



A menu icon consisting of three horizontal grey bars of decreasing length from left to right, followed by the word "MENU" in a small, uppercase, sans-serif font.

🕒 DECEMBER 2, 2020 🚩 BY ZACH

## K-Means Clustering in R: Step-by-Step Example

Clustering is a technique in machine learning that attempts to find **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**.

## What is K-Means Clustering?

K-means 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 simply the vector of the  $p$  feature means 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.

## K-Means Clustering in R

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

### Step 1: Load the Necessary Packages

First, we'll load two packages that contain several useful functions for k-means 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 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: Find the Optimal Number of Clusters

To perform k-means clustering in R we can use the built-in **kmeans()** function, which uses the following syntax:

**kmeans(data, centers, nstart)**

where:

- **data**: Name of the dataset.
- **centers**: The number of clusters, denoted  $k$ .
- **nstart**: The number of initial configurations. Because it's possible that different initial starting clusters can lead to different results,

it's recommended to use several different initial configurations.

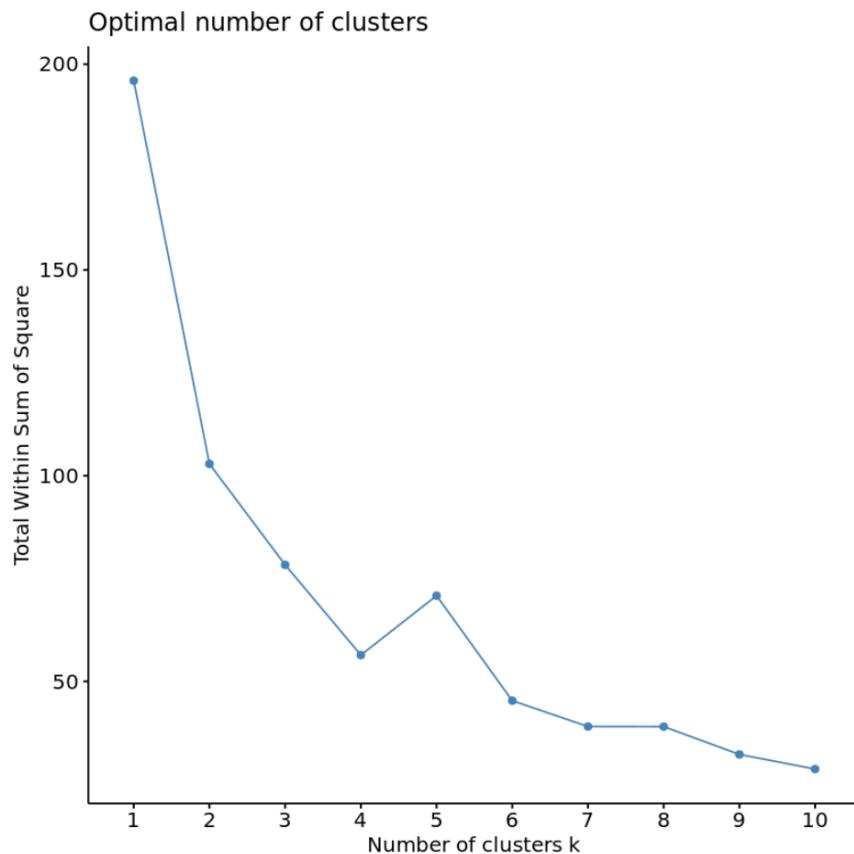
The k-means algorithm will find the initial configurations that lead to the smallest within-cluster variation.

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, kmeans, method = "wss")
```



Typically when we create this type of plot we look for an “elbow” where the sum of squares begins to “bend” or level off. This is typically the optimal number of clusters.

For this plot it appears that there is a bit of an elbow or “bend” at k = 4 clusters.

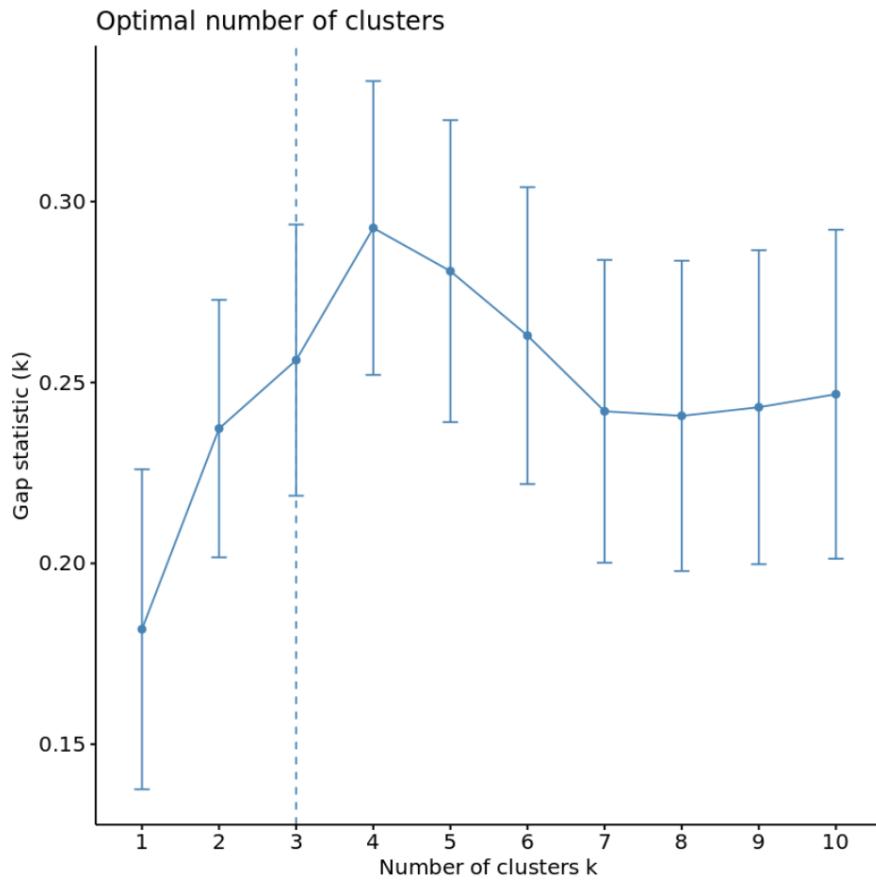
## 2. Number of Clusters vs. Gap Statistic

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 = kmeans,
                      nstart = 25,
                      K.max = 10,
                      B = 50)

#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-Means Clustering with Optimal $K$

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

```
#make this example reproducible
set.seed(1)

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

#view results
km

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

Cluster means:
Murder      Assault    UrbanPop          Rape
  5.09        5.88       59.57       21.23
```

```
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
4	3	3	4	
Connecticut	Delaware	Florida	Georgia	Hawaii
1	1	3	4	
Illinois	Indiana	Iowa	Kansas	Kentucky
3	1	2	1	
Maine	Maryland	Massachusetts	Michigan	Minnesota
2	3	1	3	
Missouri	Montana	Nebraska	Nevada	New Hampshire
3	2	2	3	
New Mexico	New York	North Carolina	North Dakota	Oregon
3	3	4	2	
Oregon	Pennsylvania	Rhode Island	South Carolina	South Dakota
1	1	1	4	
Texas	Utah	Vermont	Virginia	Washington
3	1	2	1	
Wisconsin	Wyoming			
2	1			

**Within cluster sum of squares by cluster:**

```
[1] 16.212213 11.952463 19.922437 8.316061
```

(between\_SS / total\_SS = 71.2 %)

**Available components:**

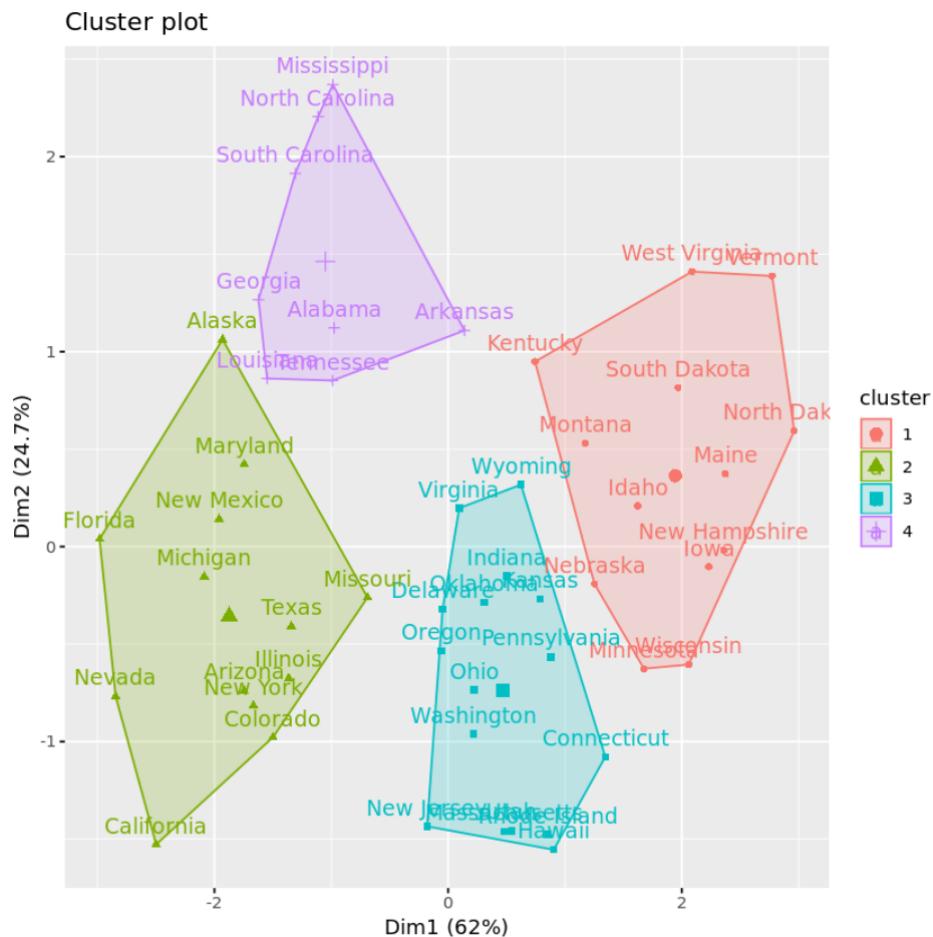
```
[1] "cluster"      "centers"       "totss"        "withinss"     "tot.wi·
[7] "size"         "iter"          "ifault"
```

From the results we can see that:

- **16** states were assigned to the first cluster
- **13** states were assigned to the second cluster
- **13** states were assigned to the third cluster
- **8** states were assigned to the fourth cluster

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-means model
fviz_cluster(km, data = df)
```



We can also use the **aggregate()** function to find the mean of the variables in each cluster:

```
#find means of each cluster
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

<b>3</b>	<b>5.65625</b>	<b>138.87500</b>	<b>73.87500</b>	<b>18.78125</b>
<b>4</b>	<b>13.93750</b>	<b>243.62500</b>	<b>53.75000</b>	<b>21.41250</b>

We interpret this output is as follows:

- The mean number of murders per 100,000 citizens among the states in cluster 1 is **3.6**.
- The mean number of assaults per 100,000 citizens among the states in cluster 1 is **78.5**.
- The mean percentage of residents living in an urban area among the states in cluster 1 is **52.1%**.
- The mean number of rapes per 100,000 citizens among the states in cluster 1 is **12.2**.

And so on.

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

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

#view final data
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 & Cons of K-Means Clustering

## K-means clustering offers the following benefits:

- It is a fast algorithm.
- It can handle large datasets well.

However, it comes with the following potential drawbacks:

- It requires us to specify the number of clusters before performing the algorithm.
- It's sensitive to outliers.

Two alternatives to k-means clustering are [k-medoids clustering](#) and [hierarchical clustering](#).

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



Published by Zach

[View all posts by Zach](#)

**PREV**

[How to Add a Numpy Array to a Pandas DataFrame](#)

**NEXT**

[How to Calculate Sample & Population Variance in R](#)

Leave a Reply