Cluster Analysis

Joven Sumal

November 16, 2022

1 Introduction

Given multivariate data sometimes we wish to associate or group subsets of this data based on a given criterion. The goal is to find groupings such that the observations within each group are similar while being dissimilar to the observations in other groups. Cluster analysis is a form of unsupervised learning if there is no class label otherwise known as supervised learning also referred to as classification. The approaches we will look at are Hierarchical clustering, K-Means clustering and K-Nearest Neighbours. Before we go into these approaches we need to formally define what a cluster is.

Given a collection of items a cluster is a subset of them such that the items are “close” in some sense to a central item in the cluster, while being “distant” to items in other clusters. There are different ways to define closeness and different methods for determining closeness which generally output different results. How the clusters are defined will depend on the concept of closeness used.

2 Measures of Dissimilarity for Observations

Defining clusters rely on a measure of dissimilarity or distance between each pair of observations. Observations that are farther apart are dissimilar while observations that are closer together are similar. Thus, we can say that distance is a measure of dissimilarity. There are different ways to define distance but before we do that, lets look at some general properties. Let x, $y \in \mathbb{R}^n$ where $n \in \mathbb{N}$, then a dissimilarity d has the following properties:

$$d(x, y) \geq 0 \tag{1}$$

$$d(x, x) = 0 \tag{2}$$

$$d(x, y) = d(y, x) \tag{3}$$

Property (1) states that the distance between any two points x, y must be greater than or equal to zero since we cannot have negative distance. Property (2) states the distance between any point x and itself is always 0 because we haven't moved from x so no distance is covered. Finally property (3) states the distance between x and y is the same as the distance between y and x since they are symmetric so they will have the same distance. Some ways we can define distance are:

Manhattan Distance:

$$d(x, y) = \sum_{i=1}^{n} |(x_i - y_i)| \tag{4}$$

Euclidean Distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{5}$$

Minkowski Distance:

$$d(x, y) = \left(\sum_{i=1}^{n} |(x_i - y_i)|^p \right)^{\frac{1}{p}} \text{ where } p \in \mathbb{Z}^+ \tag{6}$$

The Minkowski distance is a generalization of the Manhattan and Euclidean distance, i.e. when $p = 1$ we get (4) and when $p = 2$ we get (5). Another possible choice for measuring dissimilarities involves using the sample correlation matrix R by subtracting it from 1.

$$d(x, y) = 1 - R \tag{7}$$

With our definitions of distance we can define a distance matrix D where D is an n x n symmetric matrix containing the pairwise distances between elements of a set.

$$D = \begin{pmatrix} 0 & d_{12} & d_{13} & \dots & d_{1n} \\ d_{21} & 0 & d_{23} & \dots & d_{2n} \\ d_{31} & d_{32} & 0 & \dots & d_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \dots & 0 \end{pmatrix} \tag{8}$$

Note the diagonal entries are 0 from property (2) and the entries $d_{ij} = d_{ji}$ for $1 \leq i, j \leq n$ from property (3).

3 Hierarchical Clustering

Hierarchical clustering is an unsupervised clustering technique that uses a sequential approach to build a hierarchy of clusters. It attempts to find a new set of clusters from a given set of clusters which can be larger or smaller in size depending on the approach. The two common approaches are agglomerative hierarchical and divisive hierarchical. One of the benefits of using this method is we can use a dendrogram, a hierarchical tree diagram showing the arrangement of clusters along with distances where clusters merge.

3.1 Agglomerative Hierarchical Approach

The agglomerative hierarchical approach starts with each observation being its own cluster. Then at each step of the algorithm, pairs of clusters that are most similar will merge together to form a new cluster. As the algorithm progresses, the number of clusters decrease and the sizes of clusters increase until there is a single cluster containing all the observations. This is referred to as the bottom-up approach. If the goal is to get small clusters this method might be preferable. The algorithm proceeds as follows:

- Select a dissimilarity and method for computing distance between clusters.
- Select an initial set of clusters.
- While number of clusters > 1:
  - For every pair of clusters in the current set of clusters compute their distance.
  - Merge the two clusters with the smallest distance to form a new cluster.
  - Reduce the total number of clusters by one.

3.2 Divisive Hierarchical Approach

The divisive hierarchical approach starts with one cluster containing all observations. Then at each step of the algorithm, the most dissimilar cluster subdivides forming its own cluster. As the algorithm progresses, the number of clusters increase and the sizes of clusters decrease until each observation becomes its own cluster. This is referred to as the top-down approach. If the goal is to get large clusters this method might be preferable. The algorithm proceeds as follows:

- Select a dissimilarity and method for computing distance between clusters.
- Select an initial set of clusters.
- While number of clusters $<$ the total number of observations:
  - For every pair of clusters in the current set of clusters compute their distance.
  - Separate the two clusters with the largest distance to form a new cluster.
  - Increase the total number of clusters by one.

3.3 Measures of Dissimilarity for Clusters

We have measures of dissimilarities for observations however we don't have measures of dissimilarities for clusters of observations. To decide which clusters should be combined for agglomerative hierarchical approach or split for divisive hierarchical approach we will introduce the following most popular measures.

Let A, B be two sets of clusters containing observations x, y respectively and $D(A, B)$ be the distance between the clusters and $d(x, y)$ be the distance between observations, then the following methods are used to compute the distance:

Simple Linkage (Nearest Neighbour):

$$D(A,B) = \min\{d(x,y) : \ x \in A, \ y \in B\} \tag{9}$$

Complete Linkage (Farthest Neighbour):

$$D(A,B) = \max\{d(x,y) : \ x \in A, \ y \in B\} \tag{10}$$

Average Linkage (Group Average):

$$D(A,B) = \frac{1}{n_A n_B} \sum_{i=1}^{n_A} \sum_{j=1}^{n_B} d(x_i, d_j) \tag{11}$$

Wards Method:

$$D(A,B) = ESS_{AB} - (ESS_A + ESS_B) \tag{12}$$

Note, ESS is the sum of the Euclidean distance squared of each observation to the centroid. A centroid is the mean of the observations assigned to the cluster.

4 K-Means Clustering

K-Means clustering is an unsupervised clustering technique. You begin by choosing the number of clusters k and each cluster is represented by its centroid. The algorithm tries to minimize the total within-cluster variation which is ESS or the sum of the squared Euclidean distances between each observation and the centroid. Two methods that can help determine the optimal value for k is the elbow method and the average silhouette method. The elbow method is a plot of the total within-cluster variation versus the number of clusters and where the bend occurs is the suggested number of clusters. The average silhouette method is a plot of the average silhouette width versus the number of clusters where the largest average silhouette width is the suggested number of clusters. The K-Means algorithm proceeds as follows:

- Select k the number of clusters.
- Initialize clusters by either:
  - Forming an initial random assignment of the observations into k clusters.
  - Selecting k seeds and assign each observation to a cluster associated with the closest seed.
- Compute the centroid for each of the k clusters.
- For every observation compute their distance to each centroid and assign them to the cluster associated with the closest centroid.
- When the observations have settled stop the algorithm.

5 K-Nearest Neighbours

K-Nearest Neighbours is a non-parametric supervised clustering technique that is commonly used for classification. To start, you'll have your training data and test data. The goal is to classify each observation in the test data using the training data. To determine closeness we calculate the distance (usually Euclidean distance) between the observations in the test data with the observations in the training data. Then the k closest observations are selected and the class that occurs the most from those k observations is assigned. The optimal value of k is usually the square root of the total number of observations. The K-Nearest Neighbour algorithm proceeds as follows:

- Select k the number of neighbours.
- For each observation in the test data:
  - Compute its distance with the training data.
  - Sort the calculated distances in ascending order.
  - Obtain the most frequent class by looking at the top k rows of the sorted distances.
  - Assign the class to the observation.

6 Comparing the Clustering Methods

Table 2 summarizes some of the key differences and similarities of the clustering techniques we've mentioned. You can see from the table how they excel at different things and have their corresponding drawbacks. It's important to recognize when its appropriate to use a specific clustering algorithm and this table will help you decide which one to implement. One significant sign is the type of data you have, if it's highly multi dimensional then you probably wouldn't use K-Means or K-Nearest Neighbours or if the data is very large with many observations then K-Means would excel over the others. Maybe you're given class labels in your data and want to do classification, then you'd consider using K-Nearest Neighbours.

	Hierarchical	**K-Means**	**K-Nearest Neighbours**
Type of Algorithm	Unsupervised	Unsupervised	Supervised
Pre-Specify k	No	Yes	Yes
Easy to Implement	Yes	Yes	Yes
Works Well with Large Datasets	No	Yes	No
Sensitive to Outliers	Yes	Yes	Yes
Costly Computation	Yes	No	Yes
Works Well with High Dimensional Data	Yes	No	No

Table 2: Comparison table of the clustering methods.

7 R Tutorial for the Clustering Methods

Begin by loading any necessary libraries that you'll need. Then we read in our data and in our example we use an iris data set. We define a function to normalize our data because clustering algorithms rely on using a distance measure of some sort which are affected by the scale of variables. After we normalize the explanatory variables which is the first 4 columns in our data, we produce a scatter plot matrix shown in Figure 1 to get an initial observation of our data. We can visually see some of the dependencies along with any clustering.

```
# loading libraries
library(factoextra) # for dendrogram
library(class) # for K-NN
library(caret) # for confusion matrix

# Reading in iris data
iris.raw <- read.csv('iris.data', header=FALSE)

# Define min-max normalization function
min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
  }

# Normalize first four columns in iris data set
iris.norm <- as.data.frame(lapply(iris.raw[1:4], min_max_norm))

# creating scatter plot matrix of normalized iris data
pairs(iris.norm, main='Scatter Plot Matrix of Normalized Iris Data')
```

Next, we implement Hierarchical clustering. To do this, we need to compute a distance matrix, we'll be using the Euclidean distance from equation (5). Then we implement Hierarchical clustering using the hclust function and specify the method we'd like to use. We'll compute the four methods we've discussed and compare their dendrograms to see which method produces the best clusters. Since we have 3 classes of flowers in our data, we'll use 3 clusters. Aside from looking at the dendrogram, You can also view which observations are in which cluster using the cutree function.

The dendrograms of each of the four methods are produced in Figures 2, 3, 4, 5 and show that Figure 3 the farthest neighbour method forms the best clusters. Finally we create a scatter plot matrix of the normalized iris data clustered using Hierarchical clustering with farthest neighbour in Figure 6.

```
###################################
##### Hierarchical Clustering #####
###################################
# Computing distance matrix using euclidean distance
d <- dist(iris.norm, method="euclidean",diag=T,upper=T)

# Single linkage method (Nearest Neighbour)
hc_single <- hclust(d=d,method="single")
cutree(hc_single, k=3)
fviz_dend(hc_single, cex=0.5, k=3, main="Nearest Neighbour",
          color_labels_by_k=TRUE, rect=TRUE)

# Complete linkage method (Farthest Neighbour)
hc_complete <- hclust(d=d,method="complete")
cutree(hc_complete, k=3)
fviz_dend(hc_complete, cex=0.5, k=3, main="Farthest Neighbour",
          color_labels_by_k=TRUE, rect=TRUE)

#  Average linkage method (Group Average)
hc_avg <- hclust(d=d,method="average")
cutree(hc_avg, k=3)
fviz_dend(hc_avg, cex=0.5, k=3, main="Group Average",
          color_labels_by_k=TRUE, rect=TRUE)

# Ward's method
hc_ward <- hclust(d=d,method="ward.D")
cutree(hc_ward, k=3)
fviz_dend(hc_ward, cex=0.5, k=3, main="Ward's",
          color_labels_by_k=TRUE, rect=TRUE)

# scatter plot matrix using Farthest Neighbour
pairs(iris.norm, pch=21, bg=c("red","blue", "green")[cutree(hc_complete, k=3)],
      oma=c(4,4,4,10))
legend("topright", legend=c("Cluster 1","Cluster 2", "Cluster 3"),
       fill=c("red","blue", "green"),xpd=NA,cex=0.4)
```

Now we implement K-Means clustering using 3 clusters with the kmeans function. The parameter centers is the number of clusters that you want. One way to help determine the number of clusters is to create a plot of the elbow method using the function fviz_nbclust. We produce this plot in Figure 7 and it shows that as you increase the number of clusters, the total within-cluster variation decreases. Furthermore, you can use the average silhouette method using the same function by changing the method parameter. This plot is shown in Figure 8 and measures the quality of clustering by looking at the average silhouette width where higher values indicate better clustering. Despite indicting 2 clusters, we'll use 3 since we have 3 classes of flowers in our data set. We produce a scatter plot matrix of normalized iris data clustered using K-Means in Figure 9.

To help visualize K-Means clustering we can perform a PCA to reduce our data to 2 dimensions. Since our focus is on clustering we won't get into too much detail on PCA however the code below with comments will help guide you. In Figure 10 we display the results of PCA creating a plot of the first two principal components clustered using our K-Means from before and labeling the points to see which observation belong in which cluster. It reveals 3 well-defined clusters.

```
##############################
##### K-Means Clustering #####
##############################
# elbow method
fviz_nbclust(iris.norm, kmeans, method = "wss")

# Average silhouette for kmeans
fviz_nbclust(iris.norm, kmeans, method = "silhouette")

# K-means into 3 clusters
km_3 <- kmeans(iris.norm, centers=3) # centers = 3 -> 3 clusters
km_3$cluster # which observations belong to which cluster
km_3$size # number of observations in each cluster

# scatter plot matrix using k-means clustering
pairs(iris.norm, pch=21, bg=c("red","blue","green")[km_3$cluster],
      oma=c(4,4,4,10))
legend("topright", legend=c("Cluster 1","Cluster 2", "Cluster 3"),
       fill=c("red","blue","green"),xpd=NA,cex=0.5)

# Visualize k-means clusters by performing PCA and plot according to first 2 PCs
pc <- prcomp(iris.norm, center=T,scale=T)
eigen <- (pc$sdev)^2 # eigenvalues/variances
pvar= eigen/sum(eigen) # individual contribution for each variable i.e. variance
csum <- cumsum(pvar) # cumulative contribution for each variable
table <- cbind(eigen, pvar, csum)
# the average of the eigenvalues
sum(eigen)/4
# scree plot to help determine number of components
screeplot(pc,type="lines", main="Screeplot")
# eigenvectors for the principal components we retain
pc$rotation[,1:2]

# Coordinates of individuals
ind.coord <- as.data.frame(pc$x)
# Add clusters obtained using the K-means
ind.coord$cluster <- factor(km_3$cluster)
# plot PC1 vs PC2
plot(x=ind.coord$PC1, y=ind.coord$PC2, cex=2,pch=19,
     col=c("red","Yellow","green")[ind.coord$cluster],
     xlab="PC1 (72.77%)", ylab="PC2 (23.03%)")
legend("topright", legend=c("Cluster 1","Cluster 2", "Cluster 3"),
       fill=c("red","Yellow","green"),xpd=NA,cex=0.5)
abline(v=0,h=0)
# labeling points with their cluster assignment
text(PC2~PC1, labels=rownames(ind.coord), data=ind.coord,cex=0.9,font=0.5)
```

Lastly we implement our K-Nearest Neighbour model using the knn function from the class library. First we need to split our data into training and test sets. We randomly sample 70% of our data to use for our training set and the other 30% for our test set. Then we store the class labels from column 5 of our iris data set to verify our predictions. To find the optimal value of k, we can take the square root of the total number of observations in our data. Then we implement our model using this value of k. To check the accuracy of our model we create a confusion matrix shown in Figure 11 which reveals we correctly guess the flower class 93% of the time. Also to visualize the accuracy of our model for varying k values, we create an accuracy plot shown in Figure 12.

```
################################
##### K-Nearest Neighbours #####
################################
# random selection of 70% data.
iris.ind <- sample(1:nrow(iris.norm), size=nrow(iris.norm)*0.7,replace = FALSE)
# split data into training and test sets
train.iris <- iris.norm[iris.ind,] # 70% training data
test.iris <- iris.norm[-iris.ind,] # 30% test data

# Creating seperate dataframe for verification, i.e. our target
train.iris_labels <- iris.raw[iris.ind,5]
test.iris_labels <- iris.raw[-iris.ind,5]

# Take square root of the number of observations to find optimal k value
sqrt(nrow(iris.norm))
# implementing knn model
knn.12 <- knn(train=train.iris, test=test.iris, cl=train.iris_labels, k=12)

# Checking accuracy of our KNN model
confusionMatrix(table(knn.12 ,test.iris_labels))

# Calculating accuracy of k values from 1 to 13.
acc.k <- 1
k <- 1
for (i in 1:13){
  knn.model <- knn(train=train.iris, test=test.iris, cl=train.iris_labels, k=i)
  acc.k[i] <- 100 * sum(test.iris_labels == knn.model)/NROW(test.iris_labels)
  k=i
  cat(k,'=',acc.k[i],'
')
}

#Accuracy plot
plot(acc.k, type="b", xlab="k",ylab="Accuracy")
```

8 Conclusion

This concludes our R tutorial and analysis on Hierarchical clustering, K-Means Clustering and K-Nearest Neighbours. There are other algorithms that we haven't covered, however these are some of the popular ones that you should know. They each have their advantages and disadvantages as well as some similarities that they share. Hopefully, this paper has helped you in understanding a bit more about cluster analysis.

9 Appendix

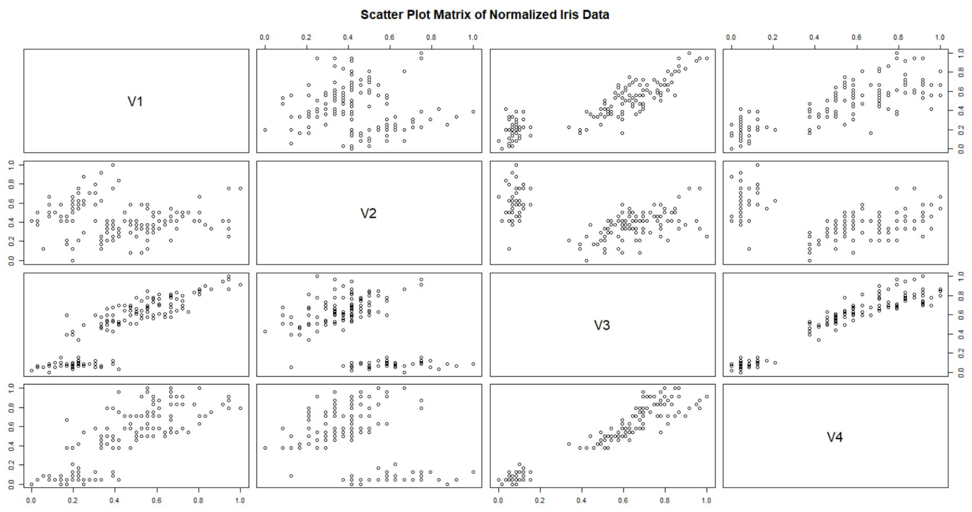


Figure 1: Scatter plot matrix of normalized iris data.

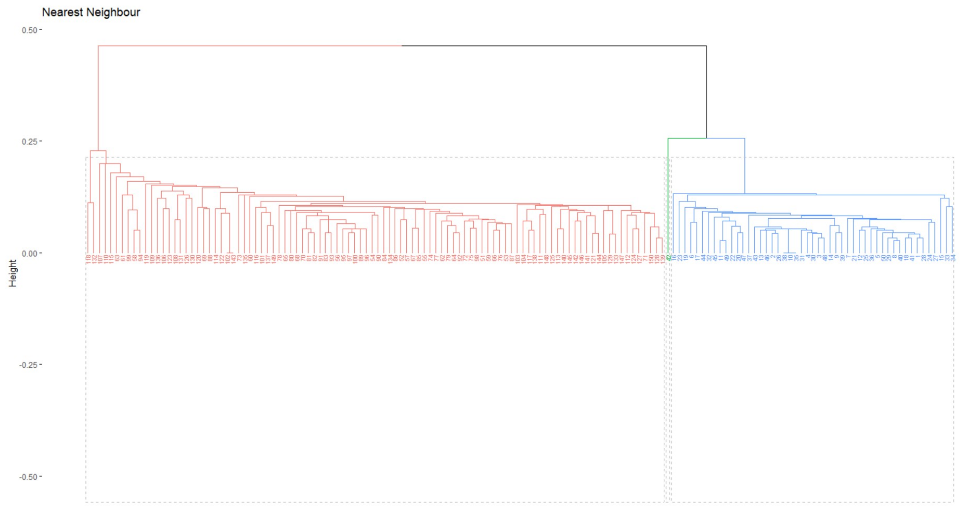


Figure 2: Dendrogram of Hierarchical clustering using Nearest Neighbour.

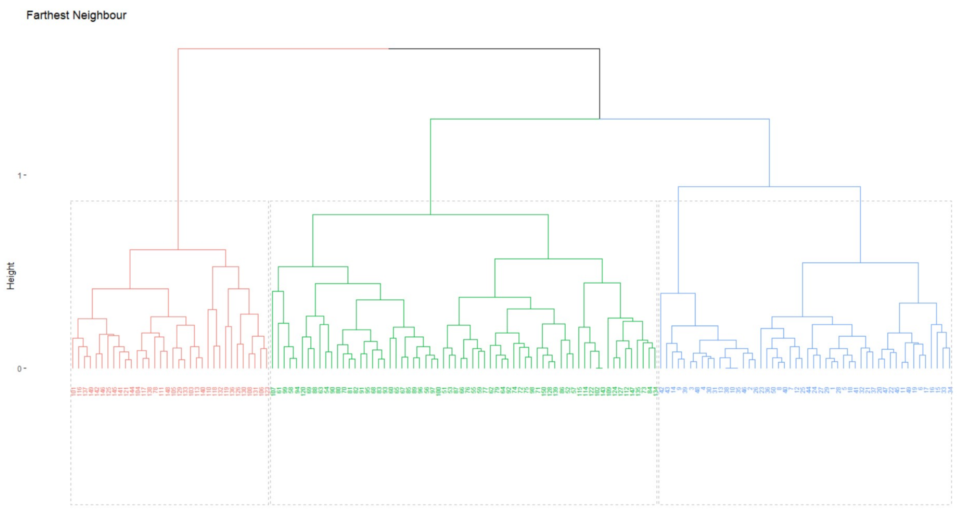


Figure 3: Dendrogram of Hierarchical clustering using Farthest Neighbour.

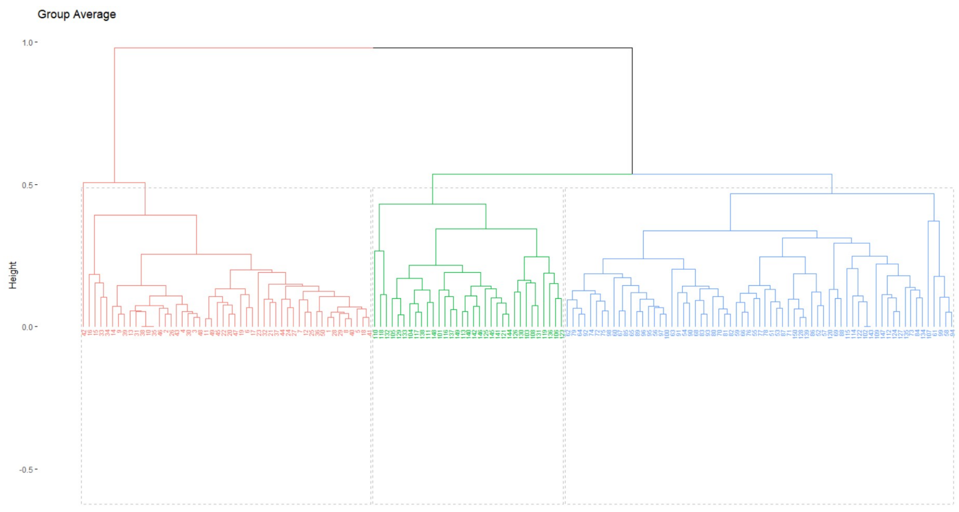


Figure 4: Dendrogram of Hierarchical clustering using Group Average.

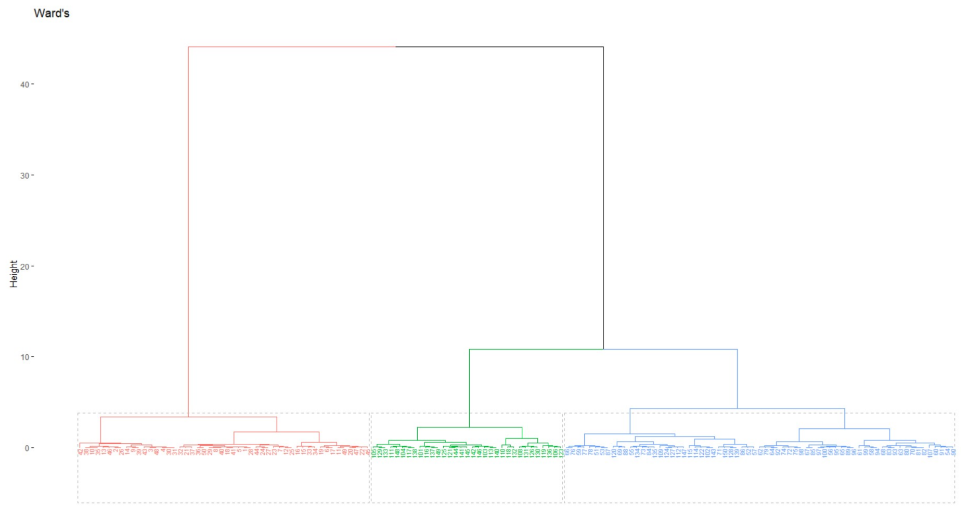


Figure 5: Dendrogram of Hierarchical clustering using Ward's method.

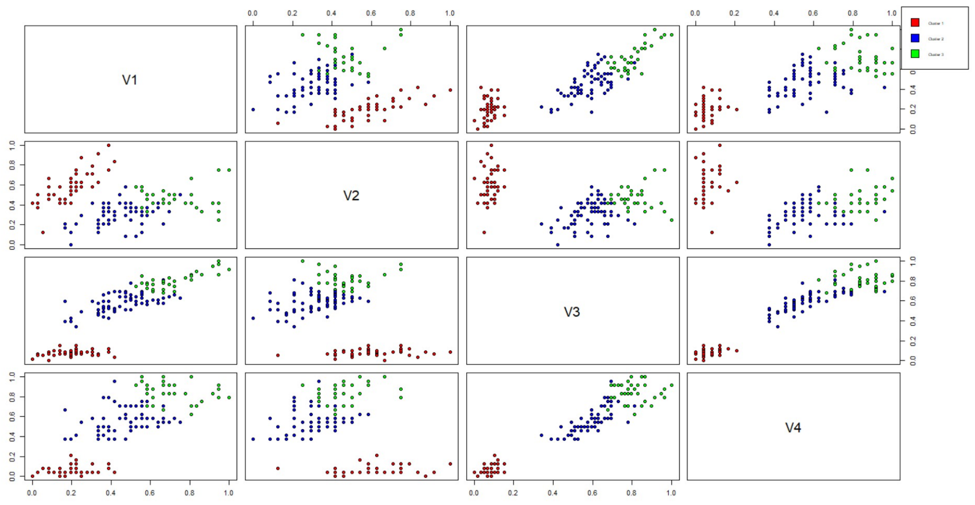


Figure 6: Scatter plot matrix of normalized iris data clustered using Hierarchical clustering with farthest neighbour method.

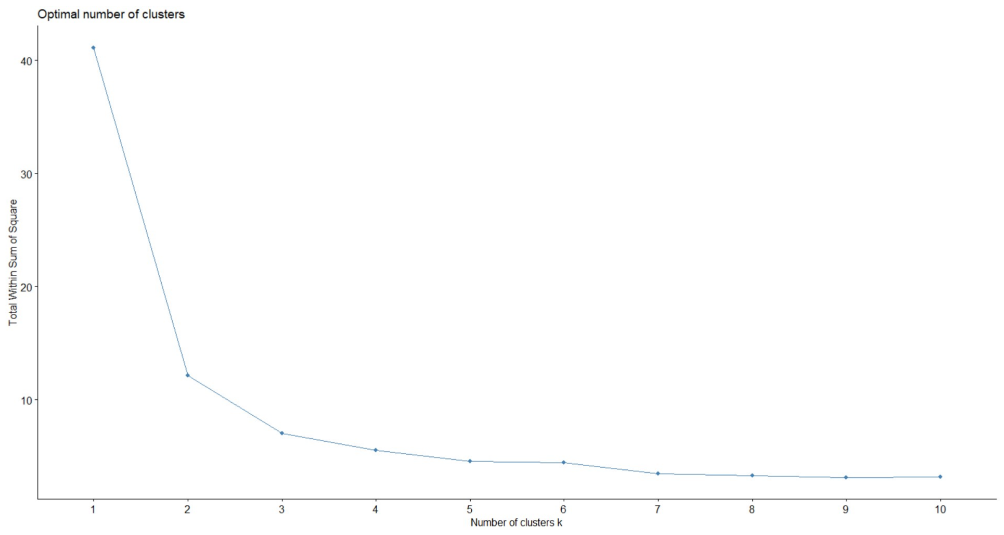


Figure 7: Elbow method plot.

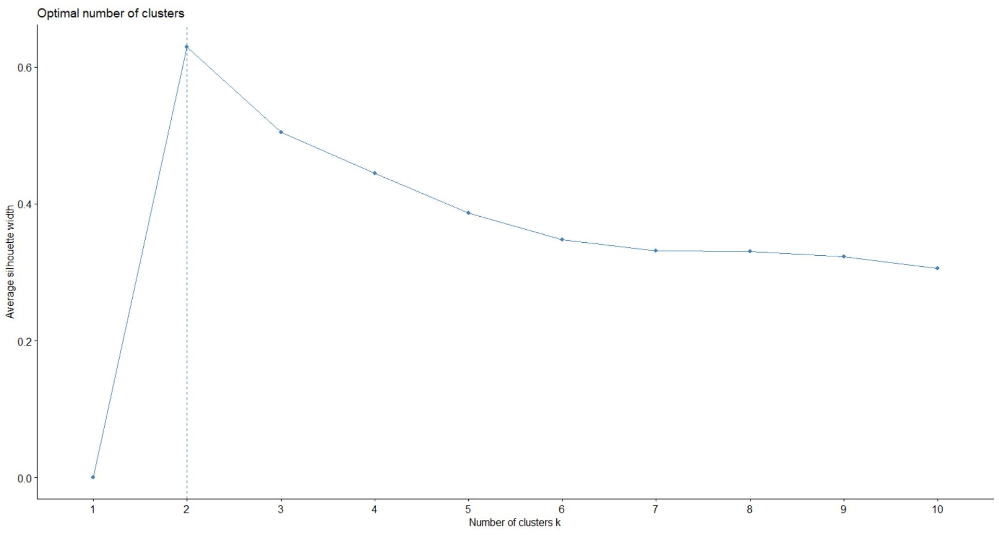


Figure 8: Average silhouette method plot.

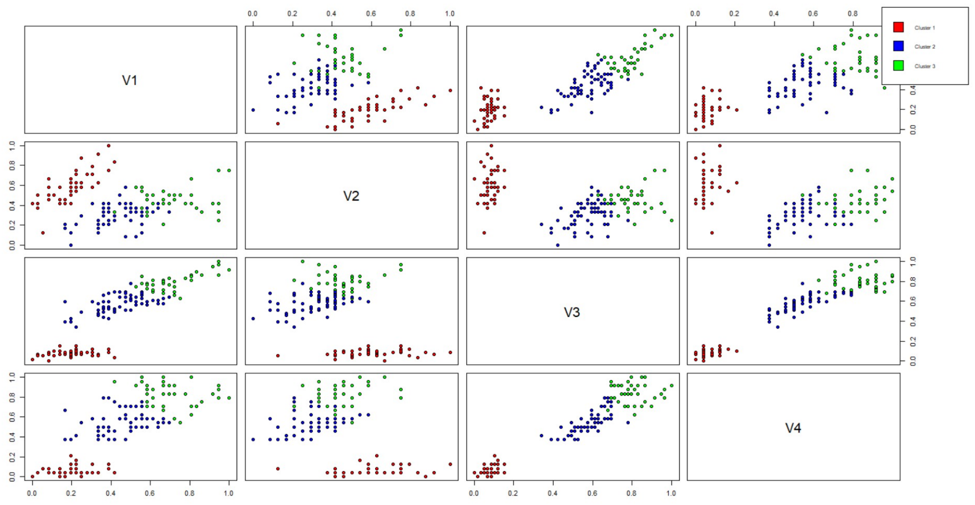


Figure 9: Scatter plot matrix of normalized iris data clustered using K-Means.

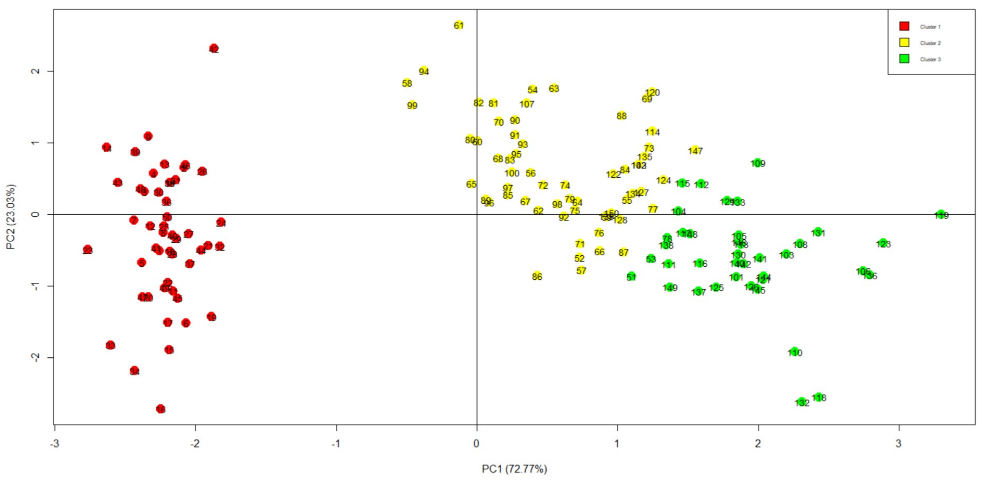
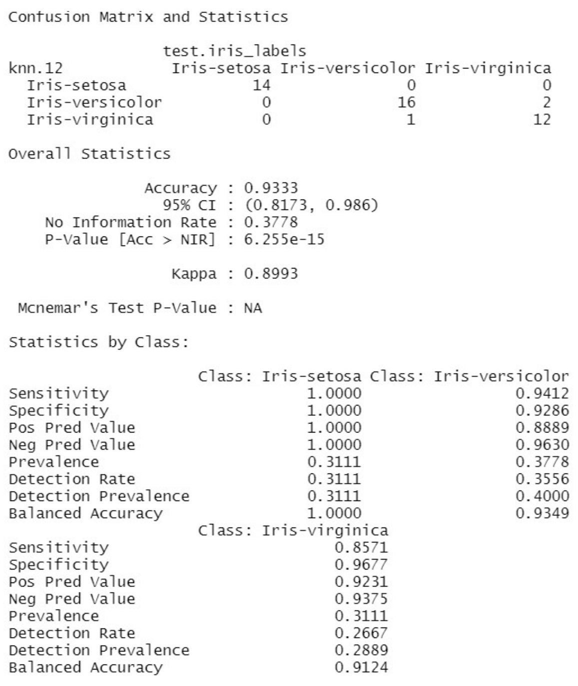


Figure 10: Scatter plot of first two principal components with labelled points.



```
Confusion Matrix and Statistics

                 test.iris_labels
knn.12            Iris-setosa Iris-versicolor Iris-virginica
  Iris-setosa              14               0              0
  Iris-versicolor           0              16              2
  Iris-virginica            0               1             12

Overall Statistics

               Accuracy : 0.9333
                 95% CI : (0.8173, 0.986)
    No Information Rate : 0.3778
    P-Value [Acc > NIR] : 6.255e-15

                  Kappa : 0.8993

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Iris-setosa Class: Iris-versicolor
Sensitivity                      1.0000                 0.9412
Specificity                      1.0000                 0.9286
Pos Pred Value                   1.0000                 0.8889
Neg Pred Value                   1.0000                 0.9630
Prevalence                       0.3111                 0.3778
Detection Rate                   0.3111                 0.3556
Detection Prevalence             0.3111                 0.4000
Balanced Accuracy                1.0000                 0.9349
                     Class: Iris-virginica
Sensitivity                         0.8571
Specificity                         0.9677
Pos Pred Value                      0.9231
Neg Pred Value                      0.9375
Prevalence                          0.3111
Detection Rate                      0.2667
Detection Prevalence                0.2889
Balanced Accuracy                   0.9124
```

Figure 11: Confusion matrix of K-Nearest Neighbour model for $k = 12$.

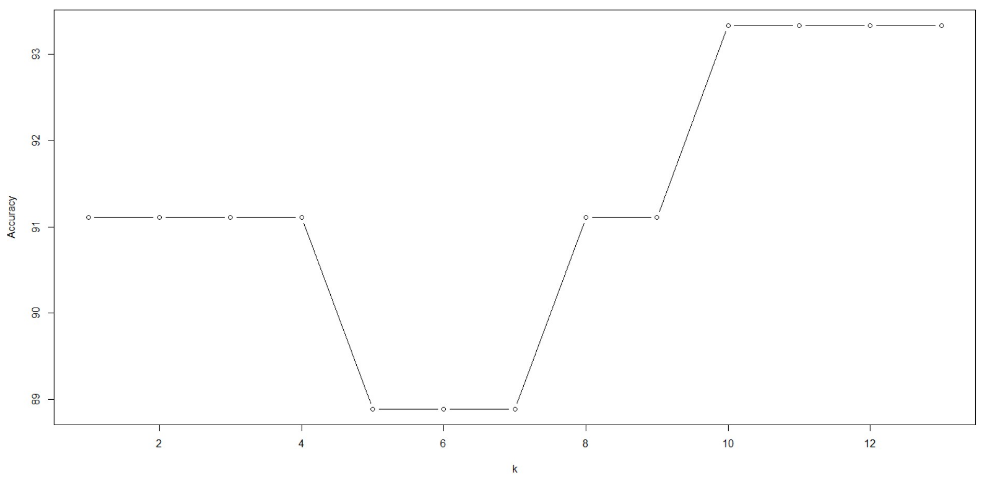


Figure 12: Accuracy Plot of K-Nearest Neighbour model for k ranging from 1 to 13.