How to do Hierarchical Clustering in R: Step-by-Step Example

Authored by

stats writer

December 17, 2025

Hierarchical clustering is a powerful technique in cluster analysis that constructs a hierarchy of nested clusters based on their similarity. Unlike partition-based methods, the process involves iteratively combining or dividing observations to build a structure that visually represents the relationships between data points. In the R programming environment, implementing Hierarchical Clustering involves several key steps: assembling a numerical data matrix, calculating the distance matrix, using the `hclust()` function to generate the clustering tree (or Dendrogram), and finally, employing `cutree()` to define the desired number of clusters. Visualization and interpretation are typically achieved using `plot()` and `table()`, respectively.

Clustering is a foundational technique in machine learning aimed at partitioning a dataset into groups, or **clusters**. The primary goal is maximizing the internal similarity of observations within any given cluster while simultaneously maximizing the dissimilarity between observations belonging to different clusters. This approach is fundamental for discovering inherent structures within data without prior labeled examples.

Specifically, clustering is categorized as a form of Unsupervised Learning. Since we are not attempting to predict a specific outcome or response variable, the focus shifts entirely to identifying patterns and inherent organization within the multivariate data structure.

A common and highly practical application of clustering lies in marketing and customer segmentation, where companies often analyze detailed household metrics such as:

Household income and purchasing power statistics.
Household size and demographic composition metrics.
Head of household **Occupation** or employment sector data.
Distance from nearest urban area, indicating lifestyle or accessibility.

By utilizing these variables, clustering allows analysts to identify groups of similar households. This segmentation is invaluable for customizing marketing strategies, ensuring that specific products or advertising campaigns are targeted effectively to the segments most likely to respond positively.

While one of the most widely used clustering methods is K-means clustering, it presents a significant drawback: it requires the user to pre-specify the number of clusters, *K*, a requirement that is often challenging to satisfy without extensive prior domain knowledge.

An advantageous alternative that circumvents this limitation is **hierarchical clustering**. This robust method does not necessitate pre-specifying the cluster count. Furthermore, it inherently generates a tree-based graphical representation of the observations, known as a Dendrogram, which aids significantly in the visualization and interpretation of the data structure.

What is Hierarchical Clustering?

Similar to other partitioning methods like K-means, the fundamental objective of Hierarchical Clustering is to create groups of observations where the elements within a group are highly homogeneous (similar to each other), and the groups themselves are highly heterogeneous (dissimilar from one another). The primary difference lies in the process: HC builds a nested sequence of partitions, ranging from single-observation clusters up to one large encompassing cluster.

In practical data analysis, the implementation of hierarchical clustering typically follows a two-step iterative procedure, usually known as agglomerative clustering (bottom-up approach):

1. Calculate the Pairwise Dissimilarity between Observations.

The process begins by defining an appropriate distance metric. A commonly utilized metric is the Euclidean Distance, which is then used to compute the dissimilarity (or distance) between every pair of observations in the dataset.
For a dataset comprising *n* observations, this step results in a total of exactly $n(n-1)/2$ unique pairwise dissimilarity measures, forming a comprehensive distance matrix.

2. Iteratively Fuse Observations into Clusters.

In the subsequent step, the algorithm repeatedly merges the two observations or clusters that exhibit the highest degree of similarity (i.e., the minimum dissimilarity) into a single, cohesive cluster.
This aggregation procedure continues until all individual observations have been merged into one colossal cluster. The entire sequence of fusions generates a hierarchical structure that is classically represented as a Dendrogram.

To properly determine how close two existing clusters are--a necessary step for the fusion process--we must employ a specific linkage method. These methods define the distance between two groups of observations based on the distances between the individual members of those groups:

Complete Linkage Clustering: Measures the distance between two clusters as the **maximum distance** observed between any two points, one belonging to each cluster. This method tends to produce more compact, spherical clusters.
Single Linkage Clustering: Defined by the **minimum distance** between points belonging to two different clusters. This method is susceptible to "chaining" and tends to produce long, loose clusters.
Mean Linkage Clustering: Calculates the distance by averaging all pairwise distances between

points in the two clusters being compared.
Centroid Linkage Clustering: Determines the distance by first finding the geometric center (centroid) of each cluster and then calculating the distance between these two centroids.
Ward's Minimum Variance Method: A highly popular method that minimizes the total within-cluster variance. It seeks to fuse the pair of clusters that results in the minimum increase in the total within-cluster sum of squares.

The selection of the appropriate linkage method is critical, as it heavily influences the shape and quality of the resulting clusters. Depending on the underlying structure of the dataset, one method--such as Ward's--might yield significantly better (i.e., more compact and meaningful) clusters than the others.

Hierarchical Clustering in R: A Practical Example

The following comprehensive tutorial provides a step-by-step practical guide on how to efficiently perform Hierarchical Clustering using the statistical programming language R, demonstrating best practices for data preparation, cluster determination, and result interpretation.

Step 1: Load the Necessary Packages

To begin our analysis, we must first load the required software libraries within the R environment. We will rely specifically on two packages that offer specialized functions for distance calculation and hierarchical clustering visualization: `factoextra` and `cluster`.

```
library(factoextra)
library(cluster)
```

Step 2: Load and Prep the Data

For this hands-on example, we will utilize the intrinsic `USArrests` dataset provided within the R base installation. This dataset contains critical socio-criminal statistics from 1973 for each U.S. state, specifically measuring the number of arrests per 100,000 residents across three categories: **Murder**, **Assault**, and **Rape**. Additionally, it includes the percentage of the state's population residing in urban areas, denoted as **UrbanPop**.

Data preparation is a mandatory step in clustering to ensure that all variables contribute equally to the distance calculation. The following sequence of code illustrates the required preprocessing steps:

Load the standard `USArrests` dataset into a new dataframe, `df`.
Explicitly remove any rows that contain missing (`NA`) values to ensure data integrity.

Critically, **scale** each variable in the dataset. This standardization process transforms the data such that every variable has a mean of 0 and a standard deviation of 1, preventing variables with larger magnitudes (like Assault counts) from dominating the clustering process.

#load data
df <- USArrests

#remove rows with missing values
df <- na.omit(df)

#scale each variable to have a mean of 0 and sd of 1
df <- scale(df)

#view first six rows of dataset to confirm scaling
head(df)

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

Step 3: Find the Optimal Linkage Method

To execute hierarchical clustering in R, we will leverage the `agnes()` function (Agglomerative Nesting) from the `cluster` package. This function is essential for performing bottom-up clustering and utilizes the following concise syntax structure:

```
agnes(data, method)
```

Here, `data` specifies the dataset (our scaled `df`), and `method` dictates the desired linkage technique used to quantify the dissimilarity between emerging clusters.

data: The name of the input data matrix or dataframe containing the observations.
method: Defines the specific linkage algorithm (e.g., "average", "ward") used to calculate dissimilarity between groups.

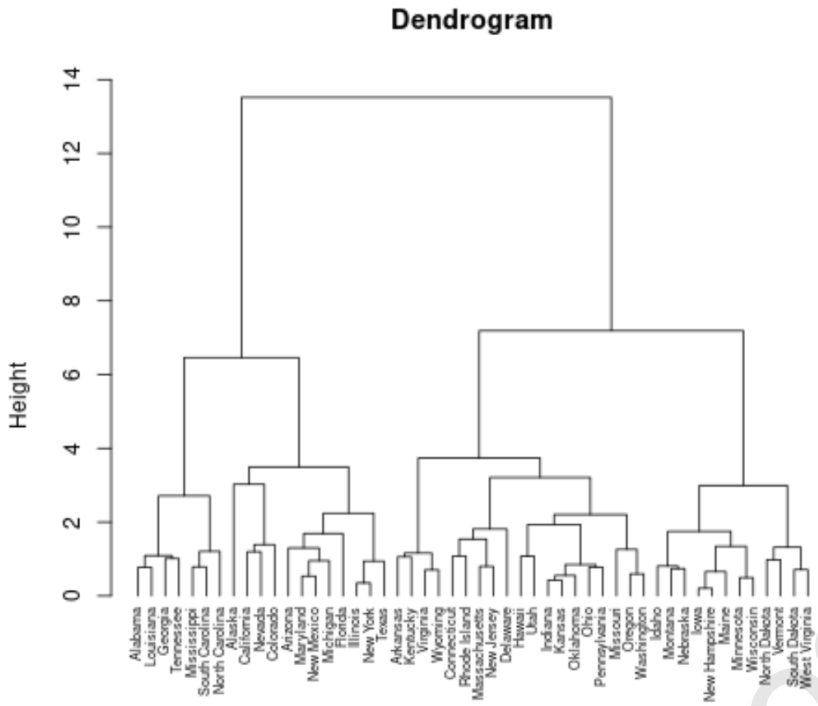
Since the optimal linkage method is typically unknown beforehand, we must test several common methods to determine which produces the strongest clustering structure. We achieve this by calculating the Agglomerative Coefficient (AC) for each method, a metric ranging from 0 to 1 that

quantifies the strength of the clustering structure. A value closer to 1 indicates a stronger, more definite cluster hierarchy.

```
#define linkage methods for testing
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")

#function to compute the Agglomerative Coefficient (AC)
ac <- function(x) {
agnes(df, method = x)$ac
}

#calculate AC for each clustering linkage method
sapply(m, ac)

average single complete ward
0.7379371 0.6276128 0.8531583 0.9346210
```

Upon reviewing the results, we observe that **Ward's minimum variance method** yields the highest Agglomerative Coefficient (0.9346). This robust finding suggests that Ward's method is the most suitable approach for structuring this specific dataset, and we will proceed by using it for our final hierarchical clustering model.

```
#perform hierarchical clustering using Ward's minimum variance
clust <- agnes(df, method = "ward")

#produce dendrogram visualization
pltree(clust, cex = 0.6, hang = -1, main = "Dendrogram")
```



The resulting Dendrogram graphically represents the clustering process. Each terminal leaf at the base corresponds to an original state (observation). As one traces paths upward, observations that exhibit high similarity are fused together into branches. The height of the fusion point indicates the level of dissimilarity (distance) at which the clusters were merged; lower heights signify greater similarity.

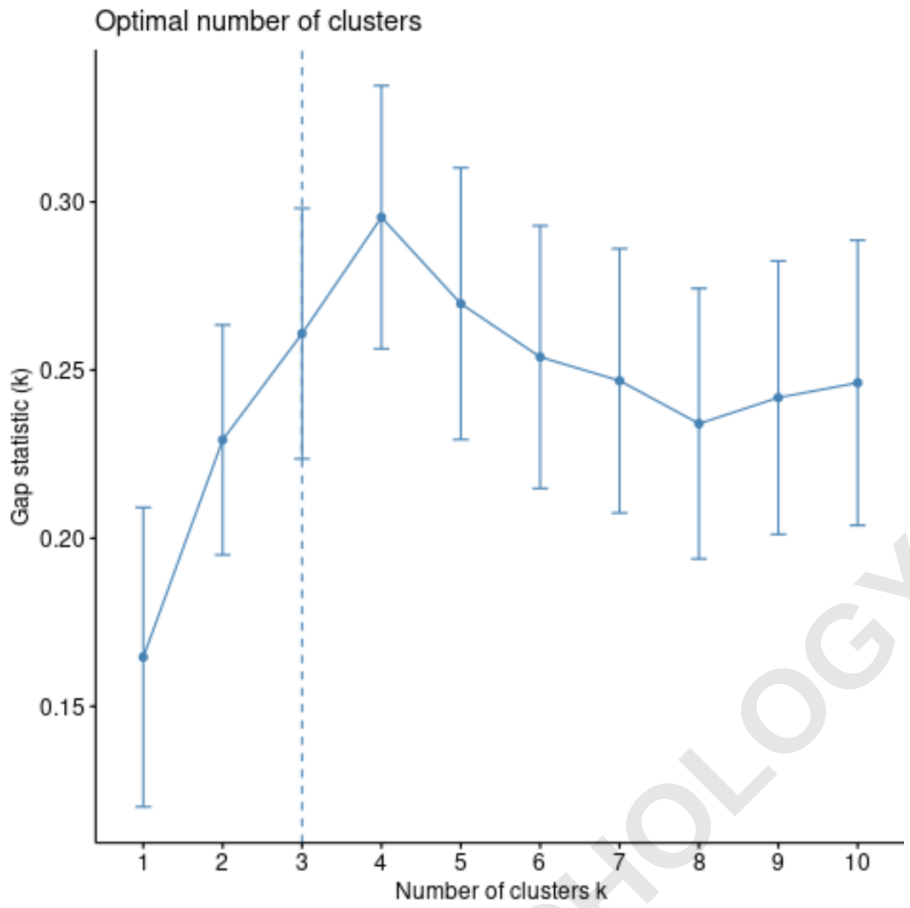
Step 4: Determine the Optimal Number of Clusters

A key challenge in clustering is defining where to "cut" the dendrogram to yield the most meaningful number of clusters, *k*. To address this objectively, we employ the Gap Statistic, a robust metric developed by Tibshirani, Walther, and Hastie. The Gap Statistic compares the total within-cluster variation (dispersion) for various values of *k* against the expected variation derived from a reference null distribution that lacks inherent clustering structure.

In R, we calculate the Gap Statistic using the `clusGap()` function from the `cluster` package. We then visualize the results using the `fviz_gap_stat()` function from `factoextra` to pinpoint the optimal value of *k*.

#calculate gap statistic for each number of clusters (up to 10 clusters)
gap_stat <- clusGap(df, FUN = hcut, nstart = 25, K.max = 10, B = 50)

#produce plot of clusters vs. gap statistic
fviz_gap_stat(gap_stat)



The resulting visualization of the Gap Statistic clearly indicates that the statistic reaches its maximum value when *k* = 4. Following the rule of maximizing the Gap Statistic, we confidently choose to partition our observations into **four distinct clusters**.

Step 5: Apply Cluster Labels and Analyze Results

With the optimal number of clusters determined (k=4), the final step is to assign these cluster labels back to the original, unscaled dataset. We achieve this by using the `cutree()` function, which systematically cuts the hierarchical tree structure (the dendrogram) into the specified number of groups. Note that for `hclust()`, the standard Ward method used is `"ward.D2"`.

#compute distance matrix using Euclidean distance
d <- dist(df, method = "euclidean")

#perform hierarchical clustering using hclust and Ward's method
final_clust <- hclust(d, method = "ward.D2")

#cut the dendrogram into 4 clusters

groups <- cutree(final_clust, k=4)

#find number of observations in each cluster
table(groups)

1 2 3 4
7 12 19 12

The output confirms the distribution of the 50 US states across the four resulting clusters. We then proceed to append these cluster assignments back as a new column to the original `USArrests` dataframe, creating `final_data`:

#append cluster labels to original data
final_data <- cbind(USArrests, cluster = groups)

#display first six rows of final data, including the new cluster label
head(final_data)

Murder Assault UrbanPop Rape cluster
Alabama 13.2 236 58 21.2 1
Alaska 10.0 263 48 44.5 2
Arizona 8.1 294 80 31.0 2
Arkansas 8.8 190 50 19.5 3
California 9.0 276 91 40.6 2
Colorado 7.9 204 78 38.7 2

Interpreting Cluster Characteristics

To gain insight into the specific characteristics that define each group, we use the `aggregate()` function. This powerful function calculates the mean values for all original variables (Murder, Assault, UrbanPop, Rape) within each of the four identified clusters. This summary allows us to profile and label the segments based on their average crime and urbanization rates.

#find mean values for each cluster
aggregate(final_data, by=list(cluster=final_data$cluster), mean)

cluster Murder Assault UrbanPop Rape cluster
1 1 14.671429 251.2857 54.28571 21.68571 1
2 2 10.966667 264.0000 76.50000 33.60833 2
3 3 6.210526 142.0526 71.26316 19.18421 3
4 4 3.091667 76.0000 52.08333 11.83333 4

Based on the calculated means, we can interpret the profiling characteristics of Cluster 1 (the high-crime, moderate-urbanization segment) as follows:

The mean number of murders per 100,000 citizens among the states in cluster 1 is **14.67**.
The mean number of assaults per 100,000 citizens among the states in cluster 1 is **251.28**.
The mean percentage of residents living in an urban area among the states in cluster 1 is **54.28%**.
The mean number of rapes per 100,000 citizens among the states in cluster 1 is **21.68.**

A similar interpretation process can be applied to Clusters 2, 3, and 4 to reveal distinct crime and demographic profiles, thereby achieving effective data segmentation through hierarchical clustering.

You can find the complete R code used in this example here.