**Clustering of US Arrests**

**Introduction**

This project was completed to build a portfolio of data analysis projects demonstrating the use of R Studio.

In this project, the goal is to show the performance of the K means clustering algorithm. This algorithm is the most common type of unsupervised learning.

**Business Applications of K-Means Clustering**

* Behavioral segmentation such as by purchase history, search history, or other activity
* Categorize inventory such as by sales activity or manufacturing metrics
* Sorting sensor measurements such as in motion sensors, images, audio, or health monitoring device activity
* Detecting bots or anomalies to separate valid activity from bots or clean up outliers

**Data Set**

The data set used in this project is the US Arrests data set that is built into R Studio.

**Preparing the Data**

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.

R Code:

### packages for analysis

library(tidyverse) # manipulate data

library(cluster) # clustering algorithms

library(factoextra) # clustering algorithms and visuals

### data prep

df <- USArrests # set the data set

df <- na.omit(df) # remove missing values

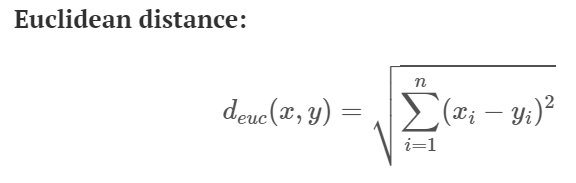
df <- scale(df) # standard scale

head(df) # return the first part of the data set matrix

**Exploratory Data**

R Studio code:

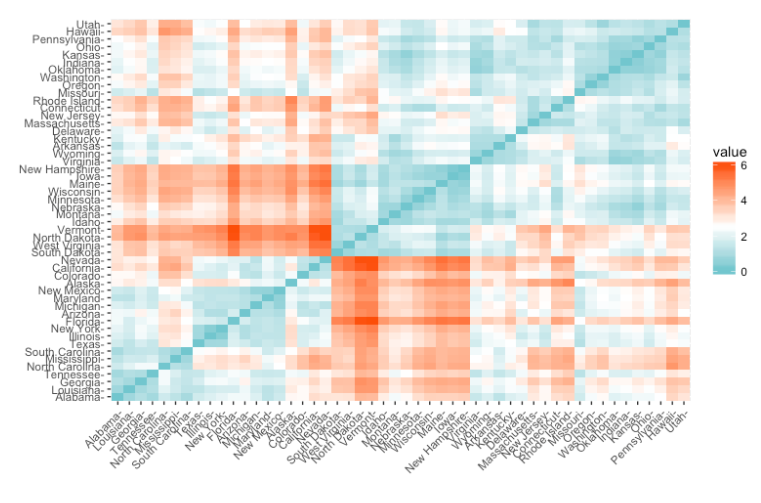
distance <- get\_dist(df) # compute distance matrix Euclidean as default distance measure



# visualize the distance matrix

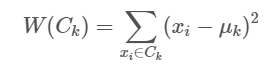
fviz\_dist(distance, gradient = list(low = "#00AFBB", mid = "white", high = "#FC4E07"))

Distance matrix produced by R Studio:



**The Idea Behind the Algorithm (Hartigan-Wong version of K-means clustering)**

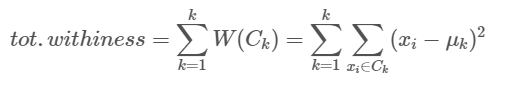
* Minimize total intra-cluster variation.
* Within-cluster variation as sum of squared Euclidean distances calculated as:



Xi is a data point belonging to the cluster Ck

µk is the mean value of the points assigned to the cluster Ck

* Each observation, denoted as x­I , is assigned to a cluster so that the sum of squares distance from x­I to its assigned cluster centers µk is minimized
* The goal is for the total within-cluster sum of squares value to be as small as possible
* Total within sum of squares defined as:



**Summary of K-Means Algorithm**

1. Specify K, the number of clusters, to be produced by the algorithm
2. Randomly choose k observations from the set of data as the first cluster centers or means (can be p-dimensional)
3. Assigns each observation to its closest cluster center, based on the Euclidean distance between the object and the centroid
4. For each of the k clusters update the cluster center by calculating the new mean values of all the data points in the cluster. The center of a Kth cluster is a vector of length *p* containing the means of all variables for the observations in the kth cluster; *p* is the number of variables.
5. Repeat steps 3 and 4 until the cluster assignments don’t change or the max number of iterations is reached. In R, 10 is the default value for the max iteration number.

**Performing the Algorithm with R**

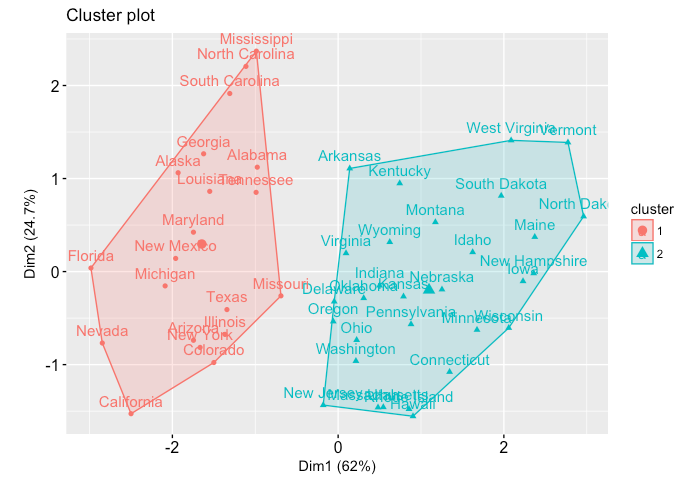
R Studio code:

k2 <- kmeans(df, centers = 2, nstart = 25) # k means with 2 clusters and 25 configurations

str(k2)

fviz\_cluster(k2, data = df) # provides a plot of the clusters

R Studio Output (for k = 2)



**Optimal Cluster Number by Elbow Method**

R Studio code:

set.seed(123)

# function to compute total within-cluster sum of square

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

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

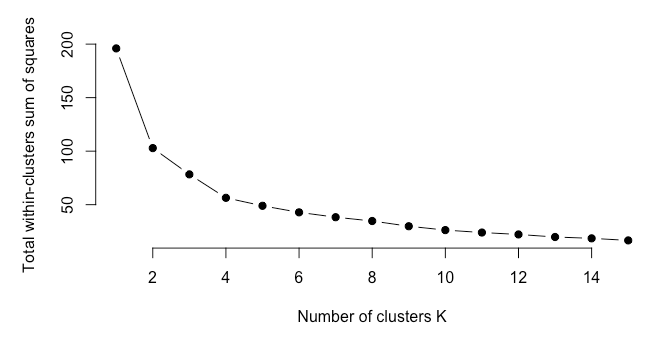
plot(k.values, wss\_values,

type="b", pch = 19, frame = FALSE,

xlab="Number of clusters K",

ylab="Total within-clusters sum of squares")

R Studio Output



2 clusters are too few. The ideal number of clusters is 4 according to the elbow method.

R Studio code:

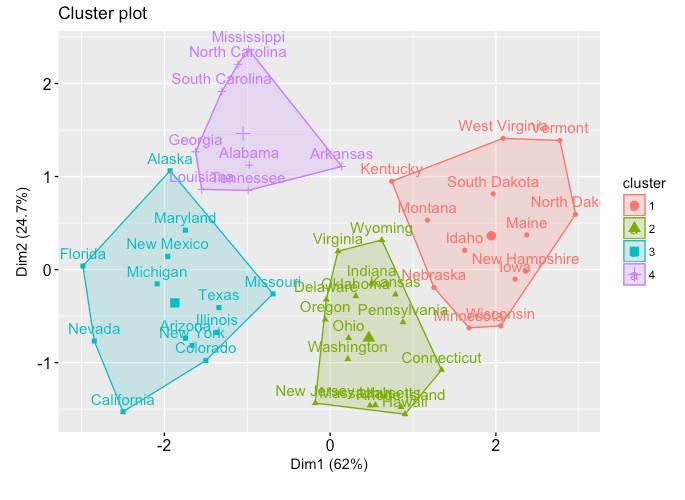
# Compute k-means clustering with k = 4

set.seed(123)

final <- kmeans(df, 4, nstart = 25)

print(final)

R Studio output:



**Final Comments**

* The algorithm is very simple and fast.
* The algorithm can handle large data sets in a time effective manner.
* One weakness of the algorithm is that the number of clusters must first be selected at the start.
* The algorithm is sensitive to outliers.