Learning Hierarchical Clustering with R: A Practical Guide

Authored by

Mohammed looti

November 6, 2025

Clustering is a fundamental technique in machine learning designed to group observations into meaningful segments, known as **clusters**. The core objective of this process is to ensure high internal coherence--that observations within a single cluster are highly similar to one another--while maintaining high external separation, meaning observations belonging to different clusters exhibit significant dissimilarity.

This methodology falls squarely under the category of unsupervised learning. Unlike supervised algorithms that are trained to predict a specific response variable, clustering focuses solely on autonomously discovering the inherent structure, underlying patterns, and natural groupings within the raw data.

A classic and highly practical application of clustering is in **market segmentation**, where organizations leverage various demographic and behavioral data points to identify distinct customer groups for targeted strategic planning. Examples of such critical data features often utilized include:

Household income demographics and wealth distribution
Household size and composition
Head of household occupation classification
Proximity or distance from the nearest major urban area

By rigorously analyzing this information, clustering helps pinpoint households that share similar characteristics, thereby making it possible to tailor marketing campaigns, accurately predict purchasing behavior, and optimize product placement strategies with precision.

While K-means clustering is frequently used, it suffers from a major prerequisite: the user must manually pre-specify the number of clusters, *K*. This limitation is restrictive, especially when the natural, underlying structure of the data is completely unknown or ambiguous.

An elegant and robust alternative that overcomes this requirement is hierarchical clustering (HAC). This method does not necessitate a pre-defined number of clusters. Instead, it generates a complete, tree-based visual representation of the observation relationships, known as a dendrogram, which provides powerful, interpretable insights into the data's structure.

Understanding Agglomerative Hierarchical Clustering (HAC)

Like its counterparts, the core objective of hierarchical clustering is to systematically group similar observations. HAC is an iterative process that builds a hierarchy of clusters. This hierarchy can be constructed using two primary approaches: the agglomerative (bottom-up) method, which merges smaller clusters, or the divisive (top-down) method, which splits larger ones. For practical implementation and simplicity, we will focus exclusively on the more common **agglomerative approach**.

The standard procedure for performing agglomerative hierarchical clustering involves two fundamental and sequential computational steps that must be addressed before the cluster structure is complete:

Calculate the Pairwise Dissimilarity Matrix. This step rigorously quantifies the distance between all data points using a chosen metric.
Iteratively Fuse Observations into Clusters. This step builds the final tree structure by continuously merging the closest clusters.

Step 1: Quantifying Dissimilarity and Distance

The initial phase demands the selection of an appropriate distance metric--most commonly the Euclidean distance--to rigorously quantify the **pairwise dissimilarity** between every single observation within the dataset. This metric determines how "far apart" or dissimilar any two points are in the feature space.

This calculation results in a comprehensive dissimilarity matrix. For a dataset containing n observations, this matrix is symmetrical and holds a total of $n(n\text{-}1)/2$ unique pairwise distance values.
The accurate measurement of initial dissimilarity is paramount, as it directly governs how observations are grouped and merged in all subsequent stages of the clustering process.

Step 2: Cluster Fusion and the Dendrogram

In the fusion phase, the algorithm begins by treating every single observation as its own individual cluster. In each subsequent iteration, the algorithm identifies the two existing clusters that are determined to be the most similar (i.e., those exhibiting the minimum calculated dissimilarity) and merges them into a single, cohesive new cluster.

This systematic merging process continues iteratively until all observations are ultimately consolidated into one large, overarching cluster containing the entire dataset. The final output of this hierarchical merging process is a structure that is most effectively visualized using a

dendrogram.

Selecting the Optimal Linkage Method

A fundamental decision in hierarchical clustering is defining how the distance (or dissimilarity) between two established clusters should be calculated when they contain multiple points. This computational rule is referred to as the linkage method, and the choice of method critically impacts the resulting shape, compactness, and quality of the final clusters.

The following list outlines the most frequently utilized linkage methods in data analysis:

Complete linkage clustering: Measures the maximum distance (farthest neighbor) between any two points belonging to the two different clusters being compared. This technique typically favors the production of tighter, more compact, and spherically shaped clusters.
Single linkage clustering: Determines the minimum distance (nearest neighbor) between points in the two clusters. This method is susceptible to the "chaining" effect, often resulting in elongated, loosely defined clusters.
Mean linkage clustering: Calculates the average of all possible pairwise distances between every point in the two respective clusters.
Centroid linkage clustering: Calculates the distance exclusively between the geometric center (centroid) of each cluster.
Ward's minimum variance method: This popular method minimizes the total within-cluster variance (sum of squared errors) when two clusters are merged. It is generally preferred for generating balanced and statistically distinct clusters.

Given that the effectiveness of any specific linkage method depends heavily on the intrinsic structure and geometry of the dataset, it is best practice for analysts to empirically evaluate several methods, often using metrics like the Agglomerative Coefficient, to identify the method that yields the most compact and interpretable cluster assignments.

Practical Implementation of Hierarchical Clustering in R

The remainder of this article provides a detailed, practical tutorial on executing hierarchical clustering using the powerful statistical programming environment, R. We will walk through the entire process, from initial data preparation and selection of the optimal parameters to the final interpretation of the results, utilizing standard R packages and the publicly available `USArrests` dataset.

Step 1: Load Essential R Packages

We initiate the process by loading the two necessary R packages: `factoextra`, which is invaluable

for streamlining the visualization and extraction of clustering results, and `cluster`, which contains the fundamental clustering functions we require.

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

Step 2: Data Loading and Preprocessing

For this analysis, we utilize the built-in `USArrests` dataset, which compiles statistics regarding arrests per 100,000 residents across the 50 U.S. states in 1973. The variables include rates for violent crimes (Murder, Assault, Rape) and the proportion of the population residing in urban areas (UrbanPop).

Effective data preparation is non-negotiable for ensuring clustering accuracy and validity. The following code snippet executes three essential preprocessing tasks before modeling:

Loading the `USArrests` dataset into a data frame.
Handling any potential missing values using `na.omit()` (a good general practice).
Critically, **Scaling** the variables. Scaling standardizes the features, ensuring that variables with naturally larger numerical ranges (like Assault rates) do not unfairly dominate the distance calculations. This process transforms the data so that each variable has a mean of 0 and a standard deviation of 1.

```
#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
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: Determining the Optimal Linkage Method

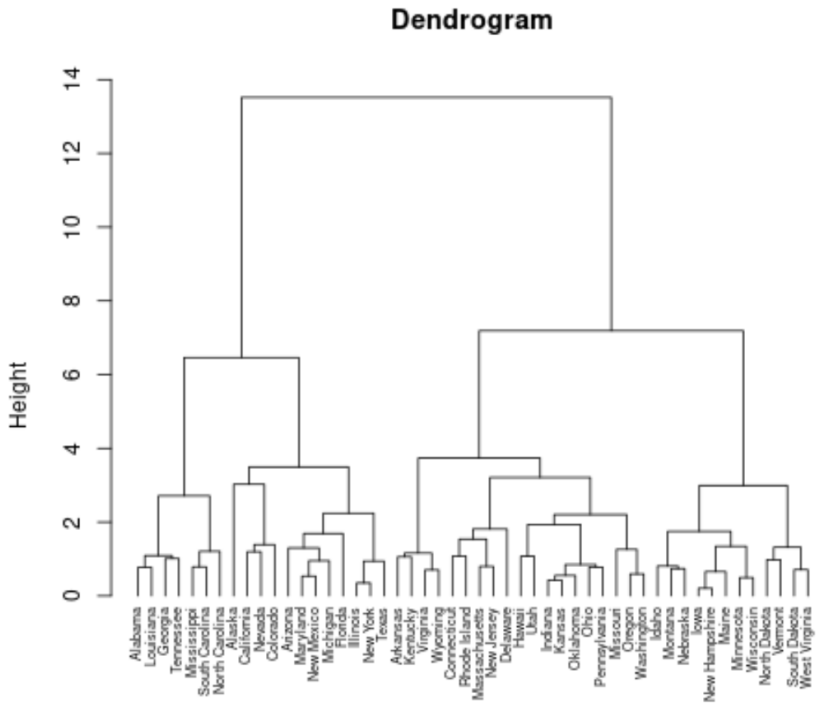
To execute the hierarchical clustering, we rely on the `agnes()` function from the `cluster` package. Since the most effective linkage method is application-dependent, we must quantitatively evaluate several popular options. The basic syntax, `agnes(data, method)`, allows us to specify the linkage criterion to test against our scaled data.

We will use the **Agglomerative Coefficient (AC)** as our primary evaluation metric. The AC quantifies the inherent strength of the clustering structure: values closer to 1 signify robust, well-separated, and highly distinct clusters. We define a helper function below to swiftly calculate the AC across four common linkage methods.

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

#function to compute agglomerative coefficient
ac <- function(x) {
agnes(df, method = x)$ac
}

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

average single complete ward
0.7379371 0.6276128 0.8531583 0.9346210
```

The results clearly demonstrate that **Ward's minimum variance method** achieves the highest Agglomerative Coefficient (0.9346). Consequently, Ward's method is selected for our final hierarchical clustering model. The resulting cluster hierarchy is then visualized as a dendrogram using the `pltree()` function, as shown below:

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

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



In this resulting **dendrogram**, each leaf at the bottom represents a unique state observation. Moving upward, observations are fused into branches based on similarity. The crucial aspect is the vertical height of the fusion point: a lower fusion height indicates greater similarity between the two merged clusters.

Step 4: Determining the Optimal Number of Clusters (k)

Although hierarchical clustering provides the full hierarchy, we must ultimately determine where to "cut" the dendrogram to extract a discrete number of groups, *k*. We determine this optimal value using a quantitative approach known as the gap statistic.

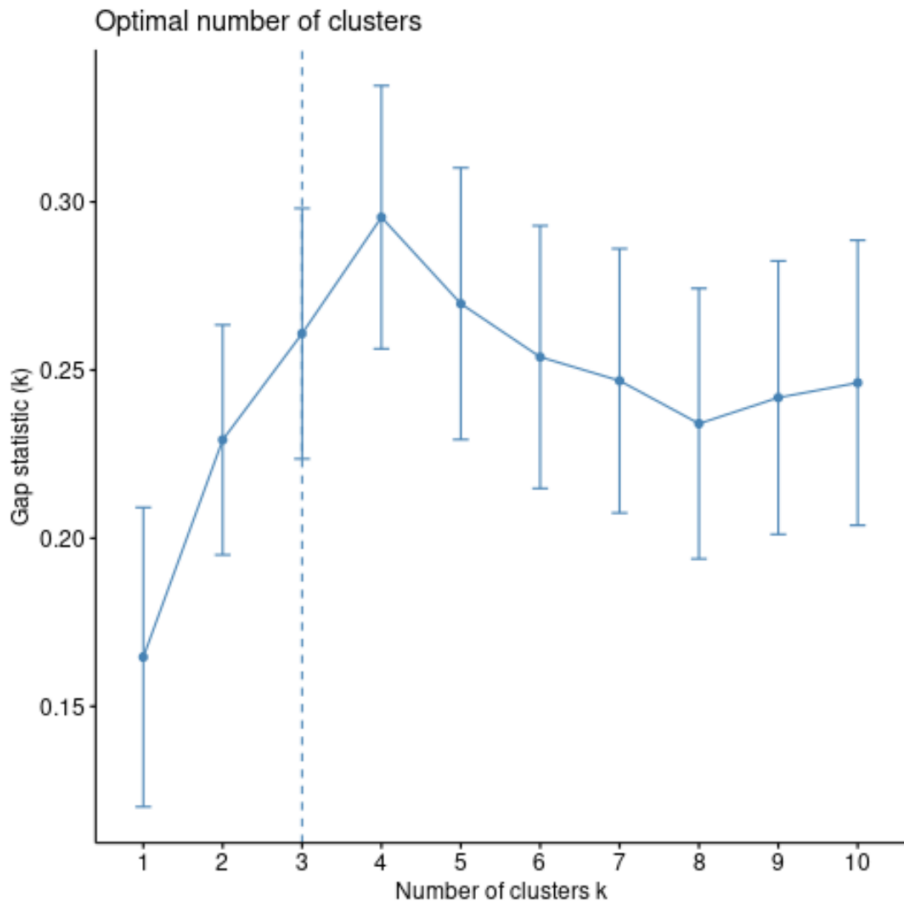
The **gap statistic** operates by comparing the total within-cluster variation across various values of *k* against the expected variation derived from a reference distribution (one assumed to have no inherent clustering structure). The value of *k* that produces the largest gap statistic is statistically considered the optimal number of clusters for the dataset.

We calculate this statistic in R using the `clusGap()` function and visualize the results immediately thereafter using `fviz_gap_stat()`:

#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 visualization provides clear confirmation: the gap statistic reaches its maximum point when *k* = 4. Based on this robust quantitative analysis, the 50 state observations will be segmented into four distinct clusters.

Step 5: Applying Cluster Assignments and Interpreting Results

The final step involves retrieving the cluster assignments (labels) and integrating them back into our original data frame. We use the `cutree()` function, which effectively performs the "cut" on the final hierarchical model to generate the 4 predetermined clusters. Note that we use the base R implementation `hclust()` here to prepare the model structure required by `cutree()`.

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

#perform hierarchical clustering using 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 resulting frequency table confirms the distribution of the 50 states across the four new clusters. We now merge these generated cluster labels with the original (unscaled) `USArrests` dataset to facilitate detailed interpretation:

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

#display first six rows of final data
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

To develop a clear profile and understanding of the defining characteristics of each segment, we employ the `aggregate()` function. This allows us to calculate the mean values of the crime and urban population variables specifically within each of the newly formed clusters.

#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

This aggregated output is the key to profiling the segments. For example, **Cluster 1** is distinctly characterized by states exhibiting exceptionally high rates of Murder and Assault, often coupled with lower-than-average urbanization rates (54.28%). Conversely, **Cluster 4** represents states with the lowest average crime rates across all categories and relatively low urbanization, defining them as the safest demographic in the dataset.

To illustrate the interpretative power of the results, the key average statistics for Cluster 1 are summarized below:

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

The complete, executable R code used throughout this hierarchical clustering example is available for review, replication, and further experimentation here.