

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. 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

Prepping Data

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.

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

Required R Packages

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

2.) **factoextra**: 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)
```

Distance Measures

The classification of items into groups requires computation of 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.

1. Euclidian Distance:

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

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

2. Pearson Correlation Distance

$$dcor(x, y) = 1 - \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}} = 1 - \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

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

Calculating 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 measurement 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)
```

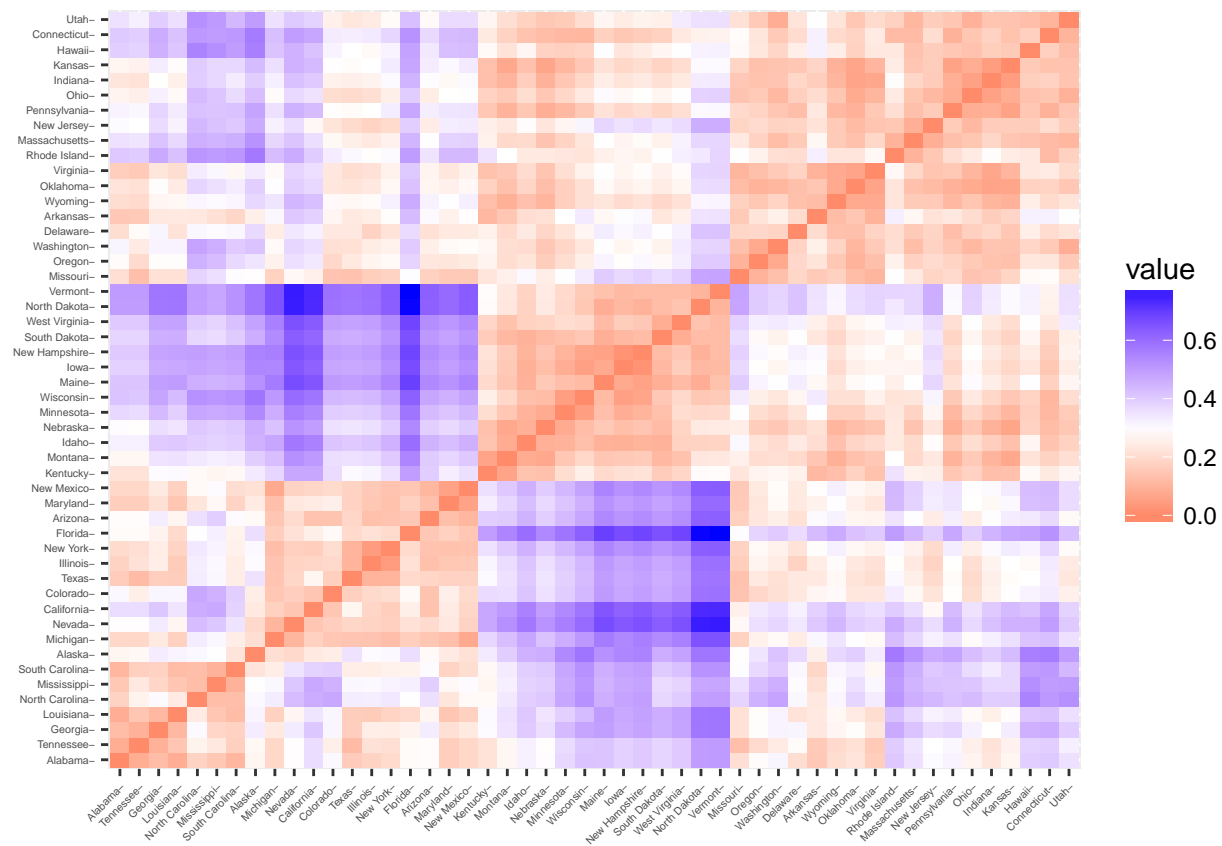
Fuction 2 fviz_dist()

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-Mean Clustering

Main Idea:

Define clusters so that the total intra-cluster variation is minimized. There are 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

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

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

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

Hartigan 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 initial centers. These are known as centroids. Each object is then assigned to the closest 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

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

is the kth cluster and is the within cluster variation.

The Elbow Method Using Base R

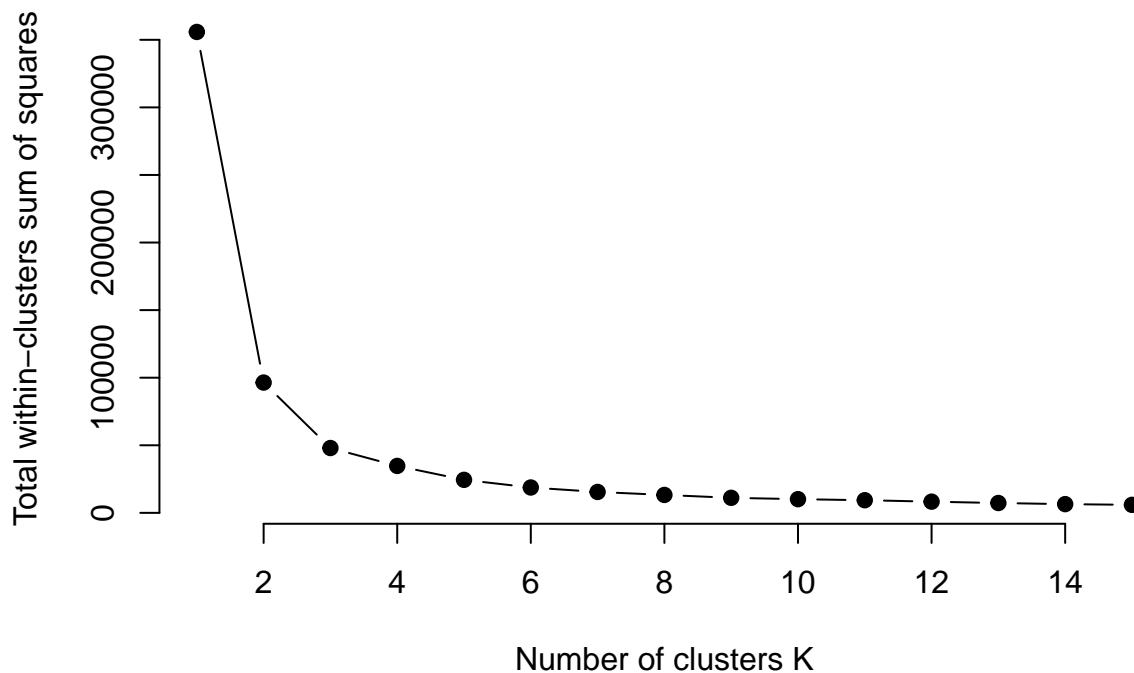
```
set.seed(123)
# function to compute the total within-cluster sum of squares
wss <- function(k){kmeans(df, k, nstart = 10)$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15

# Extract wss for 2- 15 clusters and plot

wss_values <- map_dbl(k.values, wss) # The mapl function transforms input by applying a function to each element

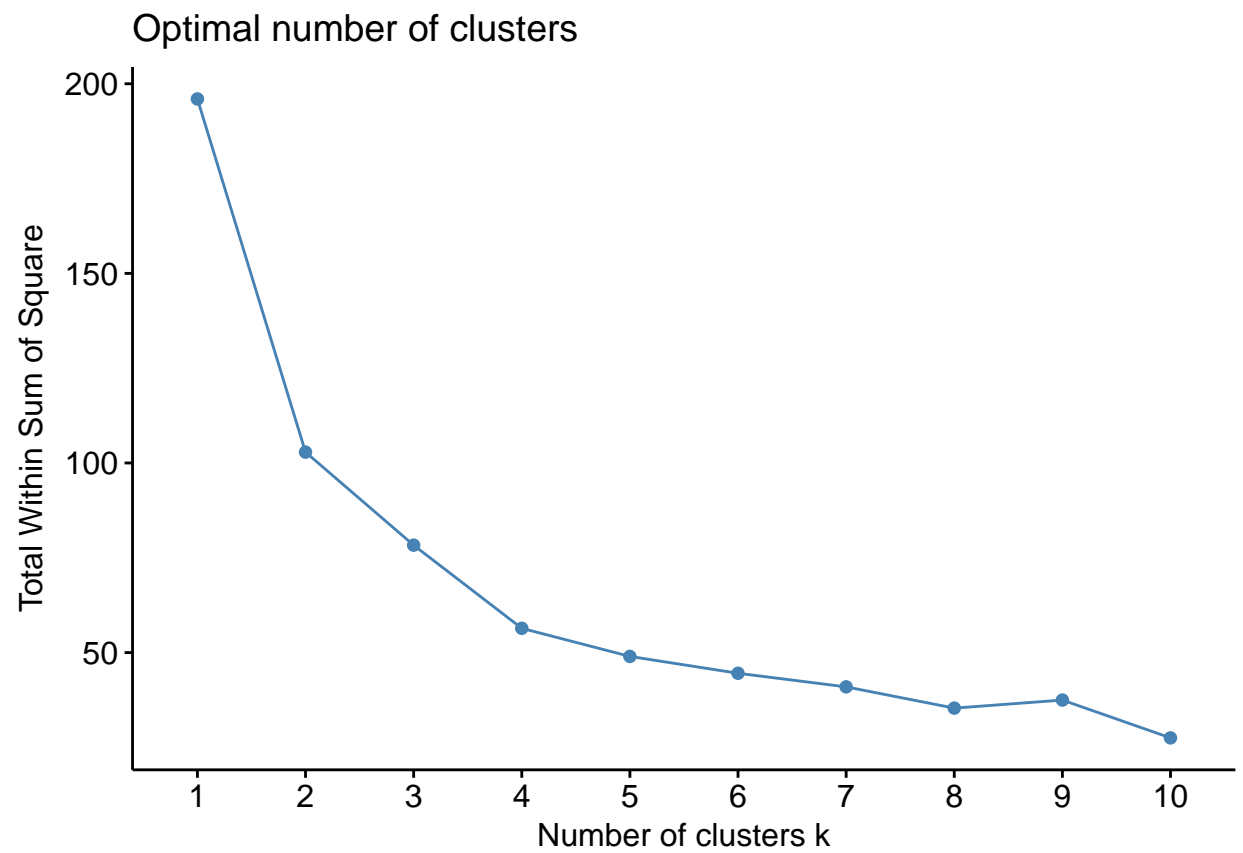
plot(k.values, wss_values, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Total within-clusters sum of squares")
```



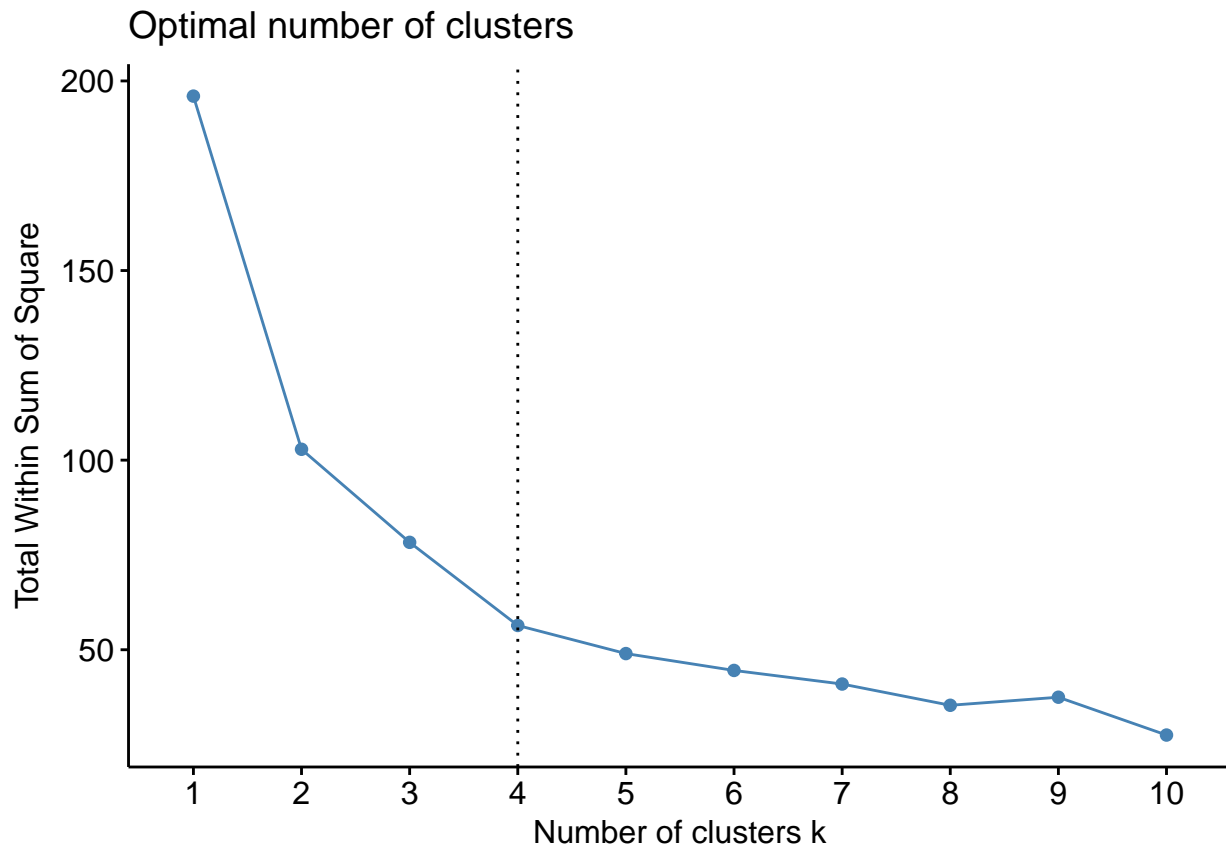
The Elbow Method in Factoextra

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 cluser
```

```
#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
```

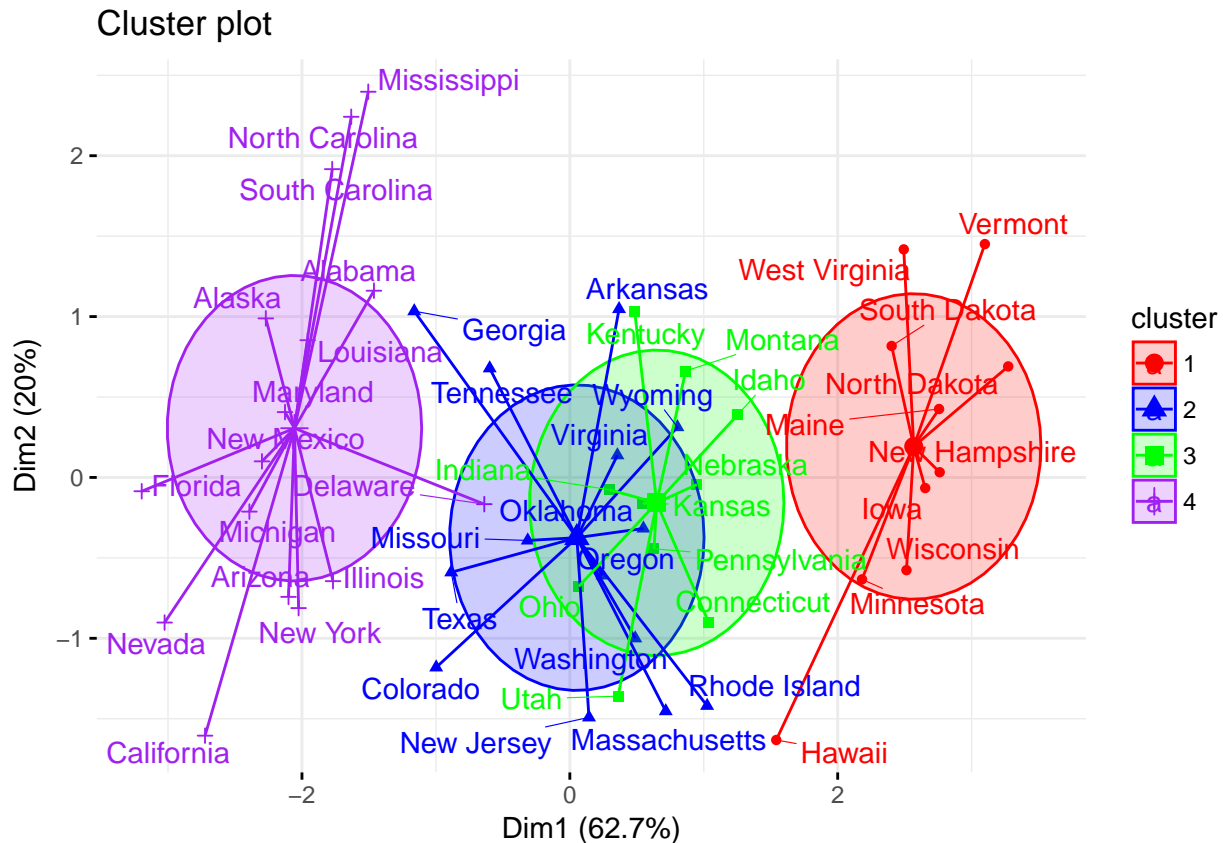
We can add the cluster assignments to the original data frame

```
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 visualize

```
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.

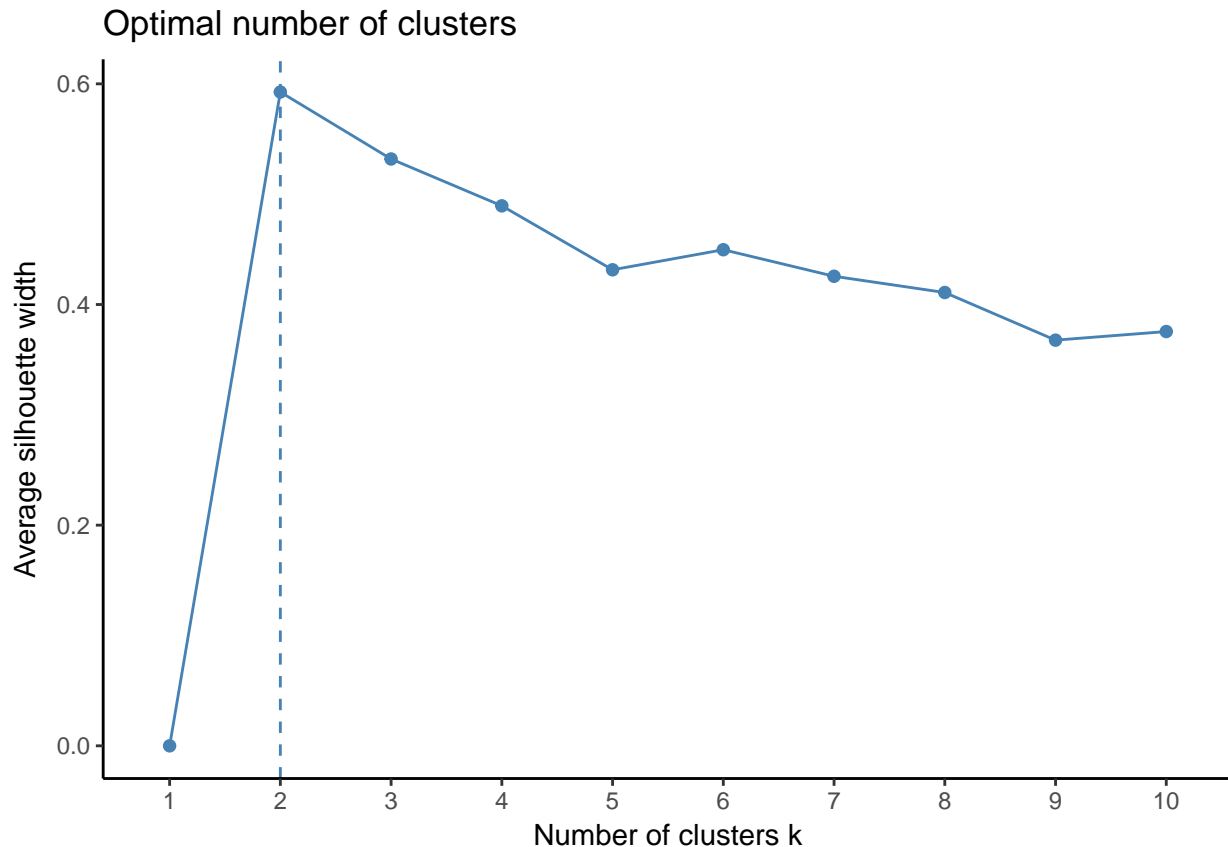
The K-mediod approach using the PAM algorithm is more robust and therefore less sensitive to outliers. A mediod is an object within a cluster for which the average dissimilarity between it and all the other objects in a cluster is minimal. (The most centrally located point). This point is representative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

1.) Estimate K- Clusters Using Silhouette Method in “factoextra”

The idea of using the silhouette 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 dissimilarity 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 dissimilarities. This is ignored when `x` is a dissimilarity 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"
```

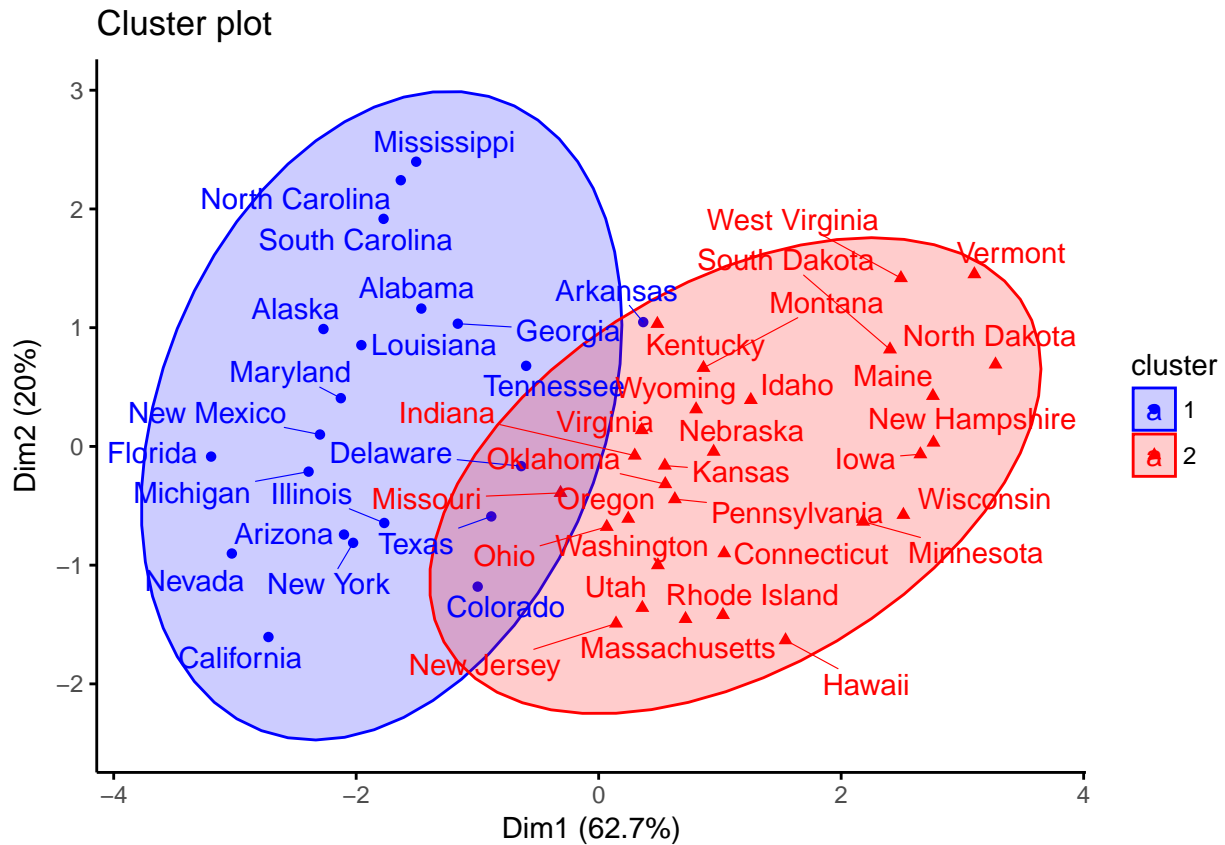
The `pam()` function returns the mediod objects! To access these:

```
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