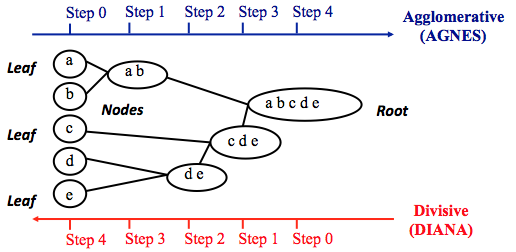
Agglomerative Clustering

Agglomerative clustering works in a “bottom-up” manner. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster (root) (see figure below).



The inverse of agglomerative clustering is divisive clustering, which is also known as DIANA (Divise Analysis) and it works in a “top-down” manner. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two.

Note that, agglomerative clustering is good at identifying small clusters. Divisive clustering is good at identifying large clusters.

**Steps to agglomerative hierarchical clustering**

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**

The data should be a numeric matrix with:

* rows representing observations (individuals);
* and columns representing variables.

**Use the R base USArrests data sets.**

**dist()** computes the distance between the rows of a data matrix using the specified distance measure method.

In R software, you can use the function dist() to compute the distance between every pair of object in a data set. The results of this computation is known as a distance or dissimilarity matrix.

### Linkage

The linkage function takes the distance information, returned by the function dist(), and groups pairs of objects into clusters based on their similarity.

Next, these newly formed clusters are linked to each other to create bigger clusters. This process is iterated until all the objects in the original data set are linked together in a hierarchical tree.

**hclust()** can be used to create the hierarchical tree.

hclust() can be used as follow:

res.hc <- hclust(d = res.dist, method = "ward.D2")

* **d**: a dissimilarity structure as produced by the **dist()** function.
* **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.

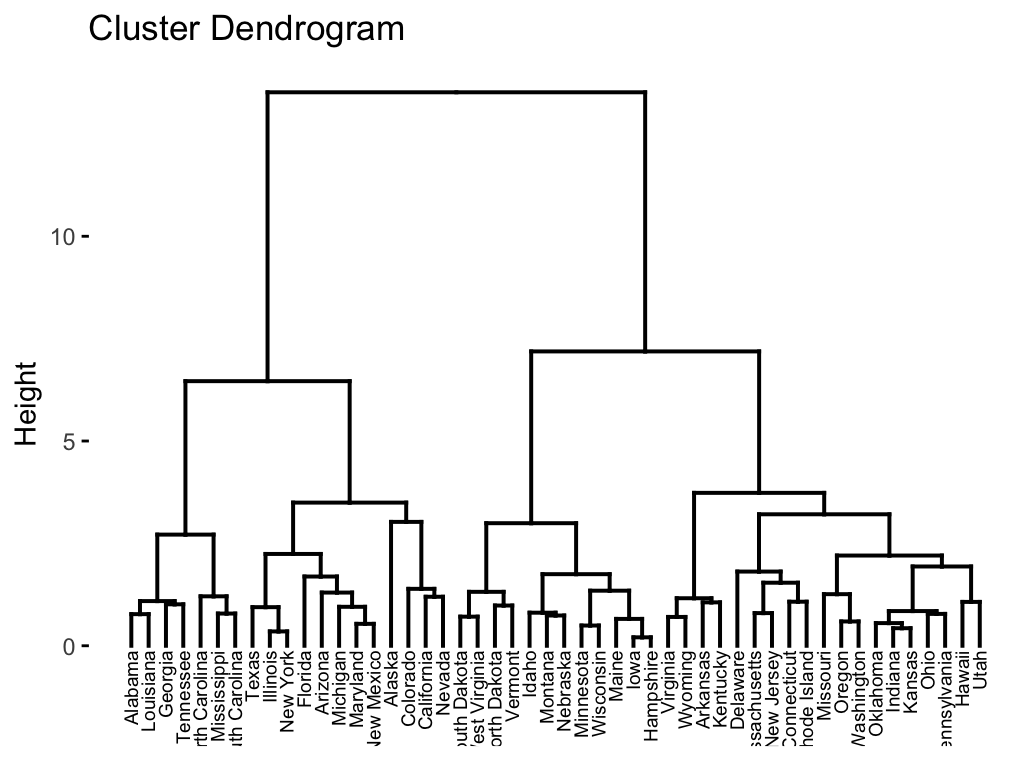
Note that, at each stage of the clustering process the two clusters, that have the smallest linkage distance, are linked together.

### **Dendrogram**

Dendrograms correspond to the graphical representation of the hierarchical tree generated by the function hclust(). Dendrogram can be produced in R using the base function plot(res.hc), where res.hc is the output of hclust(). Here, we’ll use the function fviz\_dend()[ in factoextra R package] to produce a dendrogram

**Package for dendrogram**

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



In the dendrogram displayed above, each leaf corresponds to one object. As we move up the tree, objects that are similar to each other are combined into branches, which are themselves fused at a higher height.

The height of the fusion, provided on the vertical axis, indicates the (dis)similarity/distance between two objects/clusters. The higher the height of the fusion, the less similar the objects are. This height is known as the cophenetic distance between the two objects.

**Verifying 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 coefficient 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 distance** between two objects is the height of the dendrogram where the two branches that include the two objects merge into a single branch.

## Cluster R package

The R package cluster makes it easy to perform cluster analysis in R. It provides the function agnes() and diana() for computing agglomerative and divisive clustering, respectively. These functions perform all the necessary steps for you. You don’t need to execute the scale(), dist() and hclust() function separately.

The functions can be executed as follow:

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

)

# DIvisive ANAlysis Clustering

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

stand = TRUE, # standardize the data

metric = "euclidean" # metric for distance matrix

**After running agnes() and diana(), you can use the function fviz\_dend()[in factoextra] to visualize the output:** )