**Agglomerative Clustering**

The agglomerative clustering is the most common type of hierarchical clustering use to group objects in clusters based on their similarity. It’s also known as AGNES (Agglomerative Nesting). The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects. The result is a tree-based representation of the objects, named dendrogram. Agglomerative clustering works in a “bottom-up” manner.

**Steps to agglomerative hierarchical clustering**

We’ll follow the steps below to perform agglomerative hierarchical clustering using R software:

1. Preparing the data

2. Computing (dis)similarity information between every pair of objects in the data set.

3. Using linkage function to group objects into hierarchical cluster tree, based on the distance information generated at step 1. Objects/clusters that are in close proximity are linked together using the linkage function.

4. Determining where to cut the hierarchical tree into clusters. This creates a partition of the data.

**Data structure and preparation**

> data("USArrests")

> df = scale(USArrests)

> head(df, nrow = 6)

Murder Assault UrbanPop Rape

Alabama 1.24256408 0.7828393 -0.5209066 -0.003416473

Alaska 0.50786248 1.1068225 -1.2117642 2.484202941

Arizona 0.07163341 1.4788032 0.9989801 1.042878388

Arkansas 0.23234938 0.2308680 -1.0735927 -0.184916602

California 0.27826823 1.2628144 1.7589234 2.067820292

Colorado 0.02571456 0.3988593 0.8608085 1.86496720

**2 Similarity/distance measures**

> res.dist = dist(df,method = "euclidean")

> as.matrix(res.dist)[1:6, 1:6]

Alabama Alaska Arizona Arkansas California Colorado

Alabama 0.000000 2.703754 2.293520 1.289810 3.263110 2.651067

Alaska 2.703754 0.000000 2.700643 2.826039 3.012541 2.326519

Arizona 2.293520 2.700643 0.000000 2.717758 1.310484 1.365031

Arkansas 1.289810 2.826039 2.717758 0.000000 3.763641 2.831051

California 3.263110 3.012541 1.310484 3.763641 0.000000 1.287619

Colorado 2.651067 2.326519 1.365031 2.831051 1.287619 0.000000

**3 Linkage**

**• method:**

The agglomeration (linkage) method to be used for computing distance between clusters. Allowed values is one of **“ward.D”, “ward.D2”, “single”, “complete”, “average”, “mcquitty”, “median” or “centroid”.** There are many cluster agglomeration methods (i.e, linkage methods). The most common linkage methods are described below.

• **Maximum or complete linkage:** The distance between two clusters is defined as the maximum value of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. It tends to produce more compact clusters.

**• Minimum or single linkage:** The distance between two clusters is defined as the minimum value of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. It tends to produce long, "loose" clusters.

**• Mean or average linkage:** The distance between two clusters is defined as the average distance between the elements in cluster 1 and the elements in cluster 2.

**• Centroid linkage:** The distance between two clusters is defined as the distance between the centroid for cluster 1 (a mean vector of length p variables) and the centroid for cluster 2.

**• Ward’s minimum variance method:** It minimizes the total within-cluster variance. At each step the pair of clusters with minimum between-cluster distance are merged.

# hclust() is used to create the hierarchical tree

> res.hc = hclust(d = res.dist, method = "ward.D")

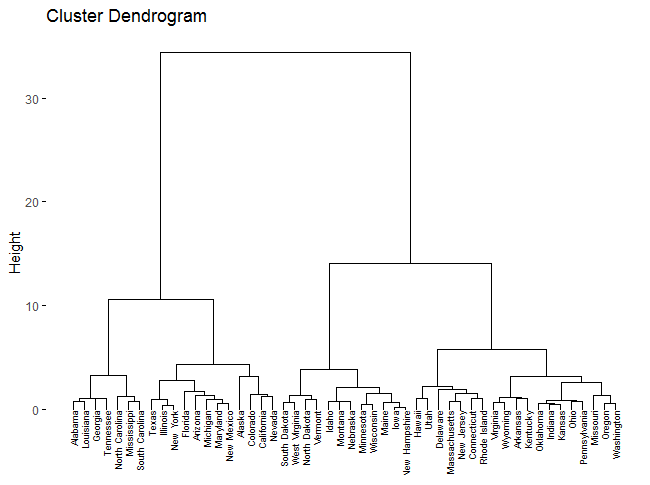
**4 Dendrogram**

Dendrograms correspond to the graphical representation of the hierarchical tree generated by the function hclust().

First install factoextra by typing this: install.packages(“factoextra”)

> library(factoextra)

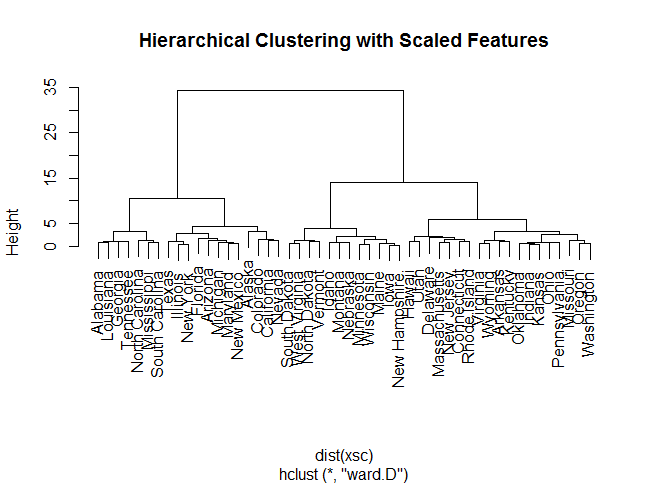
> fviz\_dend(res.hc, cex = 0.5) # cex size of labels



# Before performing hierarchical clustering of the observations, we use the scale() function

> xsc=scale (USArrests)

> plot(hclust (dist(xsc), method ="ward.D"), main =" Hierarchical Clustering with Scaled Features")



**3 Verify the cluster tree**

After linking the objects in a data set into a hierarchical cluster tree, you might want to assess that the distances

(i.e., heights) in the tree reflect the original distances accurately.

One way to measure how well the cluster tree generated by the hclust() function reflects your data is to compute

the correlation between the cophenetic distances and the original distance data generated by the dist() function. If the clustering is valid, the linking of objects in the cluster tree should have a strong correlation with the distances between objects in the original distance matrix.

The closer the value of the correlation coecient is to 1, the more accurately the clustering solution reflects your

data. Values above 0.75 are felt to be good. The “average” linkage method appears to produce high values of this statistic. This may be one reason that it is so popular.

The R base function cophenetic() can be used to compute the cophenetic distances for hierarchical clustering.

The **cophenetic** similarity or **cophenetic distance** of two objects is a measure of how similar

those two objects have to be in order to be grouped into the same cluster.

# Compute cophentic distance

> res.coph <- cophenetic(res.hc)

# Correlation between cophenetic distance and

# the original distance

> cor(res.dist, res.coph)

[1] 0.6844016

# Different method

> res.hc2 = hclust(d = res.dist, method = "average")

> res.coph <- cophenetic(res.hc2)

> cor(res.dist, res.coph)

[1] 0.7180382

> res.hc2 = hclust(d = res.dist, method = "complete")

> cor(res.dist, cophenetic(res.hc2))

[1] 0.6979437

> res.hc2 = hclust(d = res.dist, method = "single")

> cor(res.dist, cophenetic(res.hc2))

[1] 0.541272

> res.hc2 = hclust(d = res.dist, method = "centroid")

> cor(res.dist, cophenetic(res.hc2))

[1] 0.6074717

The correlation coefficient shows that using a different linkage method creates a tree that represents the original distances slightly better.

**4 Cut the dendrogram into different groups**

One of the problems with hierarchical clustering is that, it does not tell us how many clusters there are, or where to cut the dendrogram to form clusters.

You can cut the hierarchical tree at a given height in order to partition your data into clusters. The R base function cutree() can be used to cut a tree, generated by the hclust() function, into several groups either by specifying the desired number of groups or the cut height. It returns a vector containing the cluster number of each observation.

# cut the tree into 4 groups

> grp = cutree(res.hc, k=4) #cut tree into 4 groups

> head(grp, n = 5)

Alabama Alaska Arizona Arkansas California

1 2 2 3 2

> table(grp) #Number of members in each cluster

grp

1 2 3 4

7 12 19 12

> rownames(df)[grp == 1] # Get the names for the members of cluster 1

[1] "Alabama" "Georgia" "Louisiana" "Mississippi"

[5] "North Carolina" "South Carolina" "Tennessee"

# Cut in 4 groups and color by groups

> fviz\_dend(res.hc, k = 4, # Cut in four groups

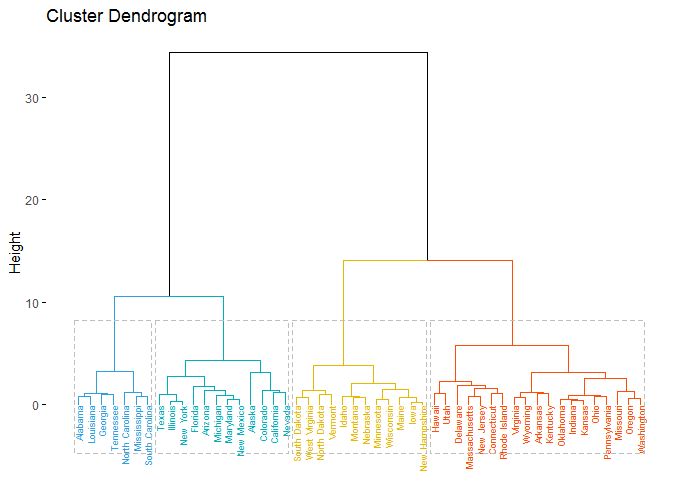
cex = 0.5, # label size

k\_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),

color\_labels\_by\_k = TRUE, # color labels by groups

rect = TRUE # Add rectangle around groups

)



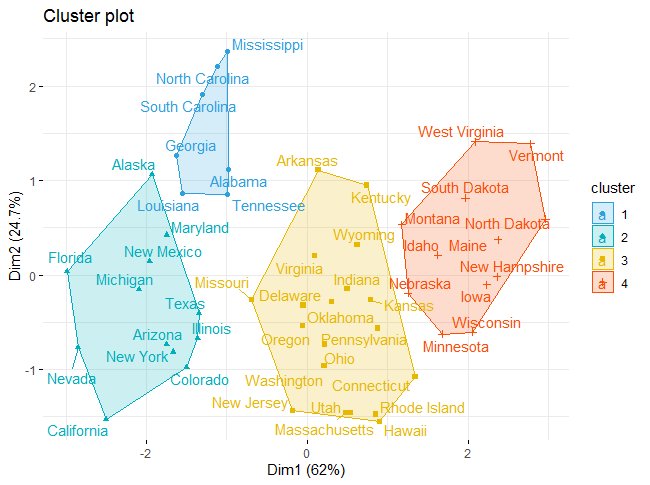
> fviz\_cluster(list(data = df, cluster = grp),

+ palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),

+ ellipse.type = "convex", # Concentration ellipse

+ repel = TRUE, # Avoid label overplotting (slow)

+ show.clust.cent = FALSE, ggtheme = theme\_minimal())



> library("cluster")

# Agglomerative Nesting (Hierarchical Clustering)

> res.agnes <- agnes(x = USArrests, # data matrix

+ stand = TRUE, # Standardize the data

+ metric = "euclidean", # metric for distance matrix

+ method = "ward" # Linkage method

+ )

> fviz\_dend(res.agnes, cex = 0.6, k = 4)

