Clustering techniques (Packages - cluster, factoextra, magrittr)

Expt No: 8 May 21, 2019
Author: Benisha.S (18244997153)

Aim

To implement different clusters using R program.

Description

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters. Types of Clusters:

- Hard Clustering: In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.
- Soft Clustering: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 10 clusters of the retail store.

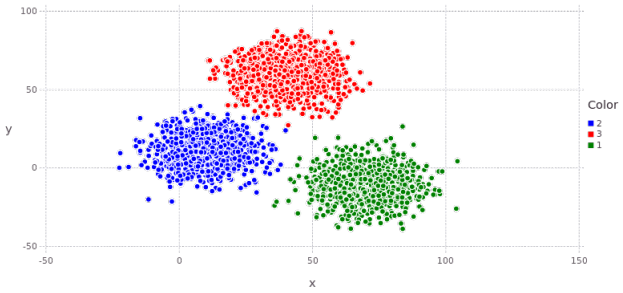

Figure 1: Cluster

Tools and Packages

1. Tools:
    - R studio
2. Packages:
    - factoextra
    - cluster
    - magrittr
    - NbClust
3. Dataset:
    - USArrests

Procedure

1. **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:
   - The centroids of the K clusters, which can be used to label new data.
   - Labels for the training data (each data point is assigned to a single cluster).
2. **K-medoids Clustering:** The k-medoids algorithm is a clustering algorithm related to the k-means algorithm and the medoidshift algorithm. Both the k-means and k-medoids algorithms are partitional (breaking the dataset up into groups). K-means attempts to minimize the total squared error, while k-medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids chooses datapoints as centers (medoids or exemplars).
3. **Hierarchical clustering:** Hierarchical clustering involves creating clusters that have a predetermined ordering from top to bottom. For example, all files and folders on the hard disk are organized in a hierarchy. There are two types of hierarchical clustering, Divisive and Agglomerative.
   - Divisive method: In divisive or top-down clustering method we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters. Finally, we proceed recursively on each cluster until there is one cluster for each observation. There is evidence that divisive algorithms produce more accurate hierarchies than agglomerative algorithms in some circumstances but is conceptually more complex.
   - Agglomerative method: In agglomerative or bottom-up clustering method we assign each observation to its own cluster. Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters. Finally, repeat steps 2 and 3 until there is only a single cluster left.
4. **Dendrogram:** A dendrogram is a diagram that shows the hierarchical relationship between objects. It is most commonly created as an output from hierarchical clustering. The main use of a dendrogram is to work out the best way to allocate objects to clusters. The dendrogram below shows the hierarchical clustering of six observations shown to on the scatterplot to the left. (Dendrogram is often miswritten as dendogram.)

Pseudo Code

1. **k-means Cluster:**
   - Choose the number of clusters(K) and obtain the data points.
   - Place the centroids c1, c2, ... ck randomly.
   - Repeat steps 4 and 5 until convergence or until the end of a fixed number of iterations
   - for each data point xi:
     - find the nearest centroid(c1, c2 .. ck)
     - assign the point to that cluster
   - for each cluster j = 1..k
     - new centroid = mean of all points assigned to that cluster
   - End
2. **k-medoids Cluster:**
   - Initialize: select k of the n data points as the medoids
   - Associate each data point to the closest medoid.
   - While the cost of the configuration decreases:
     - For each medoid m, for each non-medoid data point o:
       - Swap m and o, associate each data point to the closest medoid, recompute the cost (sum of distances of points to their medoid)

        * If the total cost of the configuration increased in the previous step, undo the swap
    - End

3. **Hierarchical Cluster:** Let X = x1, x2, x3, ..., xn be the set of data points.
    - Begin with the disjoint clustering having level L(0) = 0 and sequence number m = 0.
    - Find the least distance pair of clusters in the current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.
    - Increment the sequence number: m = m +1.Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to L(m) = d[(r),(s)].
    - Update the distance matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The distance between the new cluster, denoted (r,s) and old cluster(k) is defined in this way: d[(k), (r,s)] = min (d[(k),(r)], d[(k),(s)]).
    - If all the data points are in one cluster then stop, else repeat from step 2.

Code

```
#Installing and loading required R packages
#-----------------------------------------
#We'll use mainly two R packages:
#cluster package: for computing clustering
#factoextra package : for elegant ggplot2-based data visualization.
#---------------------

#Install:
#--------

#install.packages("factoextra")
#install.packages("cluster")
#install.packages("magrittr")
#install.packages("NbClust")

#Load packages:
#--------------

library("cluster")
library("factoextra")
library("magrittr")

#Data preparation
#----------------
#Demo data set: the built-in R dataset named USArrest
#Remove missing data
#Scale variables to make them comparable
#---------------------------------

# Load  and prepare the data
data("USArrests")

my_data <- USArrests %>%
  na.omit() %>%          # Remove missing values (NA)
  scale()                # Scale variables

# View the firt 3 rows
head(my_data, n = 3)

#Partitioning clustering
```

```
#------------------------
#Partitioning algorithms are clustering techniques that subdivide the data sets into a set
#K-means clustering
#K-medoids clustering or PAM (Partitioning Around Medoids)
#------------------------

fviz_nbclust(my_data, kmeans, method = "gap_stat")

#Compute and visualize k-means clustering
#------------------------------

set.seed(123)
km.res <- kmeans(my_data, 3, nstart = 25)
# Visualize
fviz_cluster(km.res, data = my_data, ellipse.type = "convex", palette = "jco", ggtheme = t


#Compute and visualize k-medoids clustering
#------------------------------

# Compute PAM
library("cluster")
pam.res <- pam(my_data, 3)
# Visualize
fviz_cluster(pam.res)

#Hierarchical clustering
#Hierarchical clustering is an alternative approach to partitioning clustering for identify
#----------
#compute and visualize hierarchical clustering:
#----------
# Compute hierarchical clustering
res.hc <- USArrests %>%
  scale() %>%                    # Scale the data
  dist(method = "euclidean") %>% # Compute dissimilarity matrix
  hclust(method = "ward.D2")     # Compute hierachical clustering

# Visualize using factoextra
# Cut in 4 groups and color by groups
fviz_dend(res.hc, k = 4, # Cut in four groups
          cex = 0.5, # label size
          k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
          color_labels_by_k = TRUE, # color labels by groups
          rect = TRUE # Add rectangle around groups
)

#Determining the optimal number of clusters
#--------------------


set.seed(123)

# Compute
library("NbClust")
res.nbclust <- USArrests %>%
  scale() %>%
  NbClust(distance = "euclidean",
          min.nc = 2, max.nc = 10,
```

```
            method = "complete", index ="all")

# Visualize
library(factoextra)
fviz_nbclust(res.nbclust, ggtheme = theme_minimal())
```

Output

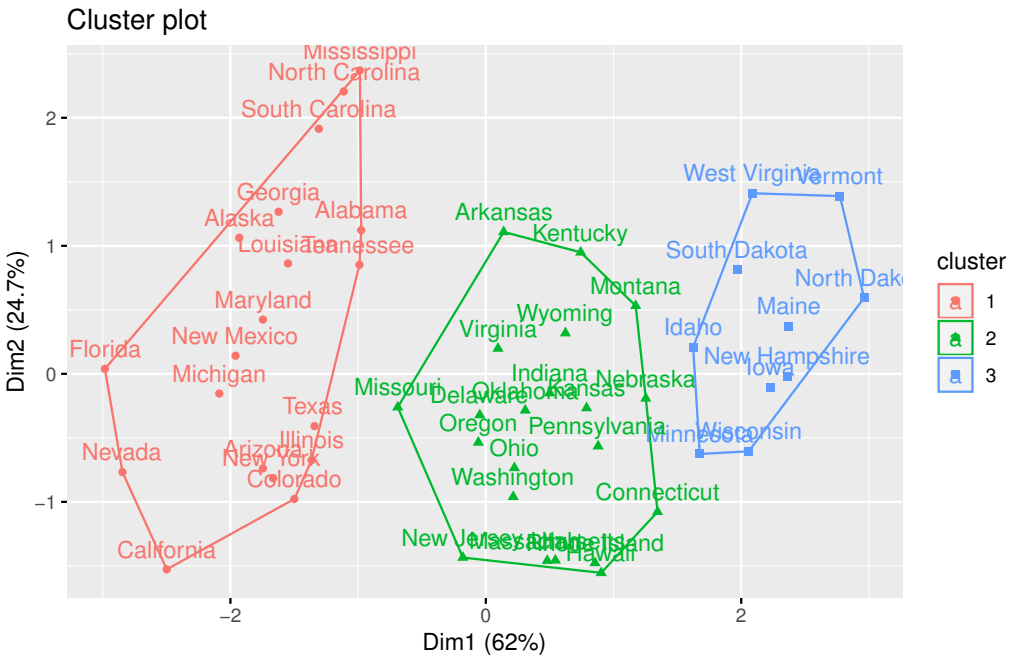

Figure 2: k-means

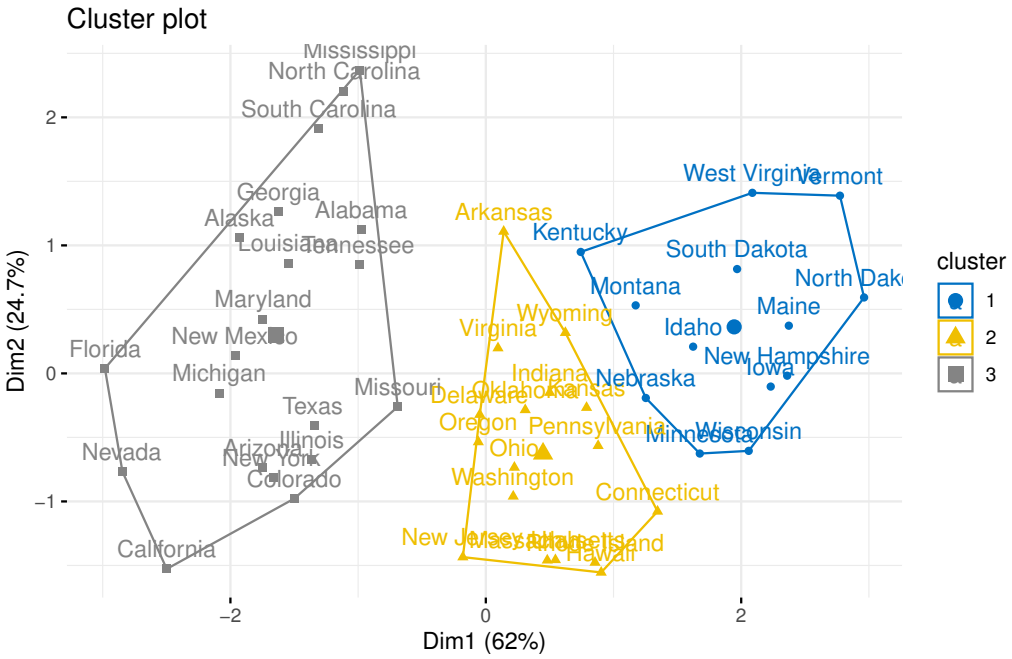

Figure 3: k-medoids

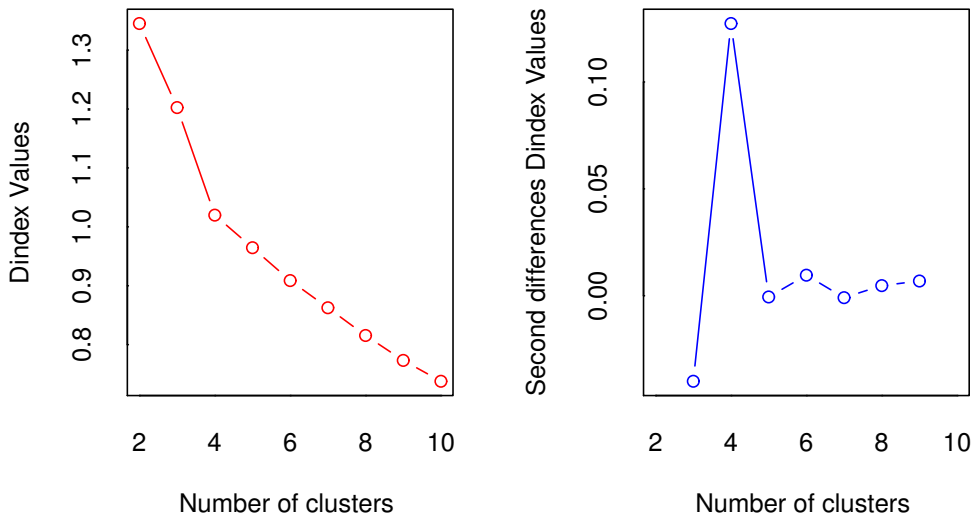

Figure 4: cluster

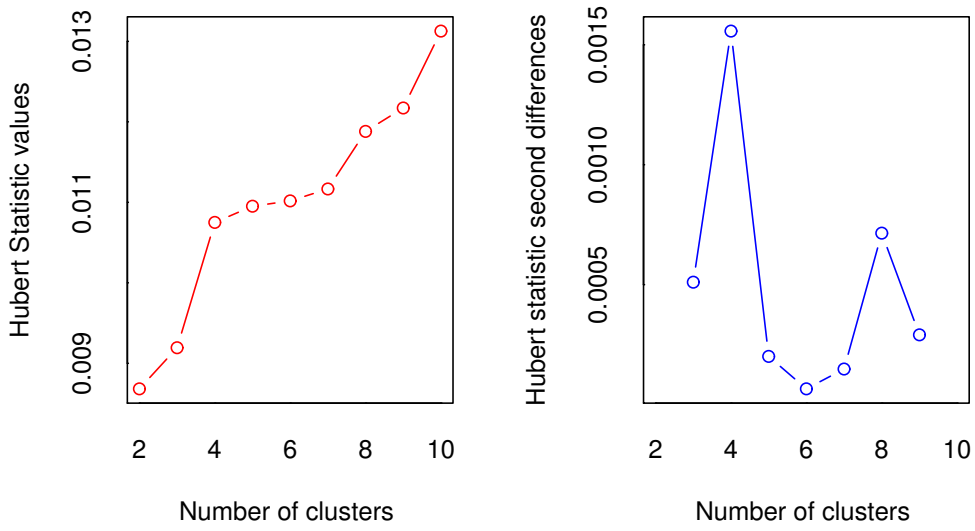

Figure 5: cluster

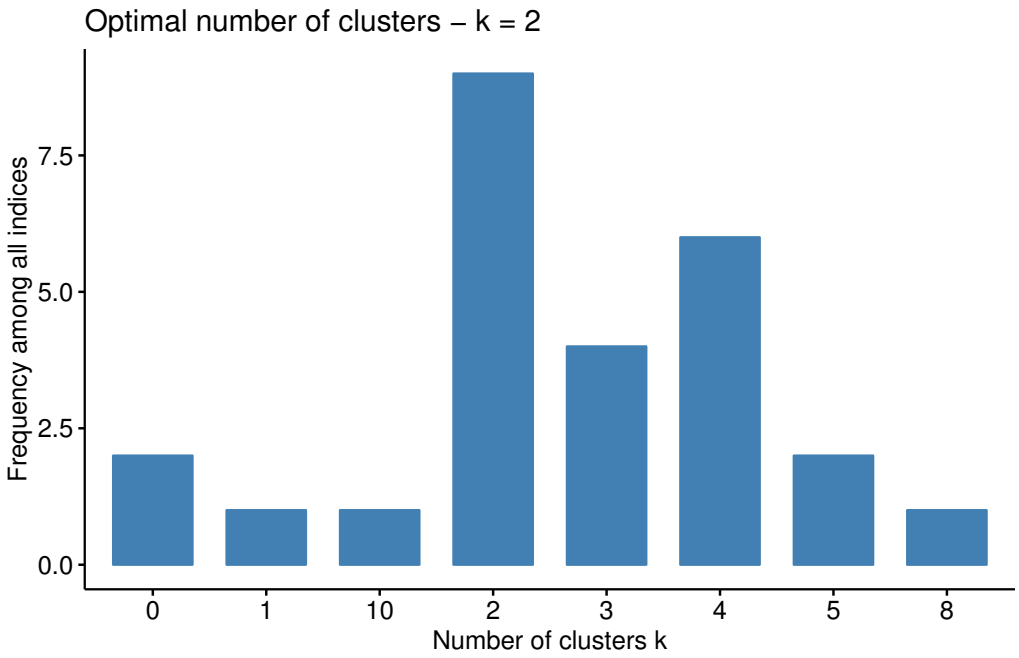

Figure 6: k-2 cluster

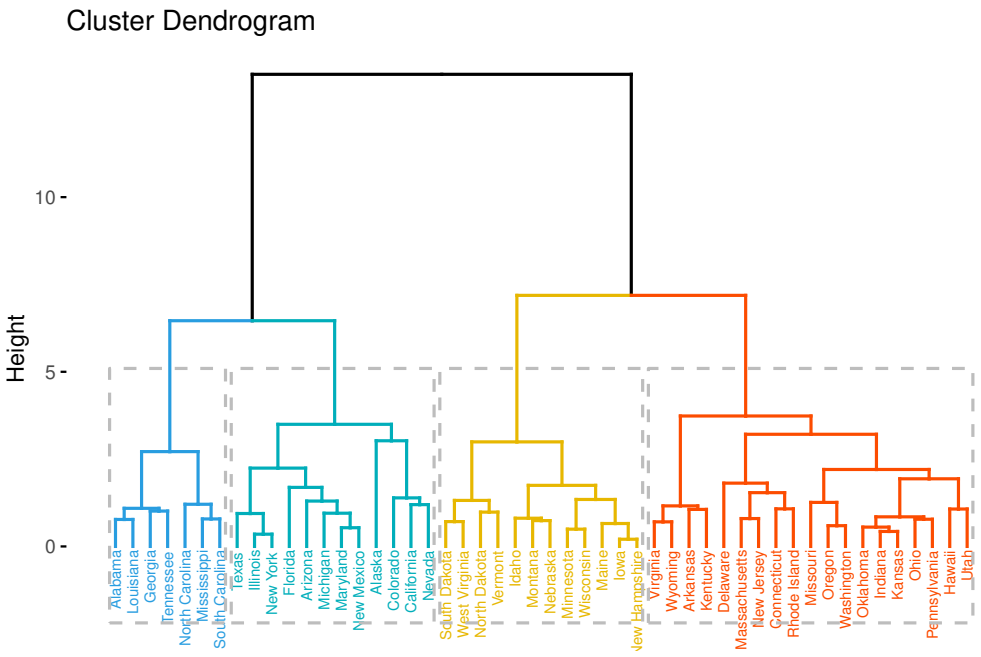

Figure 7: dendrogram