

# How to do K-Medoids in R

Authored by  
**stats writer**

December 17, 2025

## RECOMMENDED CITATION

stats writer (2025). *How to do K-Medoids in R*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=107700>

## Introduction to Clustering and Unsupervised Learning

Clustering is a fundamental method within machine learning dedicated to discovering inherent structures within datasets. The core goal is to partition a collection of data objects into groups, or **clusters**, such that objects within the same cluster are highly similar to one another, while observations belonging to different clusters are maximally dissimilar. This process is essential for pattern recognition and exploratory data analysis across diverse fields.

This analytical approach falls under the umbrella of unsupervised learning, primarily because we are seeking to find structure and organization within the data without relying on predefined labels or a known response variable. Clustering algorithms are tasked solely with identifying underlying patterns based on the inherent proximity or distance between data points, making them invaluable for exploratory data mining.

The practical applications of clustering are vast, particularly in fields like market segmentation, image processing, and bioinformatics. In marketing, for example, companies frequently employ clustering to identify distinct groups of customers based on demographic and behavioral variables such as:

**Household income** and spending habits.

**Household size** and composition.

Head of household **Occupation** or professional status.

**Distance** from nearest urban area, reflecting lifestyle factors.

By identifying these homogeneous groups, businesses can tailor marketing strategies and product offerings, significantly increasing the likelihood that customers will purchase specific products or respond favorably to targeted advertising campaigns. This optimization of resource allocation relies heavily on accurately identifying customer groups through robust clustering techniques.

## Understanding K-Medoids Clustering vs. K-Means

Historically, one of the most widely used clustering algorithms is K-means clustering. K-means operates by minimizing the total within-cluster variance, using the mean of the cluster members to define the cluster center, or **centroid**. While powerful and computationally efficient, K-means has a critical drawback: it is highly sensitive to outliers. Extreme values can significantly shift the cluster mean, distorting the boundaries and resulting in suboptimal groupings that may not truly reflect the underlying data structure.

An effective alternative designed to mitigate this weakness is **K-Medoids clustering**. This technique, also known as Partitioning Around Medoids (PAM), shares the fundamental goal of partitioning the dataset into  $K$  clusters. However, unlike K-means, K-Medoids defines the cluster

center not by the mean of all points, but by an actual data point within the cluster--the **medoid**. The medoid is the most centrally located observation within the cluster, minimizing the sum of dissimilarities between it and all other points in that cluster.

The core distinction lies in the representation of the cluster center. Because K-Medoids selects a representative observation (the medoid) instead of calculating an artificial centroid (the mean), it is inherently more **robust to outliers**. Outliers still exist, but they have a far smaller influence on the position of the medoid compared to their heavy influence on the mean calculation in K-means. If a dataset contains significant noise or extreme values, K-Medoids often provides a more reliable and stable clustering solution, leading to more trustworthy partitions.

## The Mechanism of K-Medoids (PAM Algorithm)

The K-Medoids algorithm, typically implemented via the PAM method, follows a precise iterative procedure to achieve optimal cluster separation. The process guarantees that every observation in the dataset is assigned to exactly one of the designated  $K$  clusters, striving to minimize the total dissimilarity between points and their assigned medoids. The overall objective of the algorithm is to minimize the sum of pairwise dissimilarities between the observations and their corresponding medoids.

The implementation involves sequential steps that are repeated until convergence, meaning the cluster assignments stabilize and no further movement of data points between clusters occurs that would improve the overall partitioning quality. This iterative refinement process ensures that the final medoids are the best possible representatives for their respective groups.

The K-Medoids procedure is structured as follows:

**Initialization: Choose the number of clusters,  $K$ .** The analyst must first specify the desired number of clusters. Determining the optimal  $K$  is often the most subjective part of the process, requiring the use of statistical tools like the elbow method or the gap statistic, which we will demonstrate later. Initial analysis usually involves testing several values for  $K$  to evaluate which division yields the most meaningful results for the specific problem domain.

**Initial Medoid Selection.** Randomly select  $K$  observations from the dataset to serve as the initial medoids. These initial medoids represent the starting centers of the clusters.

**Assignment Step.** Assign every non-selected observation (non-medoid) to the closest of the  $K$  medoids. The definition of "closest" relies on a chosen distance metric, most commonly the Euclidean distance, which calculates the straight-line separation between points in multi-dimensional space.

**Update Step (Swap Phase).** This is the crucial iterative step. For each cluster, the algorithm tests

whether replacing the current medoid with a non-medoid observation within the same cluster would decrease the overall clustering cost (the sum of dissimilarities). If a swap results in a reduction of the total cost, the swap is performed, and the new observation becomes the medoid.

**Convergence.** Steps 3 and 4 are repeated until no swaps occur that improve the overall objective function. At this point, the cluster assignments have stabilized, and the final  $K$  clusters are determined.

### Technical Insight: Robustness

The inherent robustness of K-Medoids stems directly from its reliance on actual data points (medoids) rather than computed means (centroids). While K-means minimizes the squared error (L2 norm), which heavily penalizes large distances associated with outliers, K-Medoids minimizes the sum of absolute differences (L1 norm or other dissimilarity measures). This mathematical difference makes K-Medoids significantly less susceptible to distortion caused by extreme or spurious data points compared to its K-means counterpart. In cases where data quality is a concern, K-Medoids is typically preferred for its resilience.

## Implementing K-Medoids in R: Setup and Data Preparation

To demonstrate the practical application of K-Medoids clustering, we will walk through a complete example using the R programming language. The primary function for this analysis is `pam()`, which is housed within the `cluster` package. We will also utilize the `factoextra` package for enhanced visualization and determination of the optimal number of clusters. It is crucial to load all necessary libraries before beginning the analysis, as these packages provide the core functionalities required for partitioning the data and assessing the quality of the resulting clusters.

### Step 1: Load the Necessary Packages

The first action is loading the two primary packages essential for executing and evaluating K-Medoids clustering in R: `factoextra` for plotting and cluster evaluation, and `cluster`, which contains the core `pam()` function. If these packages are not already installed on your system, you would need to install them using `install.packages()` before executing the `library()` calls.

```
library(factoextra)
```

```
library(cluster)
```

### Step 2: Load and Prepare the Data

For this instructional example, we will employ the built-in `USArrests` dataset in R. This dataset provides crime statistics across 50 US states, including arrest rates per 100,000 residents for

**Murder, Assault, and Rape**, along with the percentage of the population living in urban areas (**UrbanPop**). Since clustering algorithms rely heavily on distance calculations, proper data preparation--specifically handling missing values and scaling variables--is vital to ensure that all features contribute equally to the distance measurements, preventing variables with large absolute values from dominating the clustering process.

The following code snippet demonstrates how to load the data, handle any potentially missing observations, and standardize the variables. Scaling the data ensures that variables measured on different scales do not disproportionately influence the clustering outcome. Standardization transforms each variable to have a mean of zero and a standard deviation of one, a crucial step for distance-based algorithms.

We first load the `USArrests` dataset into a data frame named `df`.

We remove any rows containing **missing values** using the `na.omit()` function.

We apply the `scale()` function to standardize all numerical columns.

Finally, we preview the first six rows of the scaled dataset to confirm the transformation.

#### **#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 Number of Clusters (K)**

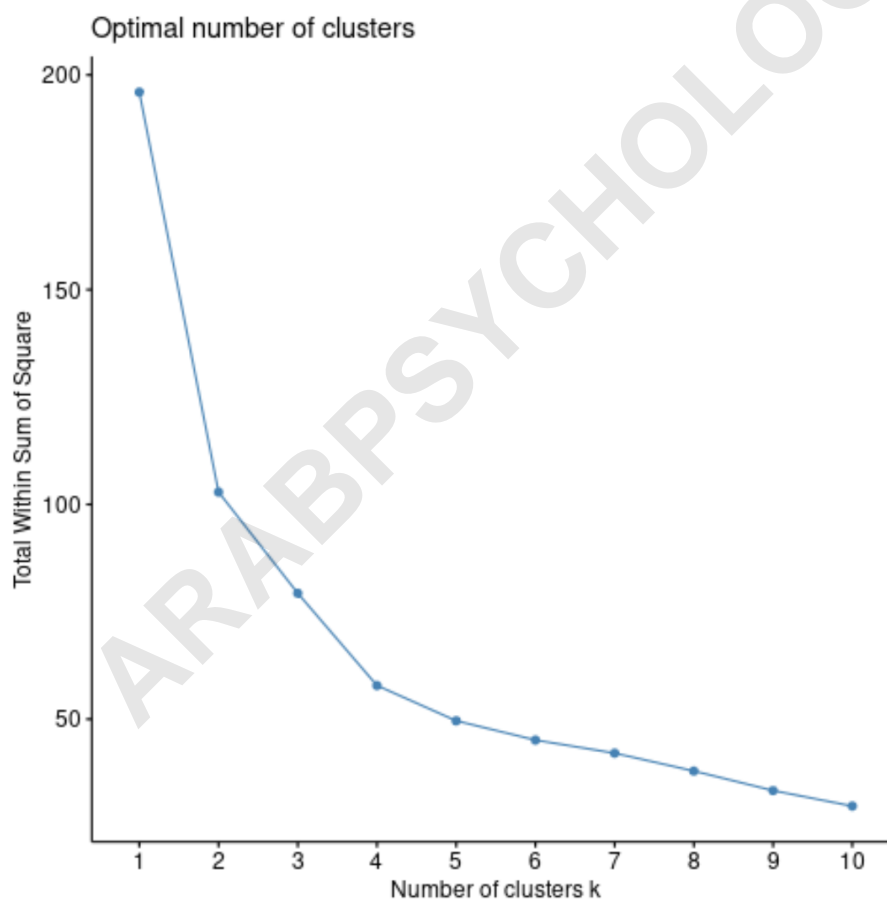
A critical decision in partitioning algorithms like K-Medoids is selecting the appropriate value for  $K$ , the number of clusters. Using the `pam()` function requires this parameter. The basic syntax for this

function is `pam(data, k, metric = "euclidean", stand = FALSE)`. Since the optimal  $K$  is rarely known a priori, we utilize evaluation metrics to identify the best fit. We will analyze two primary methods: the Elbow Method, based on the Total Within Sum of Squares (WSS), and the Gap Statistic method.

### The Elbow Method (Total Within Sum of Squares)

The first diagnostic plot uses the total **Within Sum of Squares (WSS)**. WSS measures the compactness of the clustering; lower WSS values indicate tighter, more homogeneous clusters. We use the `fviz_nbclust()` function from `factoextra`, specifying `pam` as the clustering function and `wss` as the method. This function automatically calculates and plots the WSS for a range of potential  $K$  values.

```
fviz_nbclust(df, pam, method = "wss")
```



The WSS naturally decreases as the number of clusters ( $K$ ) increases, eventually reaching zero when  $K$  equals the number of observations. To find the optimal  $K$ , we look for the "elbow" point--the location on the plot where the rate of decrease dramatically slows down or levels off. This point

signals diminishing returns; adding more clusters beyond this point provides minimal improvement in cluster tightness and risks overfitting the data, leading to partitions that are too specific to the sample. Based on the visual analysis of this plot, there is a pronounced bend occurring at **K = 4** clusters.

### The Gap Statistic Method

The second, and often more robust, method involves calculating the gap statistic. This metric compares the total intra-cluster variation of the clustered data against the variation expected from a reference null distribution (a dataset with no obvious clustering structure). The goal is to find the K value where the difference between the observed WSS and the reference WSS is maximized.

We calculate the gap statistic using the `clusGap()` function, again specifying `pam` as our clustering function. We set the maximum number of clusters to consider (`K.max = 10`) and use 50 bootstrapped iterations (`B = 50`) for stable estimation of the reference distribution. The results are then visualized using `fviz_gap_stat()` to identify the peak value.

```
#calculate gap statistic based on number of clusters
```

```
gap_stat <- clusGap(df,
```

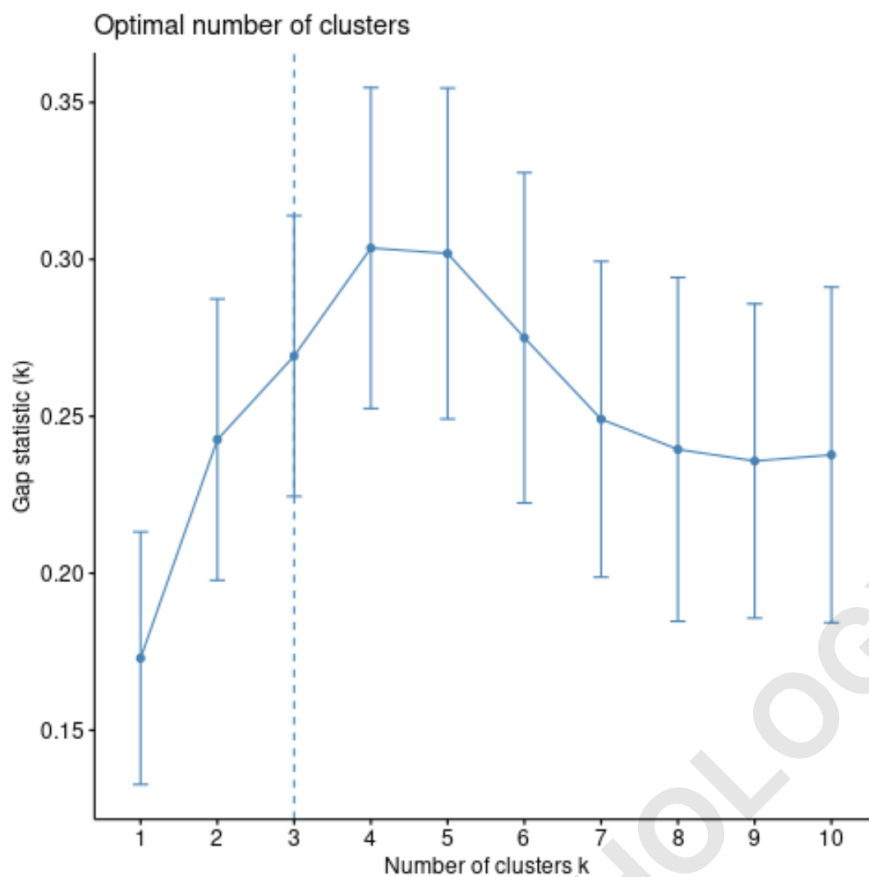
```
  FUN = pam,
```

```
  K.max = 10, #max clusters to consider
```

```
  B = 50) #total bootstrapped iterations
```

```
#plot number of clusters vs. gap statistic
```

```
fviz_gap_stat(gap_stat)
```



The optimal number of clusters is typically identified as the smallest  $K$  value for which the gap statistic curve reaches its peak. In this visualization, the highest peak clearly occurs at  $K = 4$ , which provides strong confirmation of the result derived from the Elbow Method, giving us high confidence in proceeding with four clusters.

#### Step 4: Executing the K-Medoids Model with Optimal K

Having confidently determined that  $K = 4$  is the optimal number of clusters for the `USArrests` dataset, we can now execute the final K-Medoids clustering using the `pam()` function. This step partitions the 50 US states into four distinct groups based on their scaled crime and urbanization rates. To ensure reproducibility of the random initialization process inherent in the PAM algorithm, we set a seed before running the model.

```
#make this example reproducible
```

```
set.seed(1)
```

```
#perform k-medoids clustering with k = 4 clusters
```

```
kmed <- pam(df, k = 4)
```

```
#view results
```

```
kmed
```

```
ID Murder Assault UrbanPop Rape
```

```
Alabama 1 1.2425641 0.7828393 -0.5209066 -0.003416473
```

```
Michigan 22 0.9900104 1.0108275 0.5844655 1.480613993
```

```
Oklahoma 36 -0.2727580 -0.2371077 0.1699510 -0.131534211
```

```
New Hampshire 29 -1.3059321 -1.3650491 -0.6590781 -1.252564419
```

```
Clustering vector:
```

```
Alabama Alaska Arizona Arkansas California
```

```
1 2 2 1 2
```

```
Colorado Connecticut Delaware Florida Georgia
```

```
2 3 3 2 1
```

```
Hawaii Idaho Illinois Indiana Iowa
```

```
3 4 2 3 4
```

```
Kansas Kentucky Louisiana Maine Maryland
```

```
3 3 1 4 2
```

```
Massachusetts Michigan Minnesota Mississippi Missouri
```

```
3 2 4 1 3
```

```
Montana Nebraska Nevada New Hampshire New Jersey
```

```
3 3 2 4 3
```

```
New Mexico New York North Carolina North Dakota Ohio
```

```
2 2 1 4 3
```

```
Oklahoma Oregon Pennsylvania Rhode Island South Carolina
```

```
3 3 3 3 1
```

```
South Dakota Tennessee Texas Utah Vermont
```

```
4 1 2 3 4
```

```
Virginia Washington West Virginia Wisconsin Wyoming
```

```
3 3 4 4 3
```

```
Objective function:
```

```
build swap
```

```
1.035116 1.027102
```

```
Available components:
```

```
"medoids" "id.med" "clustering" "objective" "isolation"
```

```
"clusinfo" "silinfo" "diss" "call" "data"
```

The output provides comprehensive details about the resulting model. Crucially, the `pam()` output explicitly lists the selected **medoids**--the four observations that best represent the center of their respective clusters. Since K-Medoids ensures medoids are actual data points, these four states

serve as the most typical example of their cluster profile:

Cluster 1 Medoid: **Alabama**

Cluster 2 Medoid: **Michigan**

Cluster 3 Medoid: **Oklahoma**

Cluster 4 Medoid: **New Hampshire**

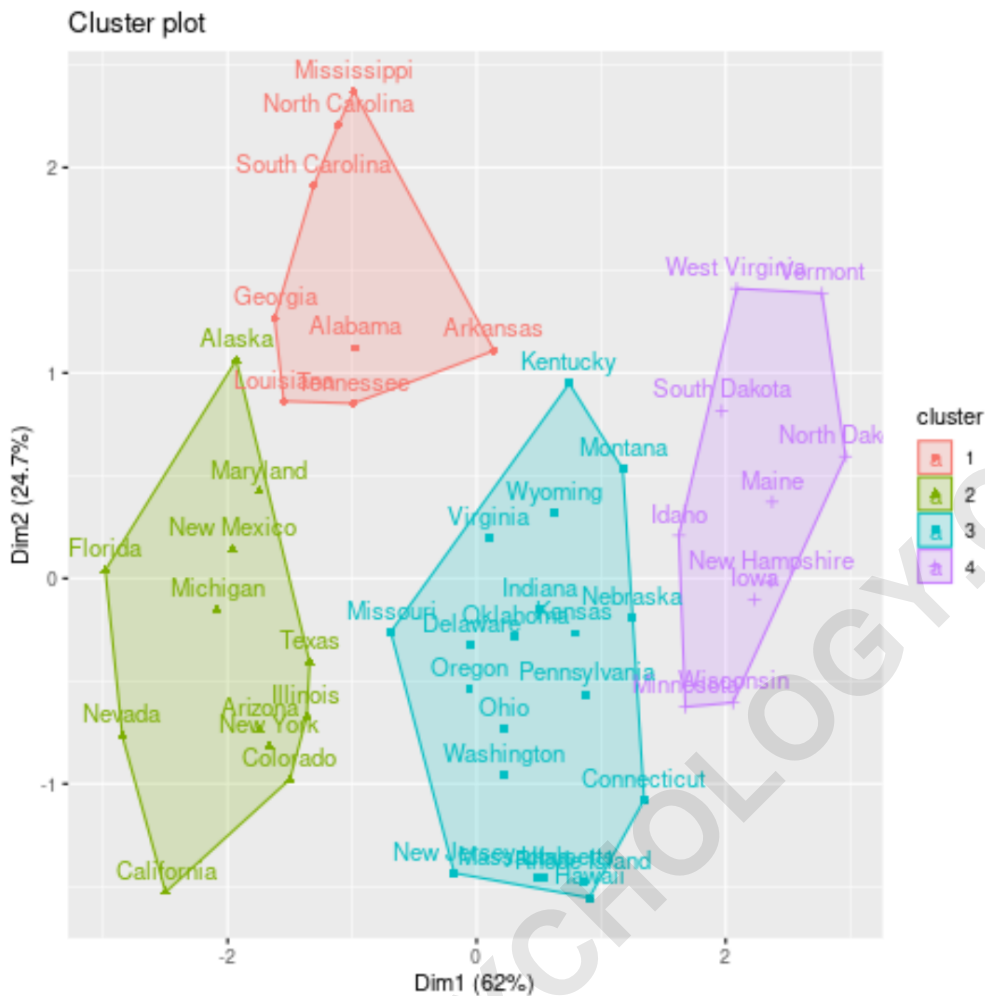
The clustering vector immediately following the medoid identification shows the cluster assignment for every state in the dataset, indicating how the algorithm has grouped the data based on crime statistics. The objective function values (build and swap) confirm the total dissimilarity cost achieved by the final iteration of the algorithm.

### Step 5: Visualizing and Interpreting the Clusters

To gain intuitive insight into how the states were grouped, we visualize the results. Since the clustering analysis utilized four variables (four dimensions), we use the `fviz_cluster()` function to project the clusters onto a two-dimensional scatterplot. This is achieved through Principal Component Analysis (PCA), where the axes typically represent the first two principal components (PC1 and PC2), which collectively capture the largest amount of variance in the original data.

**#plot results of final k-medoids model**

**fviz\_cluster(kmed, data = df)**



This visualization clearly delineates the four groups. We can observe the spatial relationship between the clusters: Clusters 1 and 2 tend to be associated with higher standardized crime rates, while Cluster 4 is distinctly separate, likely representing states with the lowest overall crime statistics and possibly lower urbanization. Cluster 3 generally occupies the middle ground in terms of these combined metrics. The plot confirms that the clusters are reasonably well-separated and defined by the medoids, which are indicated by large points within their respective clusters.

Finally, it is often necessary to integrate the cluster assignments back into the original, unscaled dataset for business or policy interpretation. By using the `cbind()` function, we attach the cluster column generated by `kmed$cluster` to the initial `USArrests` data frame, allowing direct analysis of the raw crime rates associated with each group. This final data structure is essential for deriving actionable conclusions, such as characterizing the specific crime profiles that define the differences between Cluster 1 (high crime, high assault) and Cluster 4 (low crime, low urbanization).

#### #add cluster assignment to original data

```
final_data <- cbind(USArrests, cluster = kmed$cluster)
```

```
#view 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 1
```

```
California 9.0 276 91 40.6 2
```

```
Colorado 7.9 204 78 38.7 2
```

## Conclusion and Resources

K-Medoids clustering provides a powerful, robust alternative to K-means, especially when dealing with datasets where the presence of outliers could skew traditional mean-based partitioning. By utilizing medoids--actual observations--as cluster centers, the algorithm delivers highly interpretable and stable results, as demonstrated through the rigorous analysis of the `USArrests` data in R. Successful implementation hinges on careful data preparation, robust selection of the optimal  $K$  value using statistical metrics like the elbow and gap statistic, and thoughtful interpretation of the final cluster assignments.

You can find the complete R code used in this example [here](#).