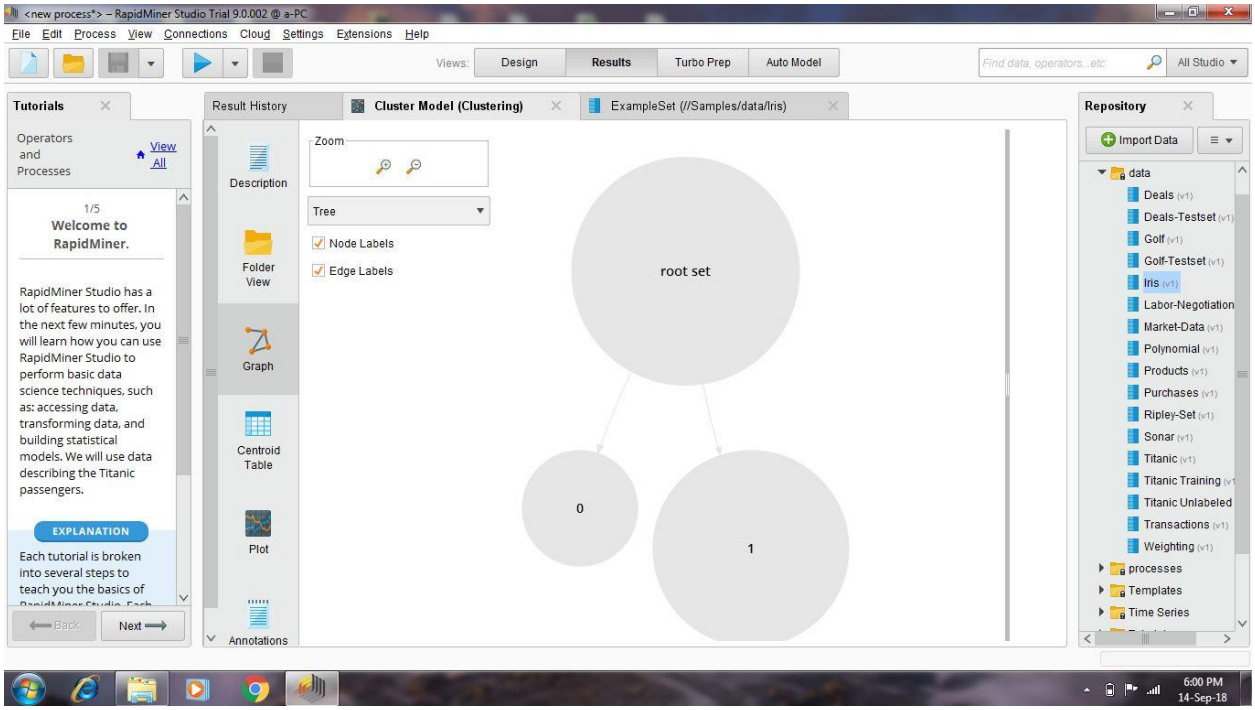


RapidMiner Implementation of K-Means





RapidMiner of Weka Implementation of K-Means



Technique 2: Hierarchical Clustering

What is Hierarchical clustering?

Given a set of N items to be clustered, and an NxN distance (or similarity) matrix, the basic process of Johnson's (1967) hierarchical clustering is this:

- Start by assigning each item to its own cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters equal the distances (similarities) between the items they contain.

- Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster.

- Compute distances (similarities) between the new cluster and each of the old clusters.
- Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

R Implementation of Hierarchical Clustering

PROGRAM:

```
df <- read.csv("iris.csv")
summary(iris)
head(iris)
names(iris)
m <- hclust(dist(iris[,1:4]), method="ave")
plot(m)
clusters = cutree(m, 3)
table(clusters, iris$Species)
```

OUTPUT:

```
> df <- read.csv("iris.csv")
> summary(iris)
   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Iris-setosa    :50
  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Iris-versicolor:50
  Median :5.800   Median :3.000   Median :4.350   Median :1.300   Iris-virginica :50
  Mean   :5.843   Mean   :3.054   Mean   :3.759   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 Iris-setosa
2         4.9         3.0          1.4         0.2 Iris-setosa
3         4.7         3.2          1.3         0.2 Iris-setosa
4         4.6         3.1          1.5         0.2 Iris-setosa
5         5.0         3.6          1.4         0.2 Iris-setosa
6         5.4         3.9          1.7         0.4 Iris-setosa
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
> m <- hclust(dist(iris[,1:4]), method="ave")
> plot(m)
> clusters = cutree(m, 3)
> table(clusters, iris$Species)

clusters Iris-setosa Iris-versicolor Iris-virginica
       1         50              0             0
       2          0             50            14
       3          0              0            36
```

hclust(d, method = "complete", members = NULL)

```
## S3 method for class 'hclust'
plot(x, labels = NULL, hang = 0.1, check = TRUE,
    axes = TRUE, frame.plot = FALSE, ann = TRUE,
    main = "Cluster Dendrogram",
    sub = NULL, xlab = NULL, ylab = "Height", ...)
```

Arguments

d	a dissimilarity structure as produced by dist.
method	the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).
members	NULL or a vector with length size of d. See the 'Details' section.
x	an object of the type produced by hclust.
hang	The fraction of the plot height by which labels should hang below the rest of the plot. A negative value will cause the labels to hang down from 0.

check	logical indicating if the x object should be checked for validity. This check is not necessary when x is known to be valid such as when it is the direct result of hclust(). The default is check=TRUE, as invalid inputs may crash **R** due to memory violation in the internal C plotting code.
labels	A character vector of labels for the leaves of the tree. By default the row names or row numbers of the original data are used. If labels = FALSE no labels at all are plotted.
axes, frame.plot, ann	logical flags as in plot.default.
main, sub, xlab, ylab	character strings for title. sub and xlab have a non-NULL default when there's a tree$call.
...	Further graphical arguments. E.g., cex controls the size of the labels (if plotted) in the same way as text.

Step 3 can be done in different ways, which is what distinguishes *single-link* from *complete-link* and *average-link* clustering

Mtcars dataset

The data was extracted from the 1974 *Motor Trend* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models)

A data frame with 32 observations on 11 variables.

[, 1] mpg Miles/(US) gallon
[, 2] cyl Number of cylinders
[, 3] disp Displacement (cu.in.)
[, 4] hp Gross horsepower
[, 5] drat Rear axle ratio
[, 6] wt Weight (lb/1000)
[, 7] qsec 1/4 mile time
[, 8] vs V/S
[, 9] am Transmission (0 = automatic, 1 = manual)
[,10] gear Number of forward gears
[,11] carb Number of carburetors

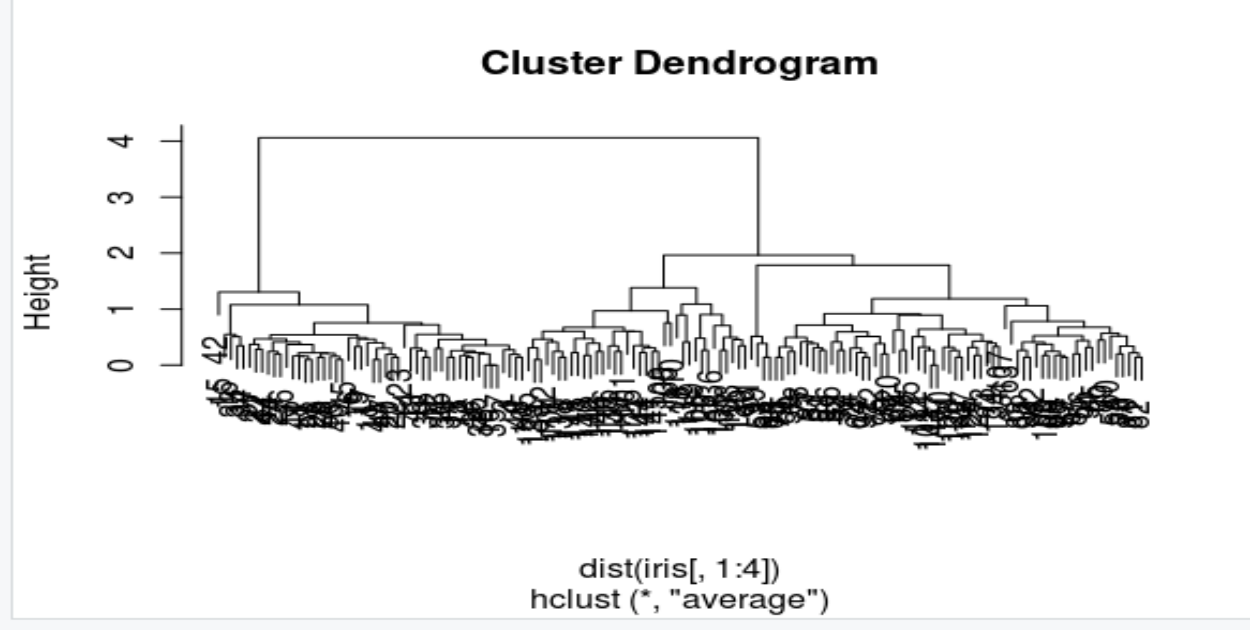
In general, there are many choices of cluster analysis methodology. The hclust function in R uses the complete linkage method for hierarchical clustering by default. This particular clustering method defines the cluster distance between two clusters to be the maximum distance between their individual components. At every stage of the clustering process, the two nearest clusters are merged into a new cluster.

With the distance matrix found in previous tutorial, we can use various techniques of cluster analysis for relationship discovery. For example, in the data set mtcars, we can run the distance matrix with hclust, and plot a dendrogram that displays a hierarchical relationship among the vehicles.

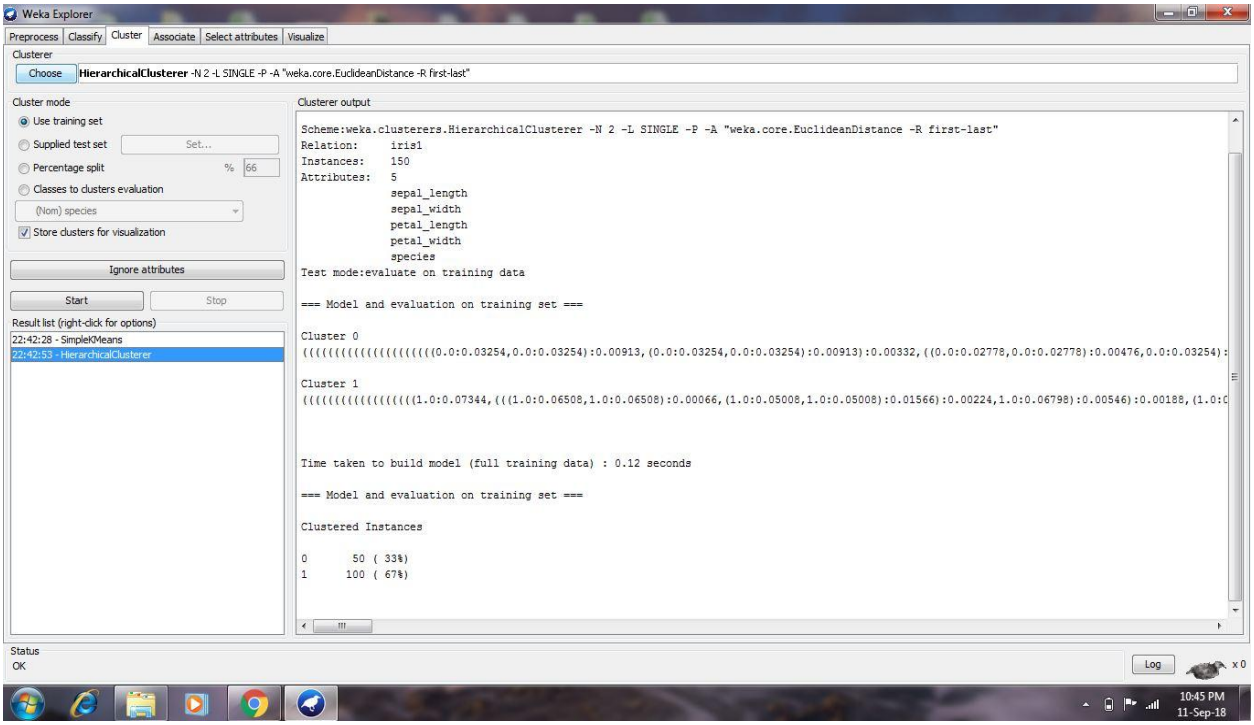
> d <- dist(as.matrix(mtcars)) # find distance matrix

> hc <- hclust(d) # apply hirarchical clustering > plot(hc) # plot the dendrogram

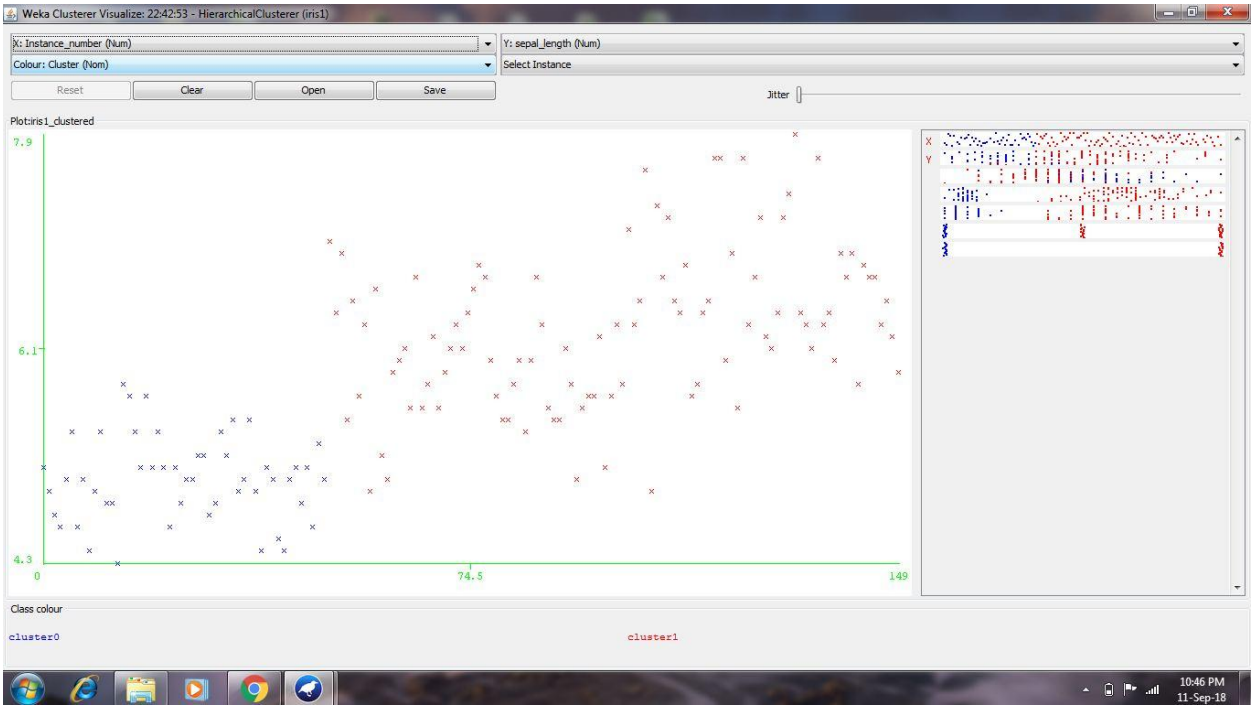
Screenshot of R Implementation of Hierarchical Clustering



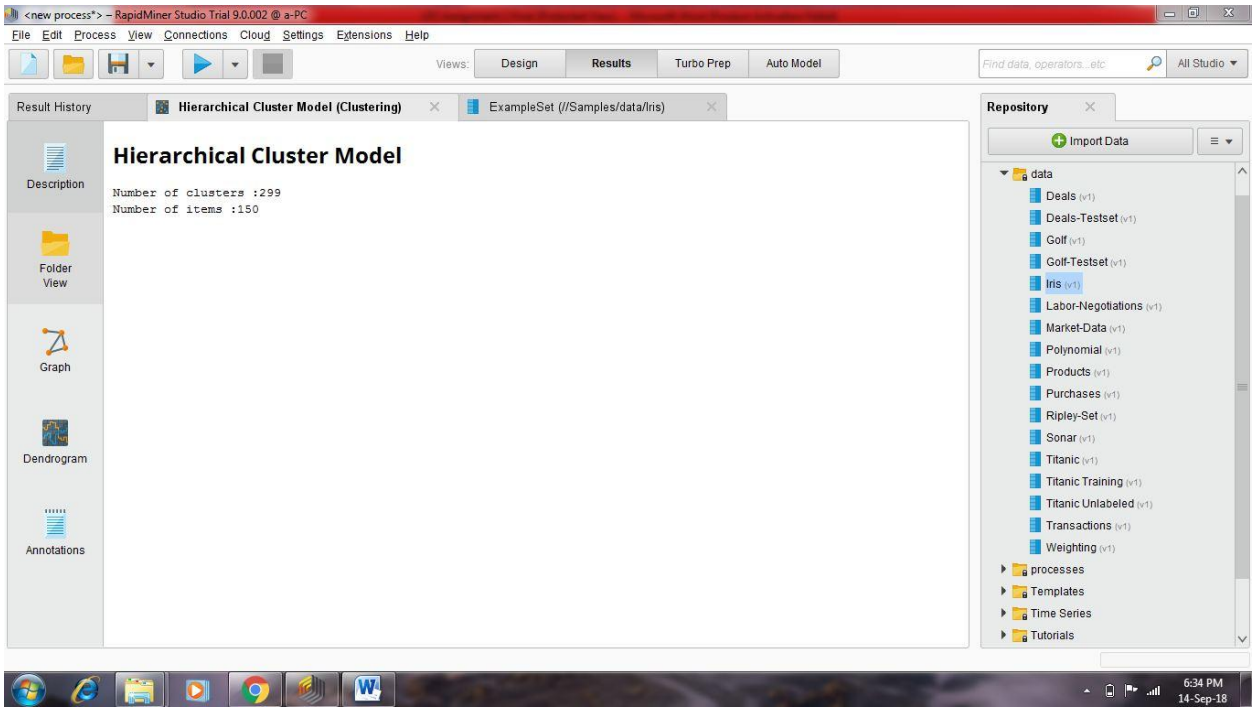
Screenshots of Weka Implementation of Hierarchical Clustering

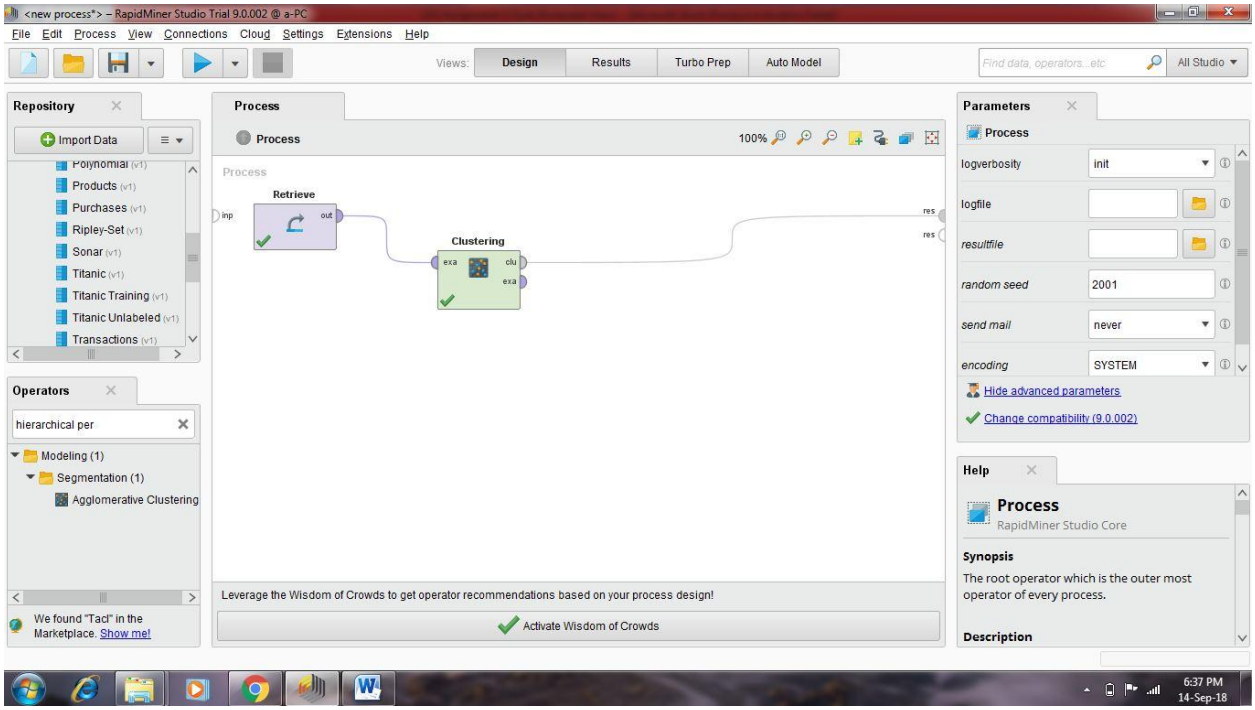


Screenshots of Weka Implementation of Hierarchical Clustering

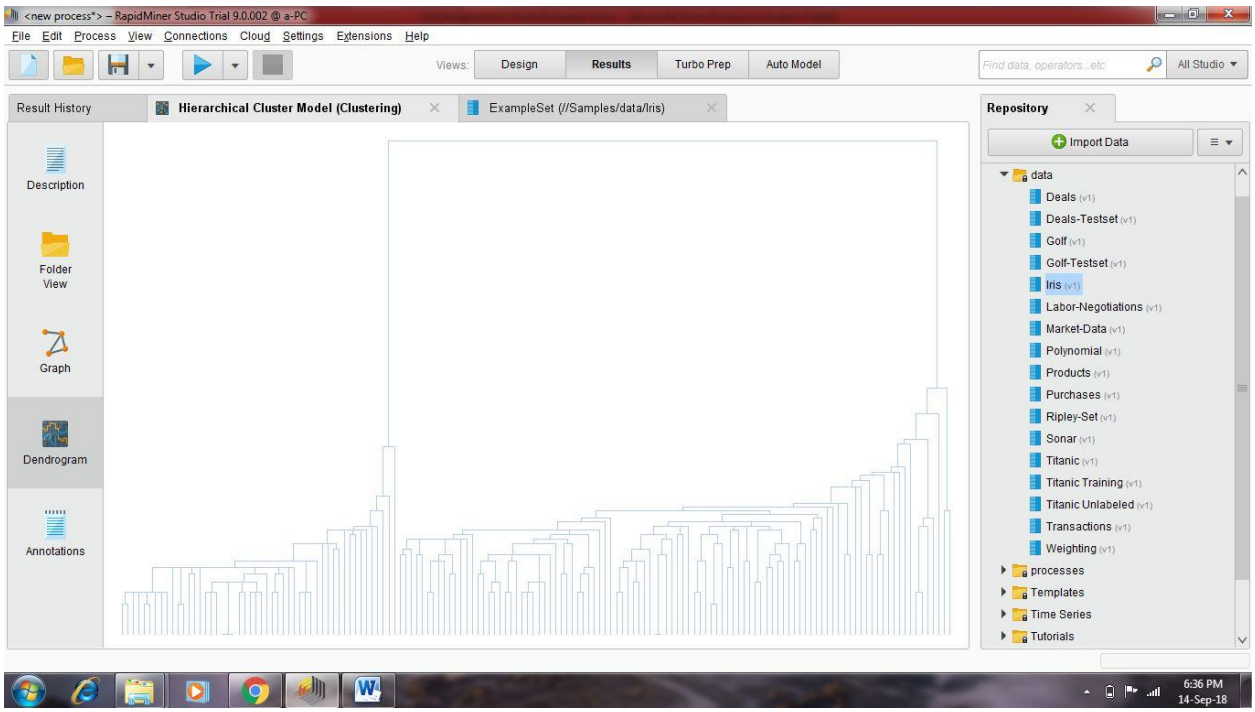


RapidMiner Implementation of Hierarchical Clustering





RapidMiner of Weka Implementation of Hierarchical Clustering



Conclusion:

Thus we have learned clustering of data using different tools and R