Cluster Analysis In R

Luke T. Daniels

4/1/2018

Cluster analysis in R is a great tool to find trends and patterns in data. Many people think of statistics as giving definitive answers to questions, however, there are techniques that simply provide further insight into data. Clustering allows users to identify which observations are alike across many different variables. The power of cluster analysis allows us to perform complex analyses that would be near impossible without programs such as R.

This demonstration serves as an a brief introduction to the statistics behind cluster analysis and the corresponding tools in R. I will highlight the differences in base `R` and packages such as `factoextra` and `cluster`. First I will begin my introducing K-means clustering

** Decide on the Clustering Variables**

Cluster analysis has no mechanism for differentiating between relevant and irrelevant variables. There must be careful consideration for the variables included in the analysis. There should be signficant differences between the variables. There are select pricipals that should be followed. 1.) Avoid using an abundance of variables since this increases the odds that variables are no longer dissimilar. 2.) If there is a high degree of collinearity, specific aspects covered by these variables will be overrepresented. 3.) Formann (1984) recommends a sample size of at least 2^m where m equals the number of clustering variables

Data Preparation

1.) Rows must be observations and columns must be variables. 2.) Missing values must be removed or estimated. 3.) The data must be standardized to make variables comparable. 4.) The data used in cluster analysis can be interval, ordinal or categorical * There are different methods / calculations that are better for dealing with each type of data type.

```
data("USArrests")
df <- USArrests
df <- na.omit(df) #Removes any missing values
df.scaled <- scale(df)  #We standardize the data using the function scale()
```

Standardization

One very important decision that needs to be made involves the scales of the variables being measured. If one of the variables is measured on a much larger scale than the other variables, then whatever measure is used will be overly influenced by that variable. For example, if we are looking at the distance between two people based on their IQs and incomes in dollars, the differences in incomes would dominate the distance measurments. We can solve this issue by standardizing the variables! bo

Required R Packages

1.) **cluster**: Computes clustering algorithms

2.) **factorextra**: Used for ggplot2 visualization of clustering results

```
library(ggplot2)
library(factoextra)
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
library(cluster)
library(latexpdf)
library(purrr)
```

Approach #1: Partitioning Methods (K-Means)

Basically K- mean is an interative process that divides a given data set into K disjoint groups

It starts by placing k centroids randomly in your space. Then you iteraviley do the following: First we run through our data set and for each individual we find the nearest centroid to that individual. To do that, for each x_i you compute the distance between in a c_j (each cluster). This is the Eucleadian Distance. Then you pick the cluster that has the minimum distance of all (nearest cluster).

Then you assign x_i to that nearest cluster. This process occurs for each individual, so each individual minimizes is distances to the randomly positioned centroid. Now we need to recompute the centroid by getting the average of all the x_i that was assigned to that cluster. (All the x_i's that were assigned to the jth cluster and you average them out.)

At this point we are restricting the analysis to continous variables. We cannot take the average or distance of categorical variables.

These are the two basic steps. You keep running theses steps until no individuals change cluster memberships

Distance Measures

K means is implicitly based on pairwise Euclidean distances because the sum of squared deviations from the centroid is equal to the sum of pairwise squared Euclidean distances divided by the Number of Points. A centroid is a multivariate mean in euclidiean spaces. Euclidean spaces is about euclidean distances.

The classification of items into groups requires computation od the distance or similarity between each observation. Distance measures defines how similar two elements (x,y) are and will influence the shape of the clusters.

There are many methods to calculating the distance between observations. The most popular method is Euclidean Distance. An important note is that the *Euclidean* method is often considered the best for dealing with interval data.

1. Euclidian Distance:

$$deuc(x,y) = \sum_i = \ln(x_i - y_i)2d(x,y) = \sqrt{(\sum_{i=1}^{n}(x_1 - y_1)^2)}$$

Other measures, such as the Pearson Equation use correlation based methods.

2. Pearson Correlation Distance

$$dcor(x,y) = 1 - \sum_{i} = \ln(xi-x)(yi-y) \sum_{1} = \ln(x1-x)2 \sum_{i} = ln(y_i-y)2 d_{cor}(x,y) = 1 - \frac{\sum_{i=n}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{(\sum_{i=1}^{n}(x_i-\bar{x}^2 \sum_{i=1}^{n}(y_i-\bar{y}^2)}}$$

A correlation based distance considers objects similar if their features are highly correlated even if the Euclidean distance is not. If there are observations with the same profiles but with different magnitudes - use correlation-based.

In a Euclidean analysis, observations with high values will be clustered together.

Other methods to consider: Manhattan distance, Spearman Correlation, Kendall Correlation

Caluclating Euclidean Distance in R

Function 1 Daisy

- The *daisy* function in `cluster` handles different variable types (binary, ordinal, nominal) and computes a distance matrix

- In the daisy function we can indicate which distance measurment equation we would like to use. "Gower" is best where the data contain non-numeric columns.

- The metric will be the *Gowers Coefficient*
- It indicates the % average similarity between all pairs of observations

daisy(x, metric = c("euclidean", "manhattan", "gower"), stand = FALSE, type = list(), weights = rep.int(1, p), warnBin = warnType, warnAsym = warnType, warnConst = warnType, warnType = TRUE)

The values in the matrix represent the distance between objects. A value of zero indicates that the two items are not different.

```
disteucl <- daisy(df.scaled, metric = "gower", stand= FALSE)
disteucl

euclmatrix <- as.matrix(disteucl)
```

We have now created a distance matrix!

```
head(euclmatrix)
```

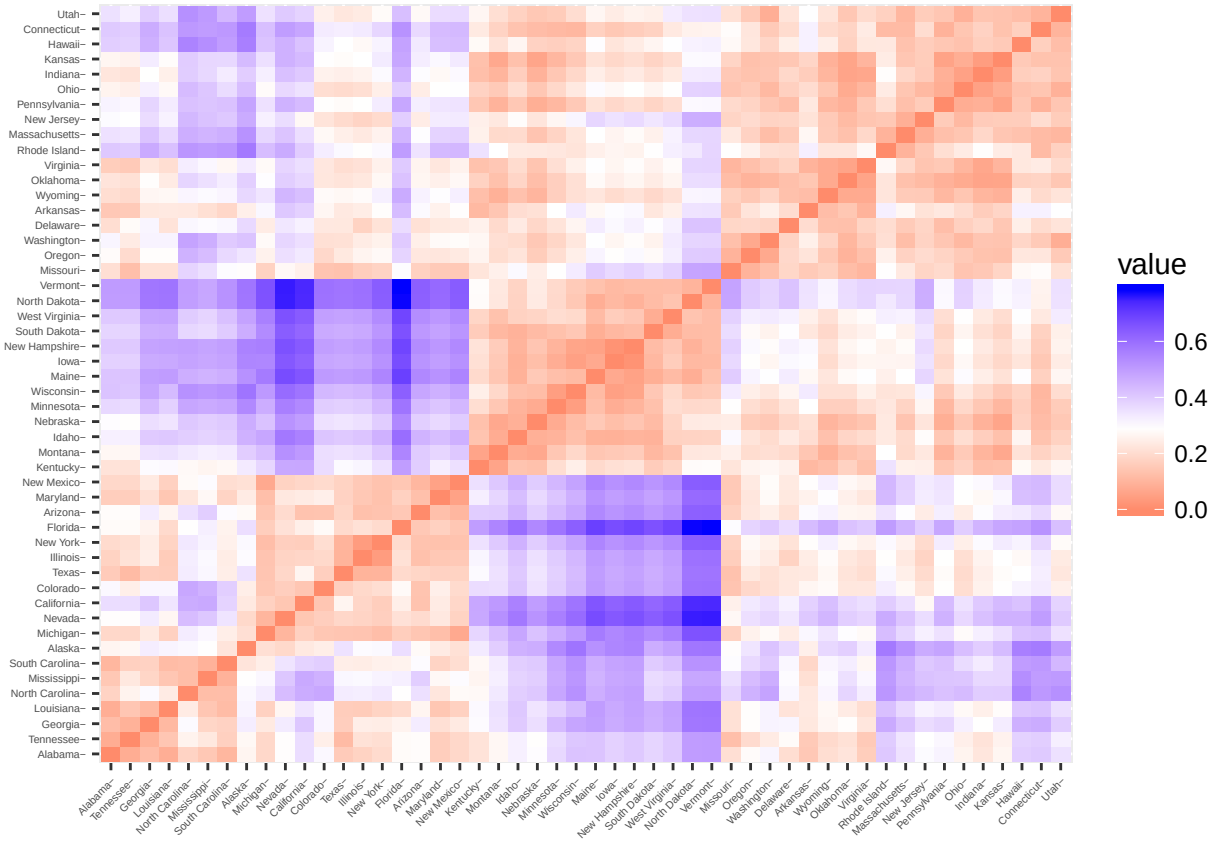
Fuction 2 fviz_dist()

An easy way to vizualize a distance matrix is to use fviz_dist() from the `factoextra` package

fviz_dist(dist.obj, order = TRUE, show_labels = TRUE, lab_size = NULL, gradient = list(low = "red", mid = "white", high = "blue"))

The Value indicates the level of dissimilarity. Zero means the two are exactly similar.

```
fviz_dist(disteucl, order = TRUE, show_labels = TRUE, lab_size = 4, gradient = list(low = "red", mid =
```



K-Means Clustering ~ Assigning Clusters

Main Idea:

Define clusters so that the total intra-cluster variation is minimized. There a many k-means algorithms available but the most commonly used is the Hartigan-Wong algorithm. This equation defines the total within cluster variation as the sum of squared distances Euclidean distances between items and the corresponding centroid

Total Within Cluster Sum of Squares Equation

$$tot.withinss = \sum_{k} = 1kW(Ck) = \sum_{k} = 1k \sum_{x} \epsilon Ck(x_i - \mu k)2tot.withinss \sum_{k=1}^{k} W(C_k) = \sum_{k=1}^{k} \sum_{x_i \in C_k} (x\_\mu_k)^2$$

Each observation of xi_k is assigned to a cluster so that the sum of squares distance of the observation to their assigned cluster centers mu_k is minimized This equation measures the compactness of the clustering - we want it to be minimized.

The Hartigon Wong algorithm requires the user to estimate the value of k clusters.

Hartigon Wong Equation In Action!

```
kmeans(df.scaled, centers = 3, iter.max = 10, nstart = 1)
```

The algorithm starts by selecting k objects from the data set to serve as the intitial centers. These are known as centroids. Each object is then assigned to the closes centroid defined by the Euclidean distance. After this assignment, the algorithm computes the new mean value of each cluster. Every observation is checked again to see if it is actually closer to another cluster. This occurs until the cluster assignments stop changing.

The main problem is that if we choose the wrong k, then the whole analysis is wrong! Human error!

How Do We Choose the Right Number of K Clusters?

There are three methods to determine the optimal clusters:

1.) Elbow Method 2.) Silhouette Method 3.) Gap Statistic.

Elbow Method

Clusters must be defined so that the total within cluster variation is minimized

$$minimize(\sum_{k=1}^{k} W(C_k)$$

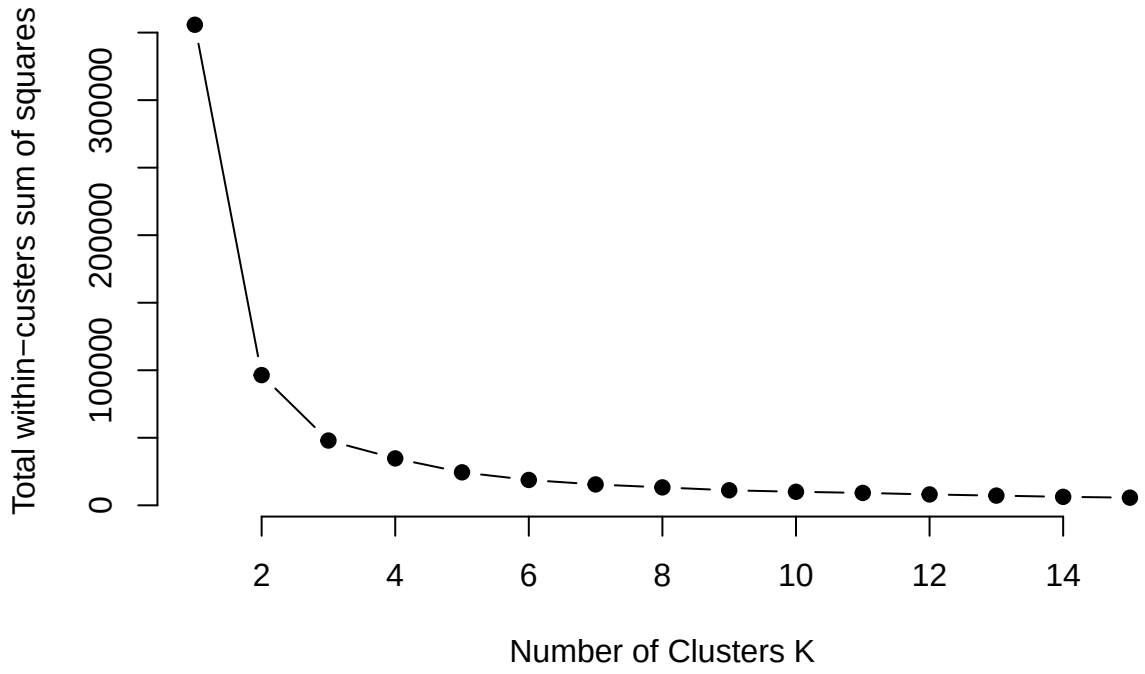
is the kth cluster and is the within cluster variation.

The Elbow Method Using Base R

```
set.seed(123)
# function to compute the total within-cluster sum of squares
k.max <- 15
wss <- sapply(1:k.max, function(k){kmeans(df,k, nstart = 50)$tot.withinss})
wss
```

```
##  [1] 355807.822  96399.028  47964.265  34728.629  24417.024  18768.001
##  [7]  15463.474  13259.146  11125.034  10021.246   9180.964   8043.326
## [13]   7174.606   6273.413   5667.766
```

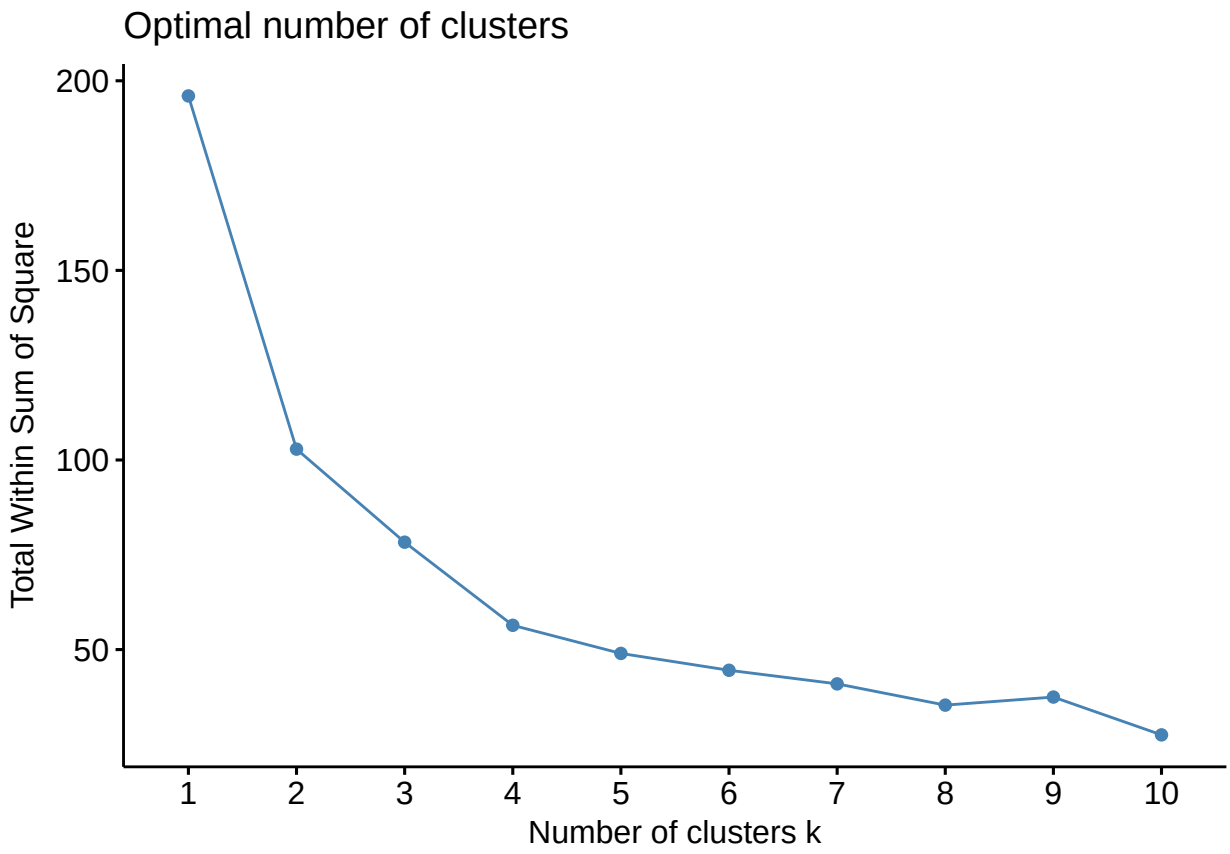
```
plot(1:k.max, wss, type = "b", pch= 19, frame = FALSE,
     xlab = "Number of Clusters K",
     ylab = "Total within-custers sum of squares")
```



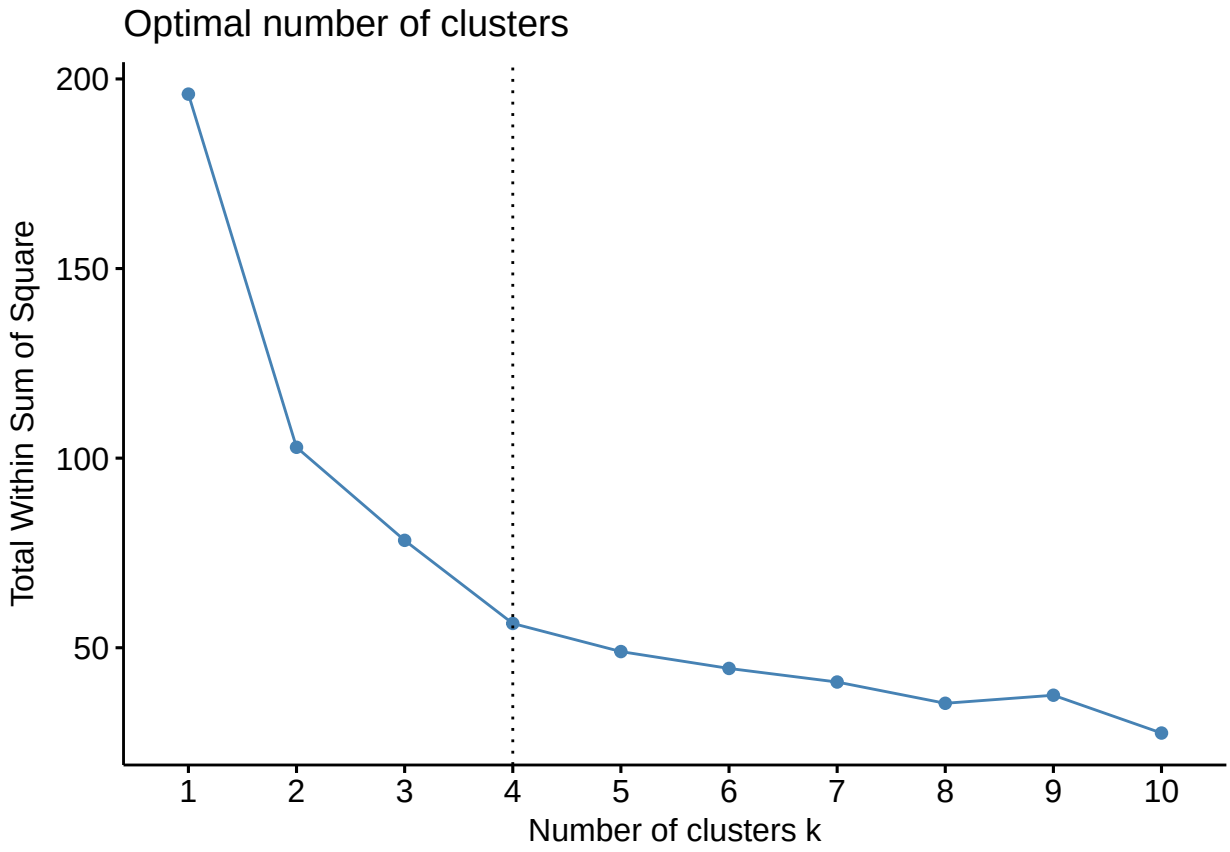
The Elbow Method in Factoextra

We rely on the 'factoextra' package once again. The "bend in the knee" in the graph below is considered to be the appropriate number of clusters

```
fviz_nbclust(df.scaled, kmeans, method = "wss")
```




```
fviz_nbclust(df.scaled, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 3)
```



**** Silhouette Method****

This approach measures the quality of clusters. It determines how well each object lies within the cluster. This algorithm computes the average silhouette of observations for different values of k. The optimal number of cluster k is the one that maximizes the average silhouette over a range of possible values

In base R

```
#Function to compute the average silhouette for k cluseter


#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 =15
k.values <- 2:15
```

```
#extracting silhouettes
#avg_sil_values <- map_dbl(k.values, avg_sil)

#plot(k.values, avg_sil_values,
#type = "b", pch = 19, frame = FALSE,
#xlab = "Number of Cluster K",
#ylab = "Average Silhouettes")
```

There will be an easier way of calculating silhouettes later in this demonstration!

Computing the Means with K = 4 Clusters

Remember, using the k-means approach, the center of a cluster is defined as being the average of all the points within a cluster!

```
set.seed(100)  # Since the alogorithm starts with k randomly selected centers we must set the seed
km.res <- kmeans(df, 4, nstart = 50) # R will try 50 different random starting assignments and then sel
km.res
```

```
## K-means clustering with 4 clusters of sizes 10, 14, 10, 16
##
## Cluster means:
##      Murder  Assault UrbanPop     Rape
## 1  2.950000  62.7000 53.90000 11.51000
## 2  8.214286 173.2857 70.64286 22.84286
## 3  5.590000 112.4000 65.60000 17.27000
## 4 11.812500 272.5625 68.31250 28.37500
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              4              4              4              2              4
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              4              4              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              1              3              4              3              1
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              4              1              4
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              2              4              1              4              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              3              3              4              1              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              4              4              4              1              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              2              2              3              2              4
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              1              2              2              3              1
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              2              2              1              1              2
##
## Within cluster sum of squares by cluster:
## [1]  4547.914  9136.643  1480.210 19563.863
```

```
##  (between_SS / total_SS =  90.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Now we have classified each unit by their respective cluster. This model has a good fit (90.2%)

Now that each unit has been assigned a cluster, we can calculate the cluster means

```
aggregate(USArrests, by=list(cluster= km.res$cluster),mean)
```

```
##   cluster    Murder  Assault UrbanPop     Rape
## 1       1  2.950000  62.7000 53.90000 11.51000
## 2       2  8.214286 173.2857 70.64286 22.84286
## 3       3  5.590000 112.4000 65.60000 17.27000
## 4       4 11.812500 272.5625 68.31250 28.37500
```

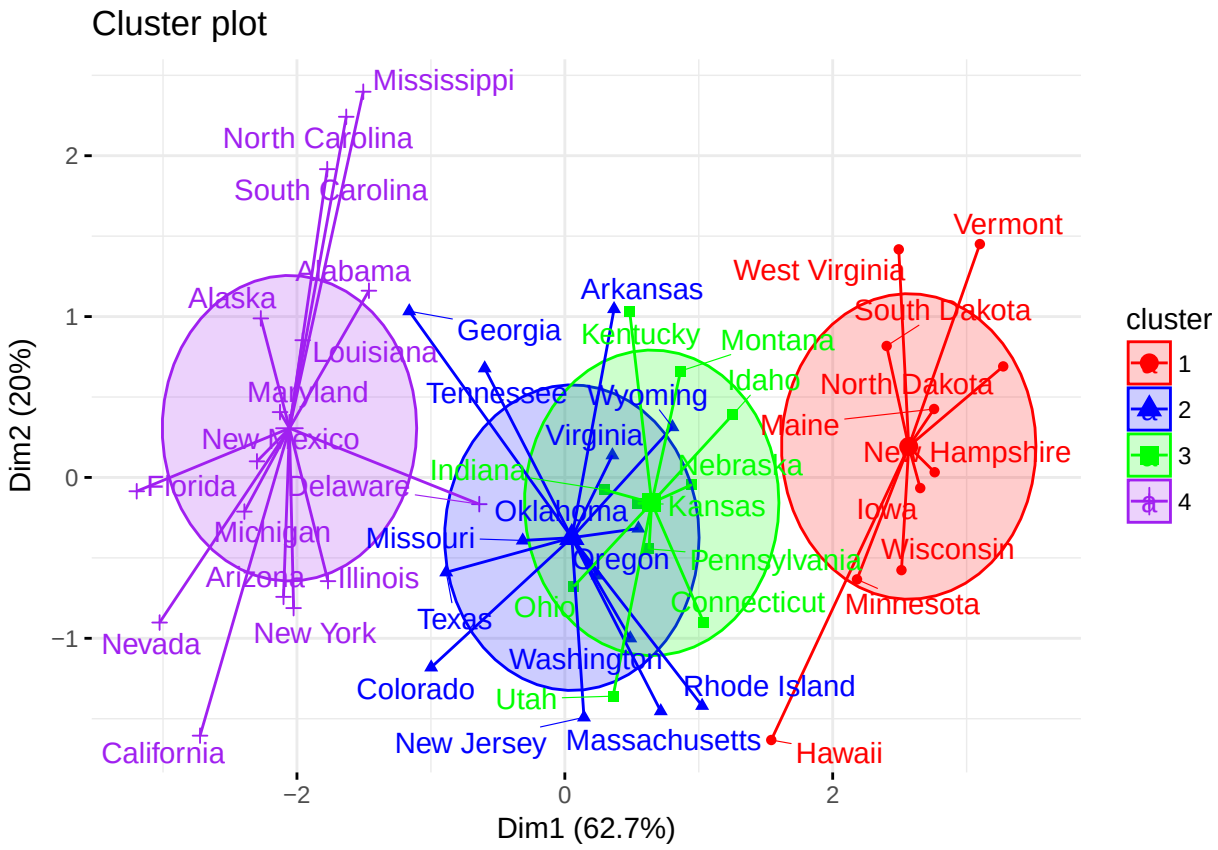
We can add the cluster assignments to the original data frame

```
df <- cbind(USArrests, cluster = km.res$cluster)
head(df)
```

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

Finally, we can vizualize

```
fviz_cluster( km.res, data = df, palette = c("Red", "Blue", "Green", "Purple"),
              ellipse.type = "euclid", star.plot = TRUE,
              repel = TRUE,
              ggtheme = theme_minimal())
```



The PAM Approach

In the K-mean approach above, the center of a cluster is calculated as the mean value of all the data points in a cluster. This makes the K-means approach very sensitive to outliers and noise.

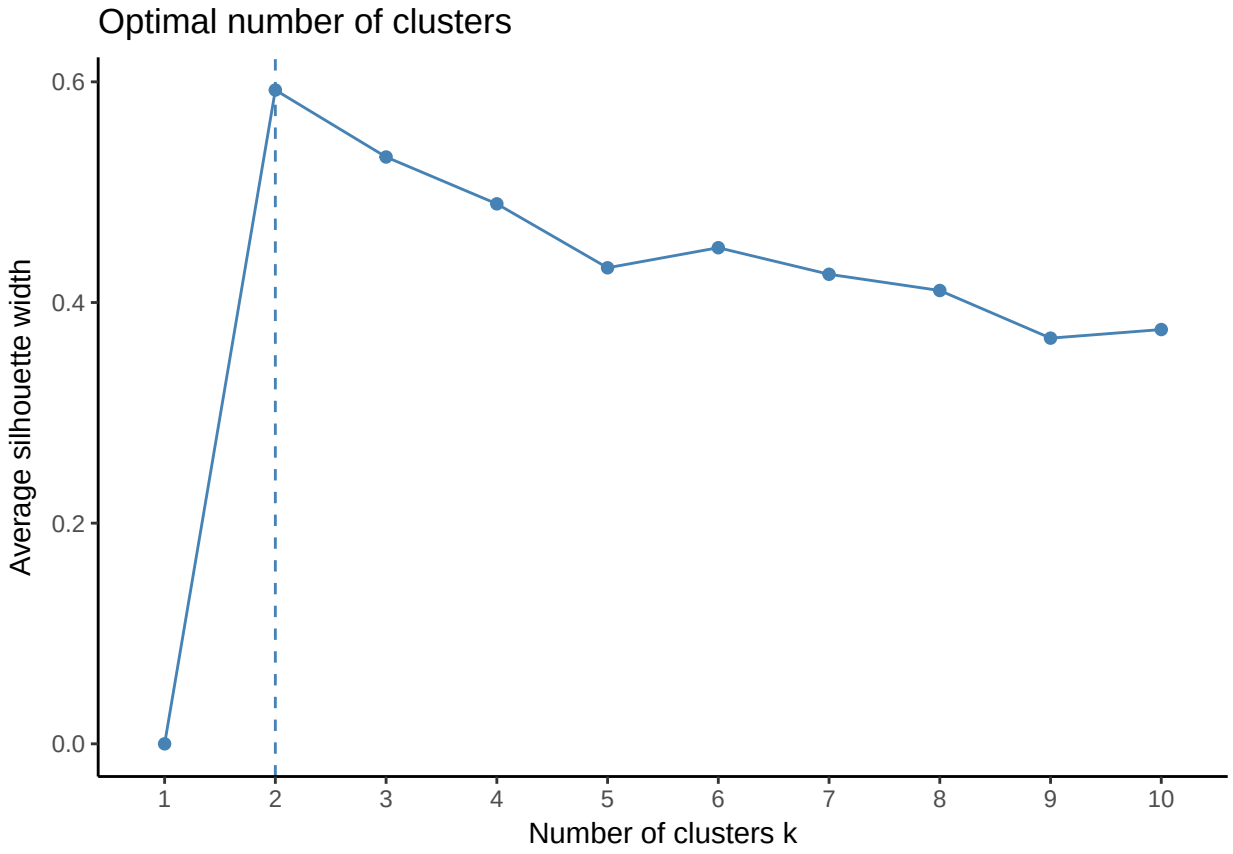
The K-mediod approach using the PAM algorithm is more robust and therefore less sensitive to outliers. A mediod is an object within a cluster for which the average dissimilarity between it and all the other objects in a cluster is minimal. (The most centrally located point). This point is representative of the cluster and less skewed by outliers.

1.) **Estimate K- Clusters Using Silohette Method in "factoextra"**

The idea of using the silhoutte method is to compute PAM using different values of clusters k.

To estimate the optimal number of clusters, we'll use the average silhouette method. The idea is to compute PAM algorithm using different values of clusters k. Next, the average clusters silhouette is drawn according to the number of clusters. The average silhouette measures the quality of a clustering. A high average silhouette width indicates a good clustering. The optimal number of clusters k is the one that maximize the average silhouette over a range of possible values for k.

```
fviz_nbclust( df, pam, method = "silhouette") + theme_classic()
```



- The maximum silhouette width occurs at k=2, suggesting that the optimal number of clusters is 2!

Computing the PAM Clustering

Function 3 pam()

pam(x, k, metric = "euclidean", stand = FALSE) * x is the disimilarity matrix created from the daisy() function * k is the number of clusters * stand is logical value, if true the columns in x were standardized before calculating disimilarities. This is ignored when x is a disimilarity matrix.

```
pam.res <- pam(df, 2)
pam.res
```

```
## Medoids:
##          ID Murder Assault UrbanPop Rape cluster
## Michigan 22   12.1     255       74 35.1       4
## Kansas   16    6.0     115       66 18.0       3
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              1              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              1              2              1              1              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              2              2              1              2              2
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              2              2              1              2              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
```

```
##              2              1              2              1              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              2              2              1              2              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              1              1              1              2              2
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              2              2              2              2              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              2              1              1              2              2
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              2              2              2              2              2
## Objective function:
##    build     swap
## 46.11127 38.43094
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

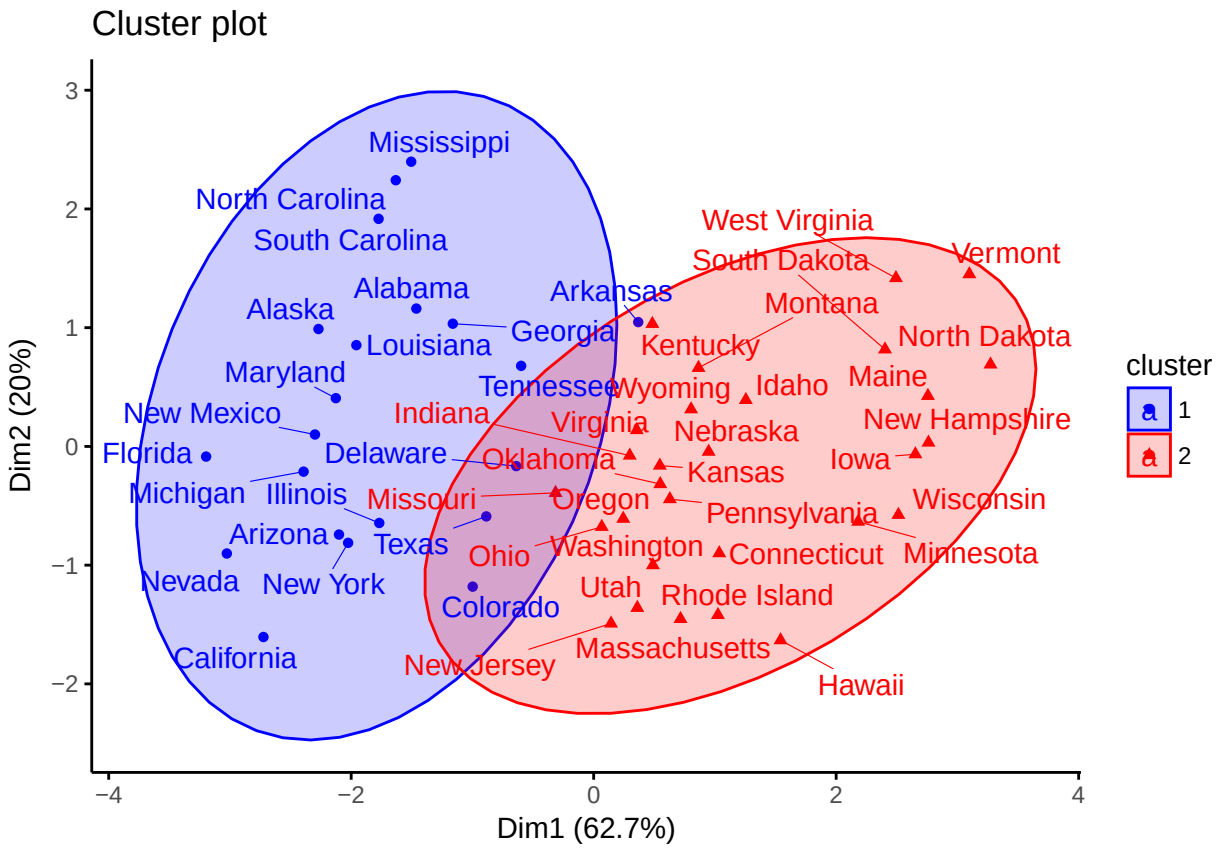
The pam() function returns the mediod objects! To access these:

```
pam.res$medoids
```

```
##          Murder Assault UrbanPop Rape cluster
## Michigan   12.1     255       74 35.1       4
## Kansas      6.0     115       66 18.0       3
```

Finally we can plot the 2 new clusters!

```
fviz_cluster( pam.res, palette = c("blue", "red"),
              ellipse.type = "t",
              repel = TRUE,  # Avoids overplotting labels
              ggtheme = theme_classic() )
```



Hierarhical Clustering

In contrast to partitioning clustering, hierarhical clustering does not require pre-specification of clusters!

Method 1: Agglomerative Clustering