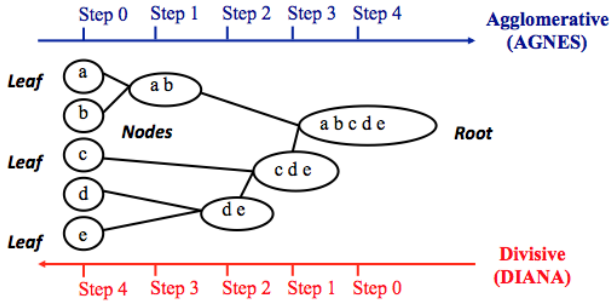
the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster (see figure below).

Note that, agglomerative clustering is good at identifying small clusters. Divisive clustering is good at identifying large clusters. In this article, we'll focus mainly on agglomerative hierarchical clustering.



7.2 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.

We'll describe each of these steps in the next section.

7.2.1 Data structure and preparation

The data should be a numeric matrix with:

- rows representing observations (individuals);

- and columns representing variables.

Here, we'll use the R base USArrests data sets.

Note that, it's generally recommended to standardize variables in the data set before performing subsequent analysis. Standardization makes variables comparable, when they are measured in different scales. For example one variable can measure the height in meter and another variable can measure the weight in kg. The R function *scale*() can be used for standardization, See ?scale documentation.

```
# Load the data
data("USArrests")

# Standardize the data
df <- scale(USArrests)

# Show the first 6 rows
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  1.864967207
```

7.2.2 Similarity measures

In order to decide which objects/clusters should be combined or divided, we need methods for measuring the similarity between objects.

There are many methods to calculate the (dis)similarity information, including Euclidean and manhattan distances (Chapter 3). 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.

By default, the function *dist*() computes the Euclidean distance between objects; however, it's possible to indicate other metrics using the argument method. See ?dist

for more information.

For example, consider the R base data set USArrests, you can compute the distance matrix as follow:

```
# Compute the dissimilarity matrix
# df = the standardized data
res.dist <- dist(df, method = "euclidean")
```

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

To see easily the distance information between objects, we reformat the results of the function *dist*() into a matrix using the *as.matrix*() function. In this matrix, value in the cell formed by the row i, the column j, represents the distance between object i and object j in the original data set. For instance, element 1,1 represents the distance between object 1 and itself (which is zero). Element 1,2 represents the distance between object 1 and object 2, and so on.

The R code below displays the first 6 rows and columns of the distance matrix:

```
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
```

7.2.3 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.

For example, given a distance matrix "res.dist" generated by the function *dist*(), the R base function *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.

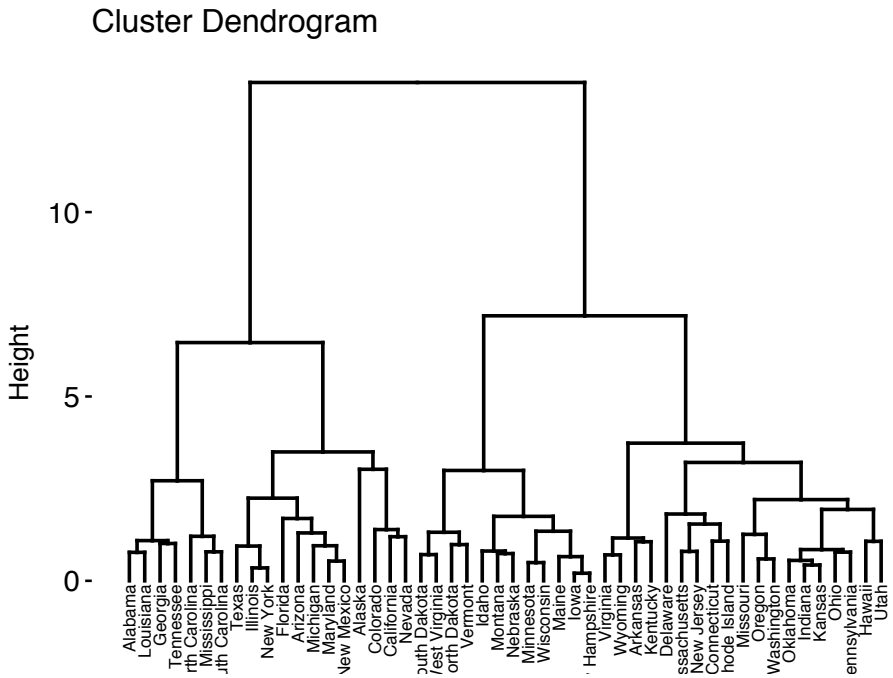
Complete linkage and Ward's method are generally preferred.

7.2.4 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 beautiful dendrogram.

First install factoextra by typing this: install.packages("factoextra"); next visualize the dendrogram as follow:

```
# cex: label size
library("factoextra")
fviz_dend(res.hc, cex = 0.5)
```



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.

> Note that, conclusions about the proximity of two objects can be drawn only based on the height where branches containing those two objects first are fused. We cannot use the proximity of two objects along the horizontal axis as a criteria of their similarity.

In order to identify sub-groups, we can cut the dendrogram at a certain height as described in the next sections.

7.3 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 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.

```
# Compute cophentic distance
res.coph <- cophenetic(res.hc)

# Correlation between cophenetic distance and
# the original distance
cor(res.dist, res.coph)
```

```
## [1] 0.6975266
```

Execute the *hclust*() function again using the average linkage method. Next, call *cophenetic*() to evaluate the clustering solution.

```
res.hc2 <- hclust(res.dist, method = "average")

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

```
## [1] 0.7180382
```

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

7.4 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 tree into 4 groups
grp <- cutree(res.hc, k = 4)
head(grp, n = 4)
```

```
##  Alabama   Alaska  Arizona Arkansas
##        1        2        2        3
```

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

```
## grp
##  1  2  3  4
##  7 12 19 12
```

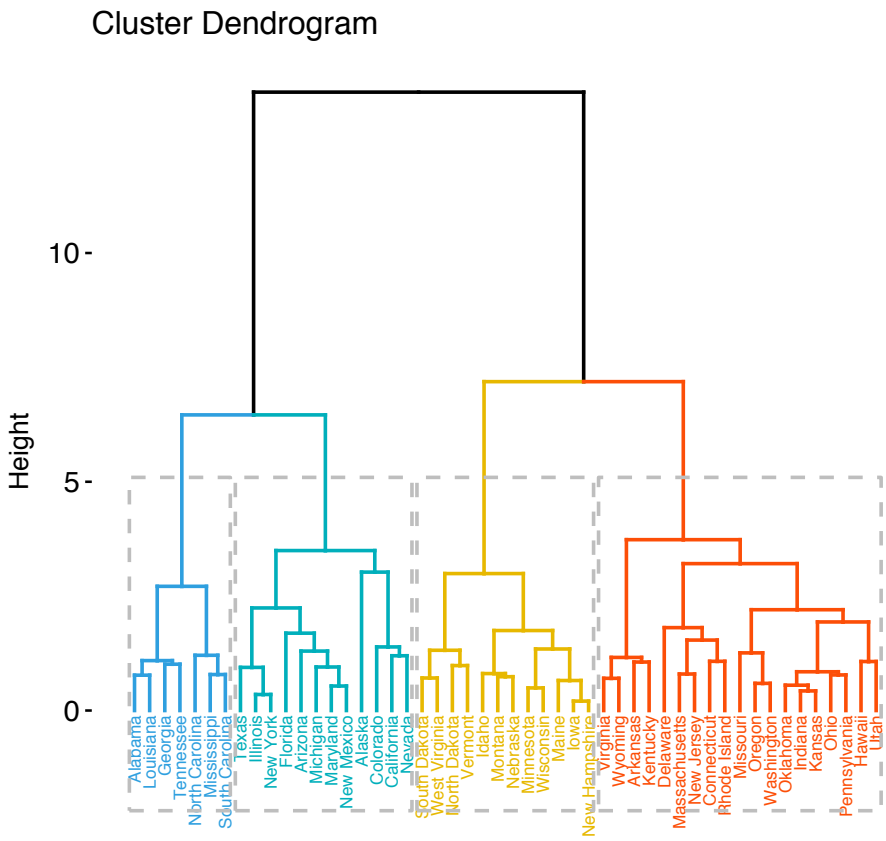
```
# Get the names for the members of cluster 1
rownames(df)[grp == 1]
```

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

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

The result of the cuts can be visualized easily using the function *fviz_dend*() [in 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
          )
```



Using the function *fviz_cluster*() [in *factoextra*], we can also visualize the result in a scatter plot. Observations are represented by points in the plot, using principal components. A frame is drawn around each cluster.