**ntroduction to Data Mining**

Clustering in R: K-Means

AUTHOR

AFFILIATION

Ivan G. Guardiola, PhD.

[Fort Lewis College](https://www.fortlewis.edu/)

**Clustering Algorithms**

Clustering algorithms can be categorized based on their cluster model, that is based on how they form clusters or groups. This tutorial only highlights some of the prominent clustering algorithms.

**Connectivity-based clustering:**

The main idea behind this clustering is that data points that are closer in the data space are more related (similar) than to data points farther away. The clusters are formed by connecting data points according to their distance. At different distances, different clusters will form and can be represented using a dendrogram, which gives away why they are also commonly called “hierarchical clustering”. These methods do not produce a unique partitioning of the data set, rather a hierarchy from which the user still needs to choose appropriate clusters by choosing the level where they want to cluster. They are also not very robust towards outliers, which might show up as additional clusters or even cause other clusters to merge.

**Centroid-based clustering:**

In this type of clustering, clusters are represented by a central vector or a centroid. This centroid might not necessarily be a member of the data set. This is an iterative clustering algorithms in which the notion of similarity is derived by how close a data point is to the centroid of the cluster. k-means is a centroid based clustering, and will you see this topic more in detail later on in the tutorial.

**Distribution-based clustering:**

This clustering is very closely related to statistics: distributional modeling. Clustering is based on the notion of how probable is it for a data point to belong to a certain distribution, such as the Gaussian distribution, for example. Data points in a cluster belong to the same distribution. These models have a strong theoretical foundation, however they often suffer from over-fitting. Gaussian mixture models, using the expectation-maximization algorithm is a famous distribution based clustering method.

**Density-based methods:**

Search the data space for areas of varied density of data points. Clusters are defined as areas of higher density within the data space compared to other regions. Data points in the sparse areas are usually considered to be noise and/or border points. The drawback with these methods is that they expect some kind of density guide or parameters to detect cluster borders. DBSCAN and OPTICS are some prominent density based clustering.

**Preliminaries**

We will reuse our data previously employed in the analysis of Linear Discriminant Analysis, however, this time we will use other algorithms to segment or cluster the data. First, lets start with the addition of some very valuable packages that are used for the purpose of clustering analysis.

Packages for Clustering:  
1. tidyverse: is very useful for all things to do with data manipulation.  
2. cluster: has an elaborate set of clustering algorithms.  
3. factoextra: clustering algorithms and visualization methods built into the package.  
4. gridExtra: provides number of user-level functions to work with “grid” graphics, meaning arranging multiple plots on a page. 5. FactoMiner: good package for dealing and analyzing factors.

Install these packages:

# un comment the lines of code below to install the packages

# install.packages('factoextra') install.packages('cluster')

# install.packages('gridExtra') install.packages('tidyverse')

# install.packages('FactoExtra)

#### load packages

library(cluster)

library(tidyverse)

library(factoextra)

library(FactoMineR)

library(fBasics) # previously installed in another session, but we will need it.

In many situations we will always want to check for whether or not we have missing data. In R, missing values are represented by the symbol NA (not available). The function is.na() will return a logical TRUE/FALSE vector if there are missing data. This should be done always. The reason to check for missing data is because many of the algorithms employed in clustering do not have the capability to remove NAs from your data. Recall, that the fBasic package has the capability to count this for us as an output, but only works for a single column. To determine if there any NAs in an entire data frame. Luckily for us there is a function that allows this. The function is.na.data.frame( your data frame ) will perform the test of NAs on an entire data frame. It will create a logical version of the original data frame with the same number of rows and columns.

Employee <- read.csv("EmployeeReview.csv", header = TRUE) # reads in the data

# check for NAs missing values

NAdf <- is.na.data.frame(Employee) # creates a data frame of logical vectors.

head(NAdf, 2) # shows the first two rows of this data frame

Obs. Group Mechanical Verbal

[1,] FALSE FALSE FALSE FALSE

[2,] FALSE FALSE FALSE FALSE

Now, we can use our old friend the any() function to check if there are any TRUE within the data frame.

any(NAdf) # checks to see if there are any TRUE in the data frame for NAs

[1] FALSE

The next step is to assure that all of the data has been scaled. However, this is only needed if the data to be used from clustering differs greatly from one variable to the next. For example, suppose that we had one variable that raged from 1-10000 and another that only ranges from 0-10. This will cause what we call “bias.” This is math, in math bigger numbers win when we maximize, and lose when the goal is to minimize. Since all clustering algorithms are doing one of these two actions, of minimizing or maximizing. We need to scale our data. This can be done quickly in R with the use of scale(). In our case the data are both exam scores, which we would expect to range from 0-100, thus, they are on similar scales. We do not need to scale the data.

**K-means Algorithm**

To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids.

It halts creating and optimizing clusters when either:

1. The centroids have stabilized — there is no change in their values because the clustering has been successful.
2. The defined number of iterations has been achieved.

**Data Preparation**

To perform a cluster analysis in R, generally, the data should be prepared as follows:

1. Rows are observations (individuals) and columns are variables.
2. Any missing value in the data must be removed or estimated.
3. The data must be standardized (i.e., scaled) to make variables comparable. Recall that, standardization consists of transforming the variables such that they have mean zero and standard deviation one.

**Performing K-means in R**

In R, we can run the K-means algorithm by simply using the kmeans() function. This is part of the built in stats within R. However, the packages we have loaded bring in the capability to visualize the results. Thus, to employ the K-means Algorithm within R we need to simply provide the columns of data we need to cluster and we need to emphasize the number of centers we want. These centers, are really how many groups we want the algorithm to split our data into, or segments of data. That is, since we have previously used these data before in other exercises, it is known that the data is split into two groups, Thus, we would want to emphasize to the algorithm that it should look for two centers, or two groups. However, unlike Linear Discriminant Analysis (LDA) where we use the known group column to derive a cut-off, the K-means is not a supervised learning method, meaning it will not use the group column in our data. This is a search algorithm, it will search for the best clusters that are the most tightly grouped according to minimization of distance between points, but also the maximization of the centers. Thus, we will need to explore the number of groups with the k-means. To specify we are wanting to split the data into two groups we will use the option centers=2 within the kmean( data , centers=2). This can also be done with kmeans(data, 2).

Dat <- Employee[3:4] # extracts the Mechanical and Verbal columns from the Employee data frame.

k2 <- kmeans(Dat, centers = 2) # runs the k means on the two columns

We have stored the output of the K-Means in the variable k2. In this variable now sits our results. The column cluster within k2 contains the predicted number. Lets add this column to our original data frame.

Employee$Result <- k2$cluster # store k2$cluster as the Result column in the Employee data frame

Employee # shows the first column of the data frame.

Obs. Group Mechanical Verbal Result

1 1 1 44.7 36.1 2

2 2 1 43.2 42.0 2

3 3 1 42.5 30.8 1

4 4 1 40.2 42.6 2

5 5 1 41.9 37.1 2

6 6 1 39.3 33.8 1

7 7 1 38.6 40.3 2

8 8 1 38.1 28.5 1

9 9 1 37.9 44.7 2

10 10 1 36.8 42.6 2

11 11 1 35.8 35.2 1

12 12 2 39.0 33.1 1

13 13 2 37.5 36.4 1

14 14 2 36.1 30.8 1

15 15 2 35.6 27.2 1

16 16 2 34.4 39.0 2

17 17 2 32.8 35.2 1

18 18 2 31.4 29.4 1

19 19 2 29.5 37.9 1

20 20 2 30.1 33.9 1

In some cases K-Means will flip the number used in the result column. Meaning, sometimes it will use different numbers than those specified. If this happens then we have a problem. It seems that the algorithm chose to flip the name or our group number. This is because we did not use the Group column to perform the classification. Thus, we will need to fix it seems that is choose to use group number 2 for group 1. It seems to be consistant on this as group 2 seems to be mostly made up of 1s we will need to flip this. *ONLY DO THIS IF IT FLIPPED THE NUMBERS IN OUR CASE THIS DID NOT HAPPEN.*

# fix the swapping in the name done by the algorithm. flip the numbers for the

# results column and store it in a new column called FixResult in Employee

Employee$FixResult <- ifelse(Employee$Result == 1, 2, 1)

Employee

Obs. Group Mechanical Verbal Result FixResult

1 1 1 44.7 36.1 2 1

2 2 1 43.2 42.0 2 1

3 3 1 42.5 30.8 1 2

4 4 1 40.2 42.6 2 1

5 5 1 41.9 37.1 2 1

6 6 1 39.3 33.8 1 2

7 7 1 38.6 40.3 2 1

8 8 1 38.1 28.5 1 2

9 9 1 37.9 44.7 2 1

10 10 1 36.8 42.6 2 1

11 11 1 35.8 35.2 1 2

12 12 2 39.0 33.1 1 2

13 13 2 37.5 36.4 1 2

14 14 2 36.1 30.8 1 2

15 15 2 35.6 27.2 1 2

16 16 2 34.4 39.0 2 1

17 17 2 32.8 35.2 1 2

18 18 2 31.4 29.4 1 2

19 19 2 29.5 37.9 1 2

20 20 2 30.1 33.9 1 2

To visualize the result we can use the fviz\_cluster() function within the factoextrapackage. This is a great tool for a variety of clustering function results, but not all. Thus, depending on the algorithm it may work or it may not. Some it will work for are pam, kmeans, clara, fanny, dbscan, HCPC, Mclust, hkmeans. These are all various algorithms from multiple packages. There are so many algorithms there is no one algorithm to rule them all. It is a lot of trial and error. However, wwith experience you will learn which work best for the type of data you are trying to cluster.

# use fviz\_cluster to view clusters

fviz\_cluster(k2, data = Employee[4:3]) # you must provide the data as well as the result from kmeans.

Chart

Description automatically generated

Notice that the function not only provides the data points acording to their row number, but also the centers and creates a legend for the graph. In this case we see that K-means works well.

However, lets scale the data to determine if it works any better.

k2Scaled <- kmeans(scale(Employee[3:4]), 2)

fviz\_cluster(k2Scaled, data = Employee[3:4])

Chart

Description automatically generated

Thus, we can see a slight difference in the results. To determine which of the clusterings is better. We often look at the variability measures.

The output provided by the k-means function are:

* *cluster*: A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
* *centers*: A matrix of cluster centers.
* *totss*: The total sum of squares.
* *withinss*: Vector of within-cluster sum of squares, one component per cluster.
* *tot.withinss*: Total within-cluster sum of squares, i.e. sum(withinss).
* *betweenss*: The between-cluster sum of squares, i.e. totss−tot.withinsstotss−tot.withinss.
* *size*: The number of points in each cluster.

Thus lets obtain the tot.withins for both clustering. The original non scaled data results in a total within cluster sum of squarestot.withins of 443.1545833, and the scaled version in tot.withins of 21.5277822. Thus, we can see that when we scaled the data the resulting variability was scaled. However, in our case we will need to look at sum of squares to determine when we have the best clustering.

**Determining the Number of Groups**

K-Means requires that we determine the best grouping as it is not a supervised method. Recall, a supervised method is one that would use our group column to learn the group participants. K-means simply applies an analysis of distance.

df <- Employee[3:4] # extract the columns Verbal and Mechanical

k3 <- kmeans(df, centers = 3, nstart = 25) # run 3 groups

k4 <- kmeans(df, centers = 4, nstart = 25) # run 4 groups

k5 <- kmeans(df, centers = 5, nstart = 25) # run 5 group

# plots to compare

p1 <- fviz\_cluster(k2, geom = "point", data = df) + ggtitle("k = 2")

p2 <- fviz\_cluster(k3, geom = "point", data = df) + ggtitle("k = 3")

p3 <- fviz\_cluster(k4, geom = "point", data = df) + ggtitle("k = 4")

p4 <- fviz\_cluster(k5, geom = "point", data = df) + ggtitle("k = 5")

library(gridExtra)

grid.arrange(p1, p2, p3, p4, nrow = 2)

Here we see that we have segmented the data into many groups. We will have to generate a scree plot. The scree plot allows us to look at the silhouette distance to determine when we have the best clustering.

# function to compute average silhouette for k clusters

avg\_sil <- function(k) {

km.res <- kmeans(df, centers = k, nstart = 25)

ss <- silhouette(km.res$cluster, dist(df))

mean(ss[, 3])

}

# Compute and plot wss for k = 2 to k = 55

k.values <- 2:5

# extract avg silhouette for 2-15 clusters

avg\_sil\_values <- map\_dbl(k.values, avg\_sil)

# plot the results

plot(k.values, avg\_sil\_values, type = "b", pch = 19, frame = FALSE, xlab = "Number of clusters K",

ylab = "Average Silhouettes")

Chart, line chart

Description automatically generated

It seems that the tightest clustering occurs when we have 4 groups. Thus, according to the result there are 4 groups within this data. Another way to do this is to use the “Elbow Method”. Fortunately, this process to compute the “Elbow method” and “silhouette” has been wrapped up in a single function (fviz\_nbclust), which is from the factoextra package.

fviz\_nbclust(df, kmeans, method = "wss") # elbow method

fviz\_nbclust(df, kmeans, method = "silhouette") # silhouette method

You can see that the two methods differ there therefore we will need to look at other measures. However, once you have determines the optimal number of clusters then we can generate a confusion matrix.

out <- table(Employee$Group, Employee$Result) # table summary of group and Result column from kmeans

library(caret) # loads package needed for confusionMatrix function

library(lattice) # required for caret functions

confusionMatrix(out)

Confusion Matrix and Statistics

1 2

1 4 7

2 8 1

Accuracy : 0.25

95% CI : (0.0866, 0.491)

No Information Rate : 0.6

P-Value [Acc > NIR] : 0.9997

Kappa : -0.5306

Mcnemar's Test P-Value : 1.0000

Sensitivity : 0.3333

Specificity : 0.1250

Pos Pred Value : 0.3636

Neg Pred Value : 0.1111

Prevalence : 0.6000

Detection Rate : 0.2000

Detection Prevalence : 0.5500

Balanced Accuracy : 0.2292

'Positive' Class : 1

You can see that K-Means is not very accurate with this particular data set since we know the actual grouping prior to analysis. Thus, when we do have group information a method that uses the information will always be best. Meaning, if we have something to learn from then supervised methods will always be better. However, K-means is a very powerful method when we do not know this prior group classification. It does very well for numerical data

**Conclusion:**

Kmeans is a great method when we don’t know the number of groups in the data, meaning we need an unsupervised method. It requires the data be segmented many times then using the silhouette method or elbow method to determine the optimal number of groups. If we have grouping information other model based clustering may be better.