How to Perform K-Means Clustering in R

Authored by

stats writer

December 17, 2025

K-means clustering is a fundamental and widely used unsupervised learning algorithm designed to partition a dataset into a pre-defined number of groups or clusters, denoted as K. This method seeks to minimize the variance within each cluster, ensuring that data points grouped together are highly similar to one another while being distinct from observations in other clusters. Implementing K-means clustering in R is straightforward, primarily utilizing the powerful `kmeans()` function. This function requires critical arguments such as the data matrix, the desired number of clusters (K), and the number of random initial configurations (nstart). Upon execution, the function efficiently returns crucial outputs, including the computed cluster centroids and the specific cluster assignments for every single data point in the input set.

The process of **clustering** is a core technique within machine learning and data mining, focusing on uncovering inherent structures and natural groupings within complex datasets. It systematically attempts to discover cohesive "clusters" of observations where the observations share high degrees of similarity based on their feature values. This methodology is vital when exploratory data analysis reveals potential non-obvious patterns that can inform decision-making or model development.

The fundamental objective of successful clustering is to achieve high intra-cluster similarity and high inter-cluster dissimilarity. Put simply, the goal is to define clusters such that all observations within a single cluster are statistically very similar to one another, typically measured by their spatial proximity in the feature space. Simultaneously, the algorithm ensures that these clusters are markedly different from observations residing in adjacent or distant clusters. This effective separation allows for meaningful categorization and subsequent targeted analysis based on the derived groupings.

Clustering is inherently classified as a form of unsupervised learning. Unlike supervised methods, which rely on labeled training data to predict a specific response variable, unsupervised methods operate solely by seeking underlying patterns, structures, or relationships within the input data itself without requiring any prior knowledge of class labels. The absence of a designated target variable defines its unsupervised nature and makes it suitable for exploratory data analysis where ground truth labels are missing.

A prime example of clustering application lies in **market segmentation**. Businesses frequently employ clustering algorithms when they possess detailed demographic and behavioral data about their customer base. This information might include various key metrics relevant to purchasing power and lifestyle:

Household income demographics
Household size and composition
Head of household **Occupation** category

Geographical factors, such as **distance from the nearest urban area**

By leveraging these diverse variables, clustering techniques allow companies to identify homogenous groups of households. These identified groups are deemed similar in ways that are highly relevant to consumer behavior. For instance, a cluster analysis might reveal groups that are more likely to purchase premium products or respond favorably to specific types of advertising campaigns, enabling highly effective and customized marketing strategies that maximize return on investment.

What is K-Means Clustering?

K-means clustering is arguably the most recognized partitioning algorithm, functioning as an iterative refinement process. The core concept involves partitioning N observations into exactly K clusters, where K is a user-defined integer representing the target number of groups. The algorithm seeks to assign each observation to the cluster whose mean (or centroid) is closest to it, effectively minimizing the within-cluster sum of squares (WCSS). The ultimate objective remains consistent: to produce K clusters where the inherent similarity among observations within any single cluster is maximized, while simultaneously maximizing the difference between observations belonging to separate clusters.

The stability and effectiveness of the K-means algorithm stem from its iterative refinement process. Since the initial assignment of observations or the random selection of starting centroids can significantly influence the final cluster configuration, the algorithm is typically run multiple times (using the `nstart` parameter) with different random starting points. The run that achieves the smallest total WCSS is selected as the optimal solution, guaranteeing robust and reliable cluster definitions that genuinely reflect the underlying structure of the data and mitigate the risk of converging to a poor local optimum.

In practice, performing K-means clustering involves a well-defined sequence of steps that guide the iterative optimization process. Understanding these steps is crucial for proper implementation and interpretation of the results, especially when dealing with high-dimensional data where visual inspection is impossible.

The standard procedure for executing K-means clustering is outlined sequentially:

Choose a Value for K (The Number of Clusters). This initial step is non-trivial as the selection of K dictates the granularity of the resulting partitions. Since there is no single predetermined correct value for K, data scientists often employ diagnostic tools (like the Elbow Method or Gap Statistic) to test several different values for K. The optimal value is determined by analyzing the resulting cluster separation, compactness, and overall interpretability for the specific business or scientific problem context.

Randomly Initialize Cluster Assignments or Centers. The algorithm begins by arbitrarily assigning each observation to one of the K clusters, or by randomly selecting K data points to serve as the initial cluster centers (centroids).
Iterative Refinement until Convergence. The core of K-means involves repeating the following two sub-steps until the cluster assignments stabilize, meaning no observation changes its cluster membership between consecutive iterations, or until a predefined maximum number of iterations is reached.

Calculate Cluster Centroids: For each of the K clusters, the new cluster centroid is computed. This centroid is the geometric center, represented by the vector of the mean values for all p features (dimensions) across all observations currently assigned to that specific kth cluster.
Reassign Observations: Every observation in the entire dataset is then reassigned to the cluster whose newly computed centroid is closest to it. The measurement of "closest" is typically defined using the squared Euclidean distance metric, which measures the straight-line distance between two points in multidimensional space, thereby minimizing the within-cluster variance.

K-Means Clustering Implementation in R

R provides excellent capabilities for performing K-means clustering, leveraging built-in functions and powerful extension packages specifically designed for cluster analysis and visualization. The following tutorial demonstrates a step-by-step process for executing, validating, and visualizing a K-means model using a classic dataset available in the R environment.

Step 1: Load the Necessary Packages

To efficiently conduct and interpret K-means clustering in R, we rely on two key external packages that enhance the capabilities of the base `kmeans()` function. We load `factoextra` for enhanced visualization tools, specifically used for determining the optimal number of clusters and plotting the final results clearly. We also load the `cluster` package, which contains essential functions related to cluster validation metrics, such as the Gap Statistic.

library(factoextra)
library(cluster)

Step 2: Load and Prepare the Data

For this practical demonstration, we will utilize the intrinsic `USArrests` dataset available within R's standard environment. This dataset contains comprehensive statistics for all 50 U.S. states in 1973, detailing crime rates per 100,000 residents across four variables: **Murder**, **Assault**, and **Rape**, alongside the percentage of the population residing in urban areas, denoted as **UrbanPop**.

The dataset is ideal for demonstrating K-means as it requires segmentation based on multivariate characteristics.

Data preparation is a critical step before applying K-means. Since K-means relies on distance measures, variables measured on vastly different scales (e.g., Assault rates are much higher than Murder rates) can unfairly dominate the clustering process. Therefore, it is imperative to normalize or scale the variables so they all possess equal importance in the distance calculation. The code below illustrates the necessary pre-processing steps: loading the data, handling missing values, and scaling the features.

Loading the Dataset: We begin by assigning the `USArrests` data to a new data frame called `df`.
Handling Missing Data: We ensure data integrity by removing any rows containing missing values (NA) using `na.omit()`.
Scaling Features: We standardize each variable using the `scale()` function, resulting in variables that have a mean of 0 and a standard deviation of 1. This ensures all variables are weighted equally when computing distances.

#load data
df <- USArrests

#remove rows with missing values (if any exist)
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: Determining the Optimal Number of Clusters (K)

As established, a critical requirement for K-means clustering is the prior specification of the number of clusters, K. Selecting an inappropriate K can lead to either overly sparse or overly broad

groupings, thus obscuring meaningful patterns. To guide this selection, R provides sophisticated tools focusing on metrics that evaluate the compactness (WCSS) and separation (Gap Statistic) of clusters for varying values of K. We recall the syntax of the R function:

kmeans(data, centers, nstart)

We must determine the value for the `centers` parameter (K). We will employ two standard, reliable diagnostic techniques to estimate the optimal K based on the characteristics of our scaled data.

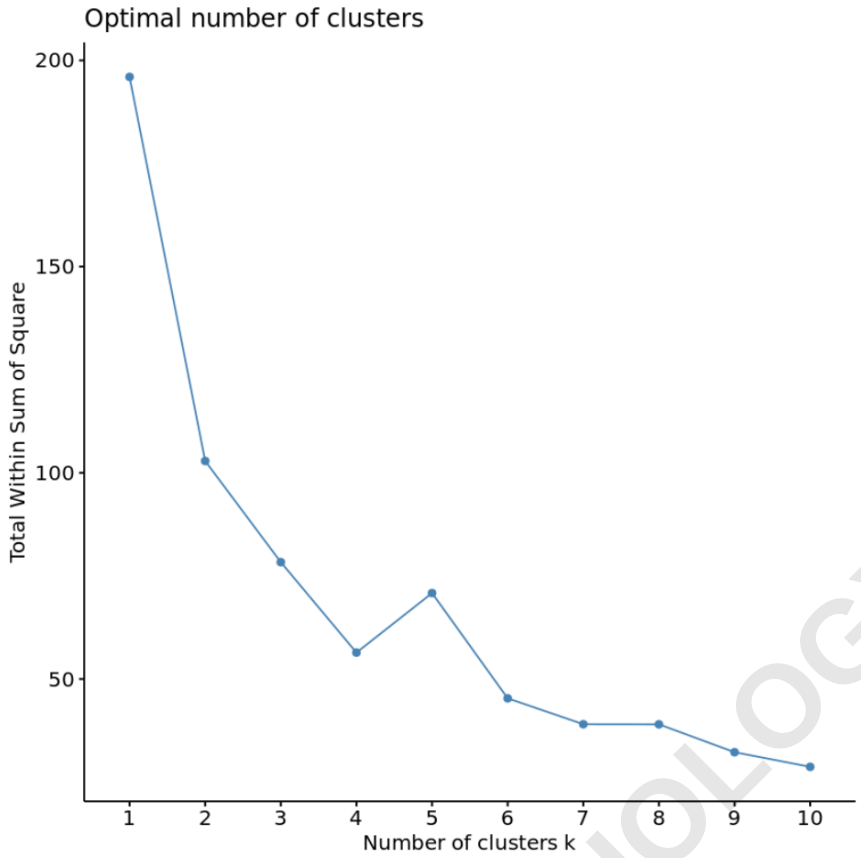
centers: The intended number of clusters, K.
nstart: Specifies the number of initial random configurations. We typically use a high value (like 25) to ensure the algorithm finds the best, most stable solution.

Method 1: The Elbow Method (Total Within Sum of Squares)

The Elbow Method evaluates the total within-cluster sum of squares (WCSS) as K increases. WCSS quantifies the compactness of the clusters; the lower the WCSS, the closer the observations are to their respective cluster centroids. We plot K against WCSS. Since adding more clusters always reduces WCSS, we look for the point of diminishing returns--the "elbow"--where the marginal gain from adding an extra cluster drops sharply, suggesting that the optimal balance between compactness and simplicity has been reached.

We utilize the `fviz_nbclust()` function from the `factoextra` package, setting the method parameter to `"wss"` (Within Sum of Squares) to generate this diagnostic plot:

fviz_nbclust(df, kmeans, method = "wss")



Analyzing the resulting plot, we look for the distinctive "bend" where the WCSS curve begins to level off. In this visualization of the USArrests data, a notable elbow is visually apparent when the number of clusters is set to $K=4$. This suggests that partitioning the data into four distinct groups provides the best trade-off between minimizing within-cluster variation and maintaining model parsimony.

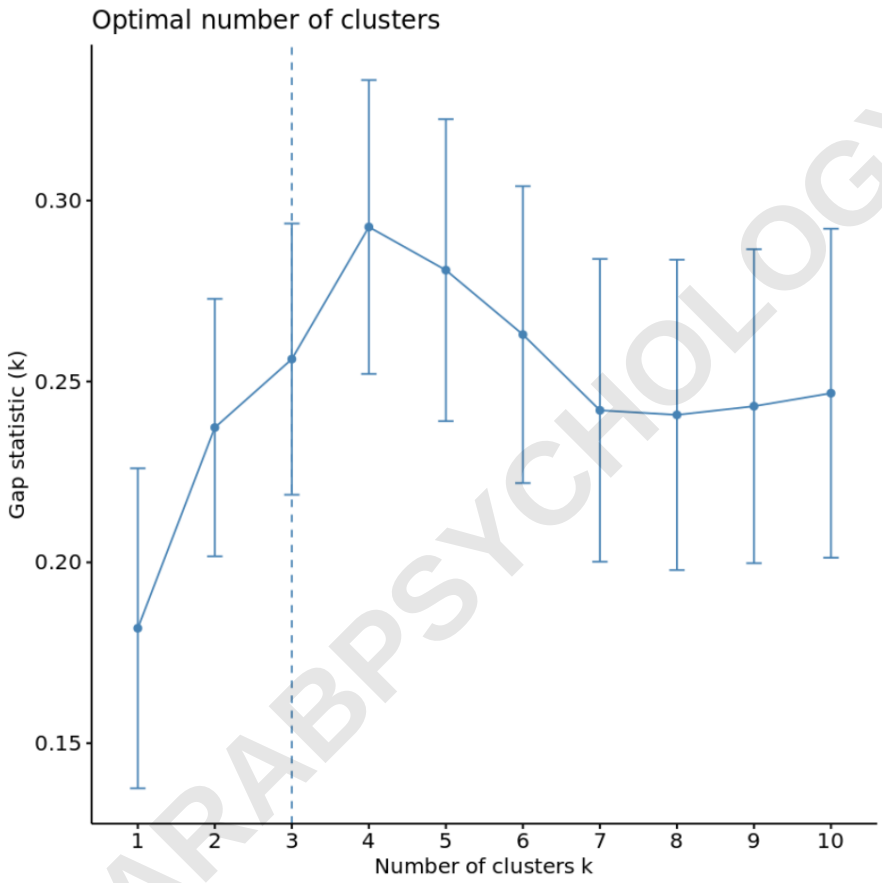
Method 2: The Gap Statistic

While the Elbow Method is intuitive, it can sometimes be subject to subjective interpretation. A more statistically rigorous approach is the Gap Statistic. This metric formally compares the total intra-cluster variation observed in the real data for different values of K against the expected variation derived from a reference null distribution--a dataset generated with no inherent clustering structure. The optimal number of clusters is the value of K that maximizes the gap statistic, indicating the most significant and statistically meaningful deviation from randomness.

We calculate the gap statistic using the `clusGap()` function from the `cluster` package, specifying the maximum K to test (`K.max = 10`) and setting the number of bootstrap samples (`B = 50`). The results are then visualized using the `fviz_gap_stat()` function:

```
#calculate gap statistic based on number of clusters
gap_stat <- clusGap(df,
FUN = kmeans,
nstart = 25,
K.max = 10,
B = 50)

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



The visualization of the Gap Statistic strongly confirms the initial finding from the Elbow Method. The gap statistic reaches its highest point precisely at $K=4$ clusters. Since both widely accepted methods converge on the same result, we can confidently proceed with $K=4$ as the optimal number of clusters for segmenting the U.S. states based on crime and urbanization rates.

Step 4: Performing K-Means Clustering with Optimal K

With the optimal number of clusters determined to be $K=4$, we now execute the final K-means

clustering algorithm on our scaled dataset. To ensure the reproducibility of our findings, particularly due to the random initialization component of the K-means algorithm, we first set a seed value. We use `nstart=25` to maximize the chance of finding the global optimum by trying 25 different initial configurations and selecting the one with the lowest overall WCSS.

#make this example reproducible
set.seed(1)

#perform k-means clustering with k = 4 clusters
km <- kmeans(df, centers = 4, nstart = 25)

#view results summary
km

K-means clustering with 4 clusters of sizes 16, 13, 13, 8

Cluster means:
Murder Assault UrbanPop Rape
1 -0.4894375 -0.3826001 0.5758298 -0.26165379
2 -0.9615407 -1.1066010 -0.9301069 -0.96676331
3 0.6950701 1.0394414 0.7226370 1.27693964
4 1.4118898 0.8743346 -0.8145211 0.01927104

Clustering vector:
Alabama Alaska Arizona Arkansas California Colorado
4 3 3 4 3 3
Connecticut Delaware Florida Georgia Hawaii Idaho
1 1 3 4 1 2
Illinois Indiana Iowa Kansas Kentucky Louisiana
3 1 2 1 2 4
Maine Maryland Massachusetts Michigan Minnesota Mississippi
2 3 1 3 2 4
Missouri Montana Nebraska Nevada New Hampshire New Jersey
3 2 2 3 2 1
New Mexico New York North Carolina North Dakota Ohio Oklahoma
3 1 2 1 1 1
Oregon Pennsylvania Rhode Island South Carolina South Dakota Tennessee
1 1 1 4 2 4
Texas Utah Vermont Virginia Washington West Virginia
3 1 2 1 1 2
Wisconsin Wyoming

2 1

Within cluster sum of squares by cluster:
16.212213 11.952463 19.922437 8.316061
(between_SS / total_SS = 71.2 %)

Available components:

"cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss"
"size" "iter" "ifault"

The output summary provides comprehensive insight into the final cluster configuration. Key results include the cluster sizes, confirming how the 50 states were distributed across the four partitions:

16 states were allocated to the first cluster.
13 states were allocated to the second cluster.
13 states were allocated to the third cluster.
8 states were allocated to the fourth cluster.

Furthermore, the output displays the scaled means (centroids) for each cluster, which are essential for interpreting the characteristic profile of each group in terms of standard deviations from the dataset mean. Importantly, the result "between_SS / total_SS = 71.2%" indicates that 71.2% of the total variability in the data is captured by the differences between the clusters, demonstrating excellent segregation between the defined groups.

Step 5: Visualizing and Interpreting the Clusters

To gain a clearer spatial understanding of how the states are grouped, we can project the multidimensional cluster structure onto a two-dimensional plane. This is typically achieved using Principal Component Analysis (PCA) to find the dimensions that capture the maximum variance. The `fviz_cluster()` function plots the clusters based on the first two principal components, visually illustrating the quality of separation achieved by the K-means algorithm.

#plot results of final k-means model, projected onto the first two principal components
fviz_cluster(km, data = df)

The visual representation confirms that the four clusters are reasonably well-separated in the feature space, aligning with the large "between SS" value observed previously. Cluster 3 and Cluster 4 show some proximity, indicating shared characteristics, while Cluster 2 and Cluster 1 are distinct and highly separated from the others.

For practical interpretation, analyzing the scaled cluster means provides information on relative differences, but examining the means of the original, unscaled variables provides immediate, real-world context. We use the `aggregate()` function to calculate the mean of the original USArrests variables for all states belonging to each cluster. This allows us to characterize the profiles of the four distinct state groups based on actual crime rates and urbanization percentages:

#find means of each cluster using the original data values
aggregate(USArrests, by=list(cluster=km$cluster), mean)

cluster Murder Assault UrbanPop Rape

1 3.60000 78.53846 52.07692 12.17692
2 10.81538 257.38462 76.00000 33.19231

3 5.65625 138.87500 73.87500 18.78125
4 13.93750 243.62500 53.75000 21.41250

By reviewing the means of the original variables, we can now define the characteristics of each cluster precisely:

Cluster 1: Represents states with generally **low crime rates** (e.g., 3.6 murders per 100k) and **average urbanization** (52.1%).
Cluster 2: Represents states with **high crime rates across all categories** (e.g., 10.8 murders, 257.4 assaults) and **high urbanization** (76.0%).
Cluster 3: Represents states with **moderate crime rates** and **high urbanization** (73.9%), positioning them as an intermediate group.
Cluster 4: Represents states characterized by **very high murder rates** (13.9) and high assault rates (243.6), yet surprisingly **low urbanization** (53.8%), suggesting high rural violence.

Finally, for subsequent analysis, mapping, or reporting, it is beneficial to append the cluster assignment back to the original, unscaled dataset. This integrates the new structural information found by K-means directly into the state records, creating a comprehensive final dataset.

#add cluster assigment to original data
final_data <- cbind(USArrests, cluster = km$cluster)

#view final data structure
head(final_data)

Murder Assault UrbanPop Rape cluster

Alabama 13.2 236 58 21.2 4
Alaska 10.0 263 48 44.5 2
Arizona 8.1 294 80 31.0 2
Arkansas 8.8 190 50 19.5 4
California 9.0 276 91 40.6 2
Colorado 7.9 204 78 38.7 2

Pros and Cons of K-Means Clustering

While K-means clustering is a powerful and popular tool for partitioning data, it is essential to understand its inherent strengths and limitations before deployment in a production environment. Its popularity in big data applications is largely attributed to its computational efficiency and simplicity, especially when handling large volumes of data.

The primary advantages offered by K-means include:

Efficiency and Speed: It is remarkably quick and computationally inexpensive, possessing a linear time complexity $O(nkt)$, where n is the number of data points, k is the number of clusters, and t is the number of iterations. This makes it highly scalable for massive datasets.
Simplicity and Ease of Use: The underlying algorithm is easy to understand and straightforward to implement, requiring minimal complex configuration beyond specifying the number of clusters K.

However, analysts must be aware of its potential drawbacks, which often dictate whether K-means is the most appropriate algorithm for a given task:

Dependency on K Specification: The necessity of manually specifying the number of clusters K beforehand is a significant operational limitation, often requiring preliminary heuristic methods to determine the optimal value.
Sensitivity to Outliers: K-means relies heavily on the mean (centroid) calculation. Consequently, it is highly susceptible to the influence of outliers, which can dramatically skew the cluster centers and lead to inaccurate segmentation.
Assumption of Spherical Clusters: K-means inherently assumes that clusters are convex and roughly spherical in shape, performing poorly when clusters have complex, non-linear boundaries or unequal sizes and densities.

For scenarios where K-means limitations become prohibitive--such as the presence of numerous outliers or non-spherical data distributions--alternative clustering techniques may offer superior performance. Two common alternatives are K-medoids clustering (which uses medoids, making it more robust to outliers) and hierarchical clustering (which avoids the need to specify K upfront).

You can find the complete R code used in this example and additional resources for clustering here.