

How do you perform the K-Medoids algorithm in R? Can you provide a step-by-step example?

Authored by
stats writer

April 22, 2024

RECOMMENDED CITATION

stats writer (2024). *How do you perform the K-Medoids algorithm in R? Can you provide a step-by-step example?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=138269>

The K-Medoids algorithm is a clustering technique used to partition a dataset into a predetermined number of clusters. It is a variation of the K-Means algorithm, with the main difference being that instead of using the mean value of the cluster as the center point, it uses the most centrally located data point, known as the medoid.

To perform the K-Medoids algorithm in R, follow these steps:

1. Prepare the dataset: Import the dataset into R and make sure it is in a format that can be used for clustering. If necessary, preprocess the data by removing duplicates, handling missing values, and scaling the variables.
2. Choose the number of clusters (K): Before running the algorithm, you must decide on the number of clusters you want to create. This can be done using domain knowledge, or by using techniques like the elbow method or silhouette analysis.
3. Install and load the necessary packages: The K-Medoids algorithm is not included in the base R package, so you will need to install and load the "cluster" package.
4. Run the algorithm: Use the pam() function from the "cluster" package to perform the K-Medoids clustering. Specify the dataset, the number of clusters (K), and any other necessary parameters.
5. Visualize the results: Once the algorithm has been run, you can use the plot() function to visualize the clusters. This will show the data points and the cluster centers (medoids).

Here is a step-by-step example of how to perform the K-Medoids algorithm in R using the "iris" dataset included in the base R package:

1. Prepare the dataset:

```
...
```

```
# Load the dataset
```

```
data(iris)
```

```
# Remove the Species column (we will not use it for clustering)
```

```
iris <- iris
```

```
# Scale the variables
```

```
iris_scaled <- scale(iris)
```

```
...
```

2. Choose the number of clusters (K):

```
...
```

```
# Use the elbow method to determine the optimal K value
```

```
wss <- (nrow(iris_scaled)-1)*sum(apply(iris_scaled,2,var))
```

```
for (i in 2:10) wss <- sum(kmeans(iris_scaled, centers=i)$withinss)
plot(1:10, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
...

```

3. Install and load the necessary packages:

```
...
```

```
# Install and load the "cluster" package
```

```
install.packages("cluster")
```

```
library(cluster)
```

```
...
```

4. Run the algorithm:

```
...
```

```
# Perform K-Medoids clustering with K=3
```

```
km <- pam(iris_scaled, k=3)
```

```
# View the cluster assignment for each data point
```

```
km$clustering
```

```
...
```

5. Visualize the results:

```
...
```

```
# Plot the clusters
```

```
plot(iris_scaled, col=km$clustering, cex=2, pch=20)
```

```
# Add the cluster centers (medoids)
```

```
points(km$medoids, col=1:3, pch=8, cex=3, lwd=2)
```

```
...
```

This will result in a plot showing the data points colored according to their assigned cluster, with the cluster centers (medoids) marked with a larger symbol.

K-Medoids in R: Step-by-Step Example

Clustering is a technique in machine learning that attempts to find groups or *clusters* of observations within a dataset.

The goal is to find clusters such that the observations within each cluster are quite similar to each other, while observations in different clusters are quite different from each other.

Clustering is a form of unsupervised learning because we're simply attempting to find structure within a dataset rather than predicting the value of some response variable.

Clustering is often used in marketing when companies have access to information like:

Household income
Household size
Head of household
Occupation
Distance from nearest urban area

When this information is available, clustering can be used to identify households that are similar and may be more likely to purchase certain products or respond better to a certain type of advertising.

One of the most common forms of clustering is known as k-means clustering.

Unfortunately, this method can be influenced by outliers so an alternative that is often used is k-medoids

clustering.

What is K-Medoids Clustering?

K-medoids clustering is a technique in which we place each observation in a dataset into one of K clusters.

The end goal is to have K clusters in which the observations within each cluster are quite similar to each other while the observations in different clusters are quite different from each other.

In practice, we use the following steps to perform K-means clustering:

1. Choose a value for K .

First, we must decide how many clusters we'd like to identify in the data. Often we have to simply test several different values for K and analyze the results to see which number of clusters seems to make the most sense for a given problem.

2. Randomly assign each observation to an initial cluster, from 1 to K .

3. Perform the following procedure until the cluster

assignments stop changing.

For each of the K clusters, compute the cluster *centroid*. This is the vector of the p feature medians for the observations in the k th cluster. Assign each observation to the cluster whose centroid is closest. Here, *closest* is defined using Euclidean distance.

Technical Note:

Because k-medoids computes cluster centroids using medians instead of means, it tends to be more robust to outliers compared to k-means.

In practice, if there are no extreme outliers in the dataset then k-means and k-medoids will produce similar results.

K-Medoids Clustering in R

The following tutorial provides a step-by-step example of how to perform k-medoids clustering in R.

Step 1: Load the Necessary Packages

First, we'll load two packages that contain several

useful functions for k-medoids clustering in R.

```
library(factoextra)
```

```
library(cluster)
```

Step 2: Load and Prep the Data

For this example we'll use the *USArrests* dataset built into R, which contains the number of arrests per 100,000 residents in each U.S. state in 1973 for *Murder*, *Assault*, and *Rape* along with the percentage of the population in each state living in urban areas, *UrbanPop*.

The following code shows how to do the following:

Load the *USArrests* dataset
Remove any rows with missing values
Scale each variable in the dataset to have a mean of 0 and a standard deviation of 1

```
#load data
```

```
df <- USArrests
```

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

```
#scale each variable to have a mean of 0 and sd of 1df
```

```
<- 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: Find the Optimal Number of Clusters

To perform k-medoids clustering in R we can use the `pam()` function, which stands for "partitioning around medians" and uses the following syntax:

```
pam(data, k, metric = "euclidean", stand = FALSE)
```

where:

data: Name of the dataset.
k: The number of clusters.
metric: The metric to use to calculate distance.

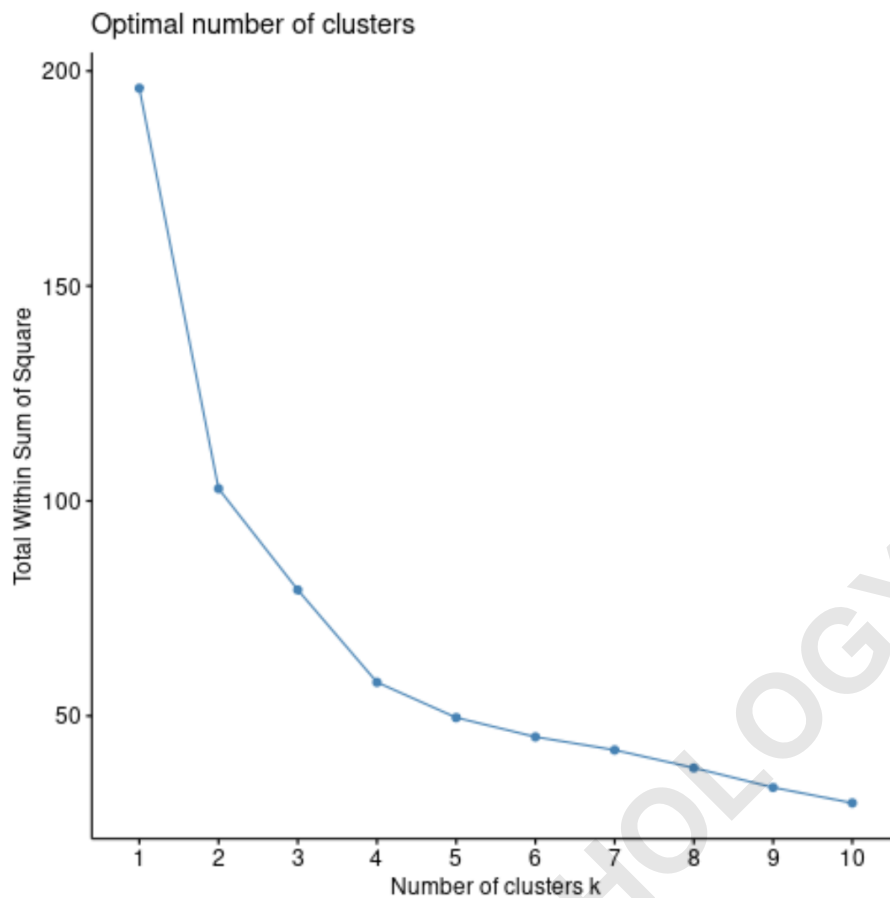
Default is *euclidean* but you could also specify *manhattan*. **stand**: Whether or not to standardize each variable in the dataset. Default is **FALSE**.

Since we don't know beforehand how many clusters is optimal, we'll create two different plots that can help us decide:

1. Number of Clusters vs. the Total Within Sum of Squares

First, we'll use the `fviz_nbclust()` function to create a plot of the number of clusters vs. the total within sum of squares:

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



The total within sum of squares will typically always increase as we increase the number of clusters, so when we create this type of plot we look for an "elbow" where the sum of squares begins to "bend" or level off.

The point where the plot bends is typically the optimal number of clusters. Beyond this number, overfitting is likely to occur.

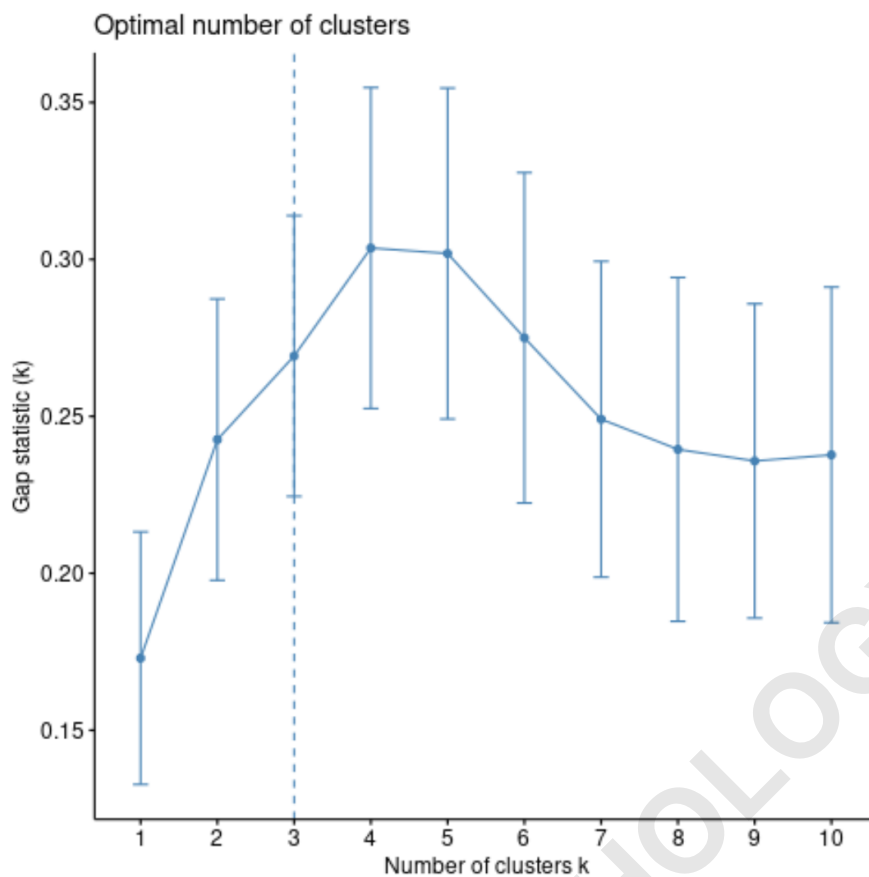
For this plot it appear that there is a bit of an elbow or "bend" at $k = 4$ clusters.

2. Number of Clusters vs. Gap Statistic

Another way to determine the optimal number of clusters is to use a metric known as the gap statistic, which compares the total intra-cluster variation for different values of k with their expected values for a distribution with no clustering.

We can calculate the gap statistic for each number of clusters using the `clusGap()` function from the *cluster* package along with a plot of clusters vs. gap statistic using the `fviz_gap_stat()` function:

```
#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)
```



From the plot we can see that gap statistic is highest at $k = 4$ clusters, which matches the elbow method we used earlier.

Step 4: Perform K-Medoids Clustering with Optimal K

Lastly, we can perform k-medoids clustering on the dataset using the optimal value for k of 4:

```
#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"**

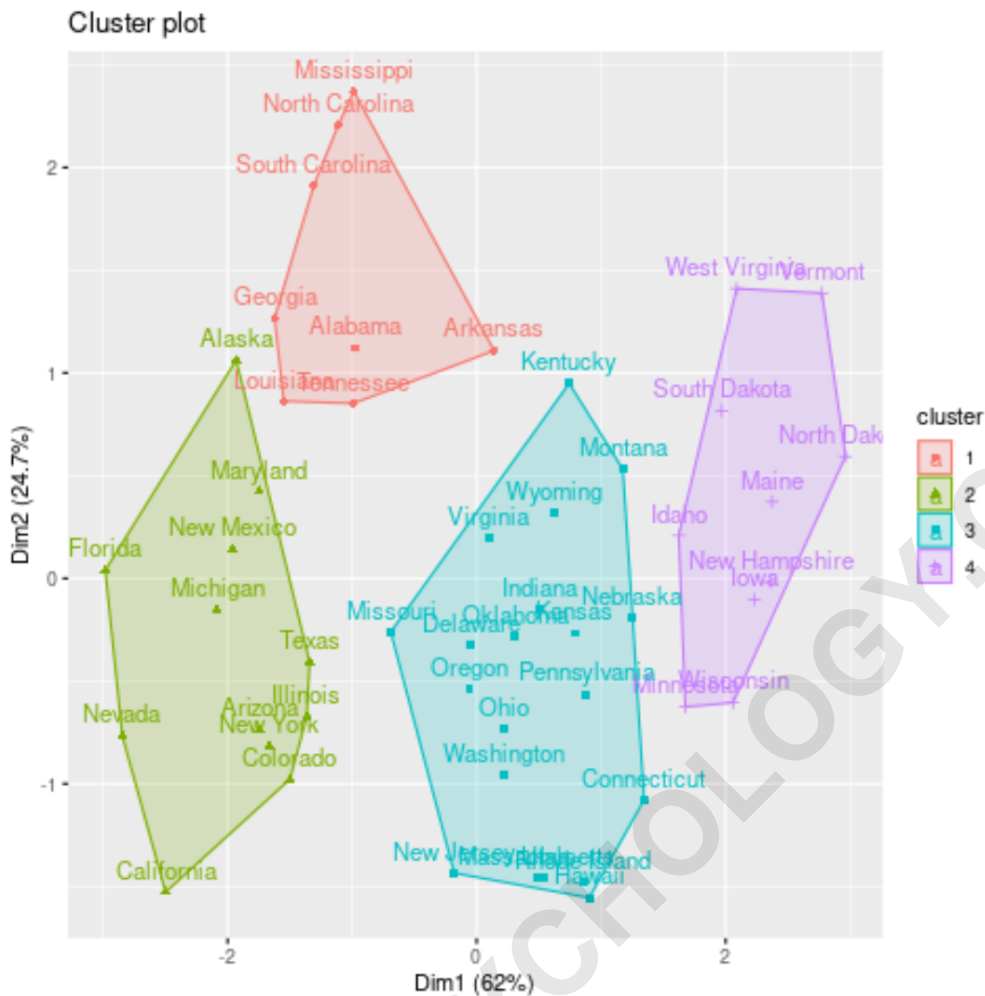
Note that the four cluster centroids are actual

observations in the dataset. Near the top of the output we can see that the four centroids are the following states:

AlabamaMichiganOklahomaNew Hampshire

We can visualize the clusters on a scatterplot that displays the first two principal components on the axes using the `fviz_cluster()` function:

```
#plot results of final k-medoids model  
fviz_cluster(kmed, data = df)
```



We can also append the cluster assignments of each state back to the original dataset:

```
#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

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

ARABPSYCHOLOGY.COM