Essay 5: Principal Component Analysis and Clustering

Dhruvi, Vi, Huy, Harshal

4/24/2021

(1) INTRODUCTION

Principal Component Analysis and K-means Clustering

Principal Component Analysis

Principal component analysis (PCA) is a statistical procedure that helps us summarize the information content in large dataset by means of each smaller group. Its purpose is to capture the covariance information and supply it to the algorithm to build the model.

K-means Clustering

K-means clustering is the most commonly used unsupervised machine learning algorithm for partitioning a given data set into a set of k groups (i.e. k clusters), where k represents the number of groups pre-specified by the analyst. It classifies objects in multiple groups (i.e., clusters), such that objects within the same cluster are as similar as possible (i.e., high intra-class similarity), whereas objects from different clusters are as dissimilar as possible (i.e., low inter-class similarity). In k-means clustering, each cluster is represented by its center which corresponds to the mean of points assigned to the cluster. The purpose of this technique is to split a dataset into a set of k groups and minimize the sum of distances between the points within a cluster with their respective cluster centroid.

There are several k-means algorithms available. The standard algorithm is the Hartigan-Wong algorithm (1979), which defines the total within-cluster variation as the sum of squared distances Euclidean distances between items and the corresponding centroid:

$$W(C_k) = \sum_{x \in C_k} (x_i - \mu_k)^2$$

where:

- x_i is a data point belonging to the cluster C_k
- μ_k is the mean value of the points assigned to the cluster C_k

Each observation x_i is assigned to a given cluster such that the sum of squares (SS) distance of the observation to their assigned cluster centers μ_k is minimized.

The total within-cluster sum of square measures the compactness of the clustering and the smaller it is, the better. The total within-cluster variation formula is:

$$Total.withiness = \sum_{k=1}^{k} W(C_k) = \sum_{k=1}^{k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

K-means Algorithm

To start an algorithm, we first choose the number of clusters k, then select k random points from the data as centroids. Once we have defined the centroids, we will assign each point to the closest cluster centroid based on its distance from each cluster mean. The distance is determined by using Euclidean distance formula:

$$d_{euc}(xy) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

.

Then we update the cluster centroid by calculating the new mean values of all the data points in the cluster. Now, we have the new means of each centroid and we repeat the previous steps until the clusters formed in the current step are equal to those formed in the previous step.

Loading R packages

```r
library("dplyr")
library("tidyverse") #for data manipulation and visualization
library("corrplot")
library("animation")
library("cluster")
library("GGally")
library("factoextra")
library("ggcorrplot")
```

(2) DATA DESCRIPTION

Examples of data and problem

Wine Data set

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The dataset contains 13 attributes with 178 rows.

1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Non-flavanoid phenols
9) Proanthocyanins
10) Color intensity
11) Hue
12) OD280/OD315 of diluted wines
13) Proline

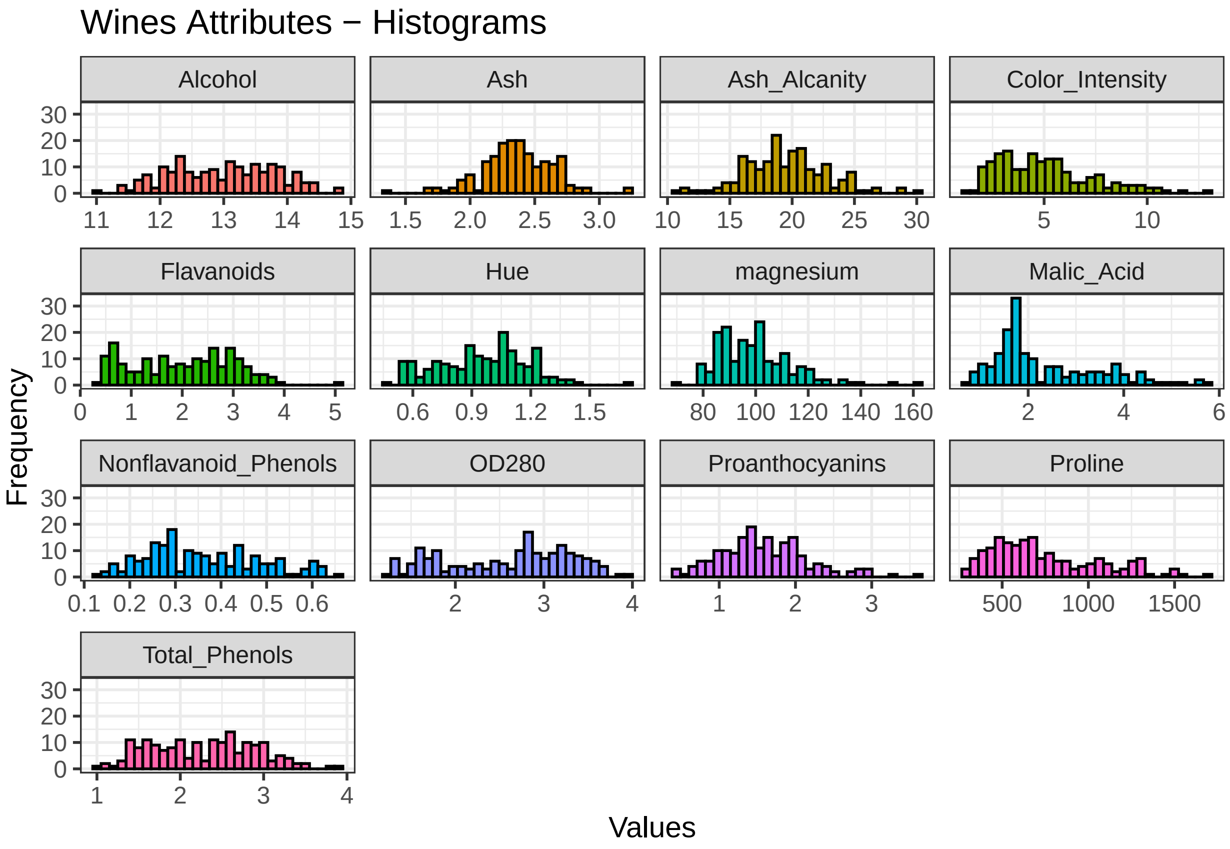
Our goal is to try to group wines with similar characteristics together and determine the number of possible clusters. This would help us make predictions and reduce dimensionality.

Importing the data file

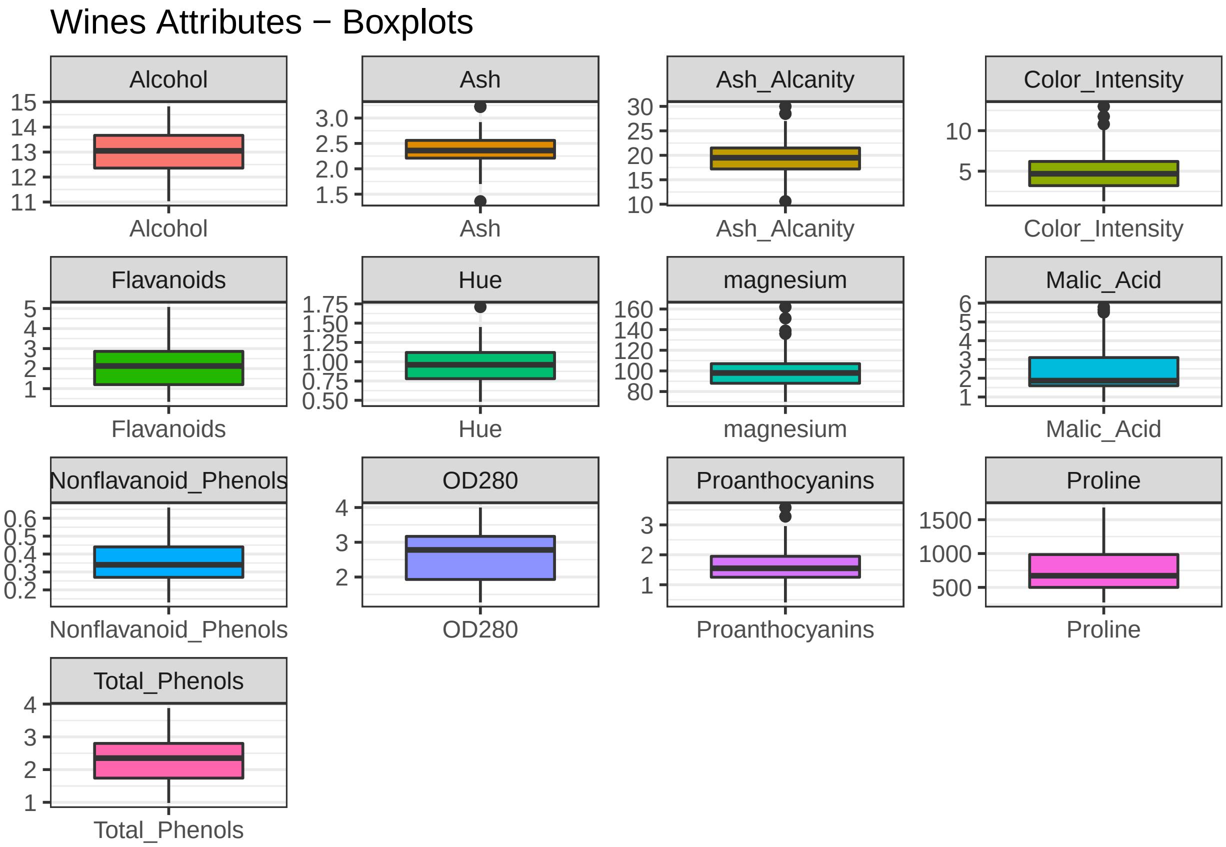
```
wine.cols = c("Alcohol", "Malic_Acid", "Ash", "Ash_Alcanity", "magnesium", "Total_Phenols", "Flavanoids
              "Nonflavanoid_Phenols", "Proanthocyanins", "Color_Intensity", "Hue", "OD280", "Proline")

wine <- read.table("wine.data", sep = ",", row.names=NULL, col.names=wine.cols, nrows=178)[ ,2:14]

wine <- wine %>%
  drop_na() %>% #remove NA values
  distinct() # remove duplicated rows

summary(wine)
```

Visualization

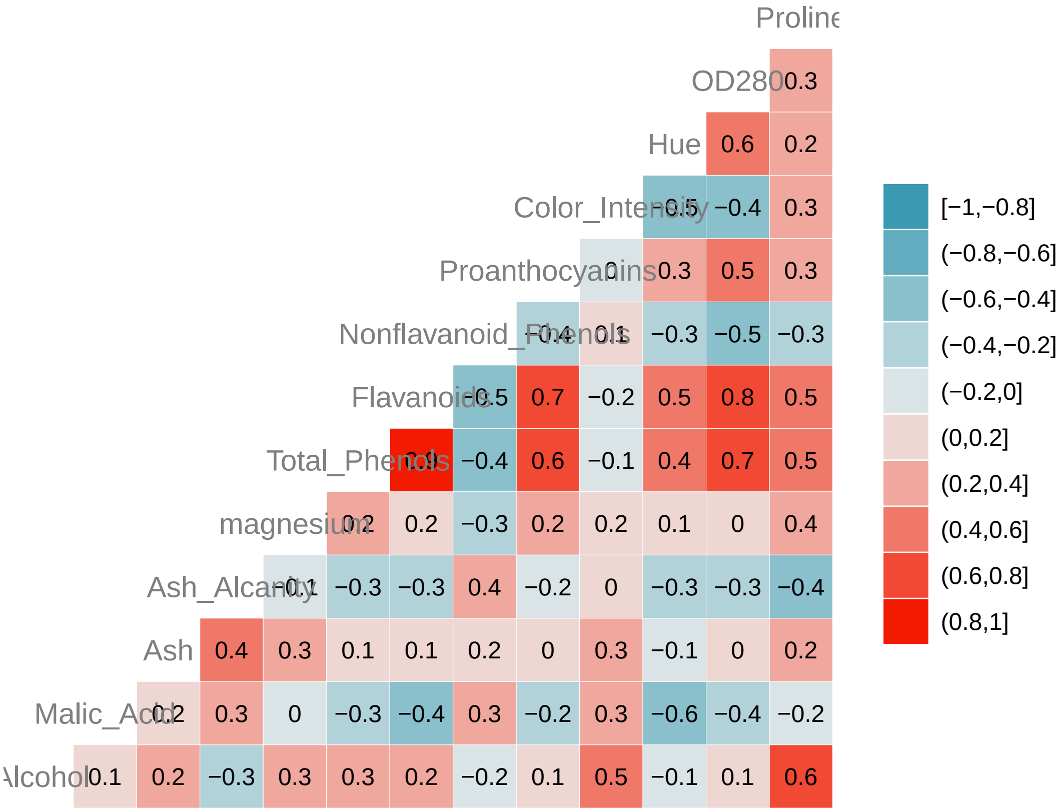
First we plot the histogram of each attribute. The idea here is to visualize the data distribution for each feature. This helps in determining in comparing how the atrributes are disttributed under each cluster as compared to the distribution prior. We notice, for example, that that Alcohol contents are more evenly distributed however, for Ash it follows a perfect bell curve. Malc acid, OD280 and proline are more skewed and don't follow an even distribution.



The above noted observations can be confirmed through the boxplot below which helps visualize the skewness of data in the corresponding attributes clearly by looking where the horizontal line, representing the mean, lies within the boxplot.



Below we plot a correlation plot to see how each attribute is correlated. We find a strong linear correlation between Total_Phenols and Flavanoids & OD280 and Flavanoids. This could turn out to be useful when we try to make predictions about what each charcateristic value should be for a given wine across different clusters if we know the value of one of the attributes.



Data Normalization

We need to normalize the data to express it in the same range of values.

```
winesNorm <- as.data.frame(scale(wine))

summary(winesNorm)
```

```
##     Alcohol           Malic_Acid            Ash            Ash_Alcanity
##  Min.   :-2.42786   Min.   :-1.4293   Min.   :-3.65769   Min.   :-2.672889
##  1st Qu.:-0.78346   1st Qu.:-0.6610   1st Qu.:-0.56768   1st Qu.:-0.694514
##  Median : 0.06964   Median :-0.4198   Median :-0.02239   Median :-0.005081
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.000000
##  3rd Qu.: 0.83620   3rd Qu.: 0.6791   3rd Qu.: 0.70467   3rd Qu.: 0.594427
##  Max.   : 2.27041   Max.   : 3.0913   Max.   : 3.14032   Max.   : 3.142334
##    magnesium       Total_Phenols       Flavanoids      Nonflavanoid_Phenols
##  Min.   :-2.0875   Min.   :-2.09471   Min.   :-1.6857   Min.   :-1.8637
##  1st Qu.:-0.8175   1st Qu.:-0.88155   1st Qu.:-0.8246   1st Qu.:-0.7406
##  Median :-0.1120   Median : 0.09217   Median : 0.1067   Median :-0.1790
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5230   3rd Qu.: 0.81048   3rd Qu.: 0.8377   3rd Qu.: 0.6232
##  Max.   : 4.4033   Max.   : 2.53444   Max.   : 3.0607   Max.   : 2.3881
##  Proanthocyanins   Color_Intensity        Hue               OD280
##  Min.   :-2.05924   Min.   :-1.6240   Min.   :-2.08167   Min.   :-1.8923
##  1st Qu.:-0.58954   1st Qu.:-0.7937   1st Qu.:-0.77240   1st Qu.:-0.9563
##  Median :-0.06465   Median :-0.1612   Median : 0.01317   Median : 0.2492
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.63521   3rd Qu.: 0.4927   3rd Qu.: 0.71144   3rd Qu.: 0.8023
##  Max.   : 3.48713   Max.   : 3.4181   Max.   : 3.28635   Max.   : 1.9794
##     Proline
##  Min.   :-1.4834
##  1st Qu.:-0.7784
##  Median :-0.2321
##  Mean   : 0.0000
##  3rd Qu.: 0.7619
##  Max.   : 2.9690
```

(3) ANALYSIS

Computation

K-means Algorithm

K-means algorithm can be summarized as follows:

1. Specify the number of clusters (K) to be created (by the analyst)
2. Select randomly k objects from the data set as the initial cluster centers or means
3. Cluster assignment step: Assigns each observation to their closest centroid, based on the Euclidean distance between the object and the centroid
4. Cluster centroid update: For each of the k clusters update the cluster centroid by calculating the new mean values of all the data points in the cluster. The centroid 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. Iteratively minimize the total within sum of square. That is, iterate steps 3 and 4 until the cluster assignments stop changing or the maximum number of iterations is reached. By default, the R software uses 10 as the default value for the maximum number of iterations.

Computing k-means clustering in R

We can compute k-means in R with the *kmeans* function. Here will group the data into two clusters (*centers = 2*). The *kmeans* function also has an *nstart* option that attempts multiple initial configurations and reports on the best one. For example, adding *nstart = 25* will generate 25 initial configurations. This approach is often recommended.

```
k2 <- kmeans(winesNorm, centers = 2, nstart = 25)
str(k2)
```

```
## List of 9
##  $ cluster     : int [1:177] 2 2 2 2 2 2 2 2 2 2 ...
##  $ centers     : num [1:2, 1:13] -0.0645 0.0374 0.6582 -0.382 0.1901 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:13] "Alcohol" "Malic_Acid" "Ash" "Ash_Alcanity" ...
##  $ totss       : num 2288
##  $ withinss    : num [1:2] 512 1130
##  $ tot.withinss: num 1642
##  $ betweenss   : num 646
##  $ size        : int [1:2] 65 112
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

The output of *kmeans* is a list with several bits of information. The most important being:

- *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. In an optimal segmentation, one expects this ratio to be as lower as possible for each cluster, since we would like to have homogeneity within the clusters.
- *tot.withinss*: Total within-cluster sum of squares, i.e. sum(withinss). It measures the compactness (i.e goodness) of the clustering and we want it to be as small as possible.
- *betweenss*: The between-cluster sum of squares, i.e. $totss - tot.withinss$. The between-cluster sum of squares. In an optimal segmentation, one expects this ratio to be as higher as possible, since we would like to have heterogeneous clusters.
- *size*: The number of points in each cluster.

If we print the results we'll see that our groupings resulted in 2 cluster sizes of 54 and 123. We see the cluster centers (means) for the two groups across the thirteen variables . We also get the cluster assignment for each observation.

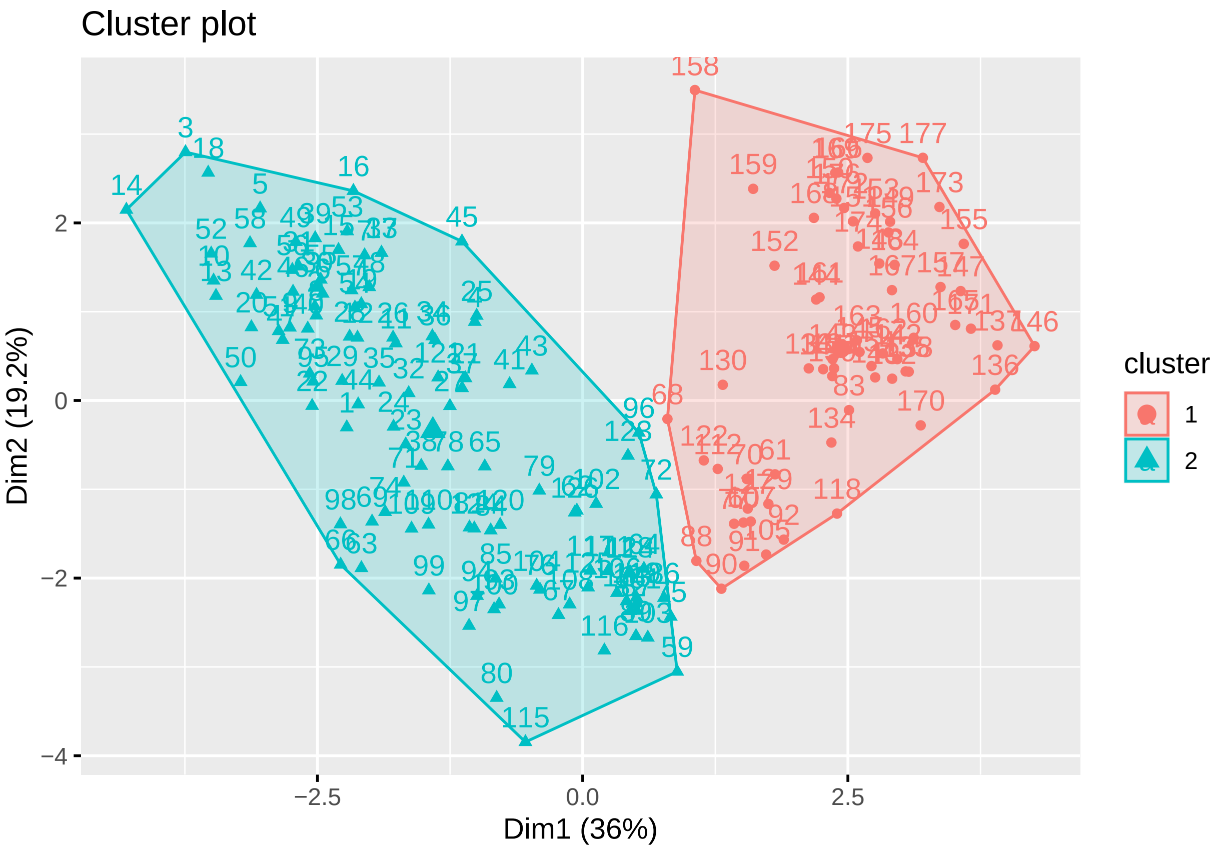
```
k2
```

```
## K-means clustering with 2 clusters of sizes 65, 112
##
## Cluster means:
##       Alcohol Malic_Acid        Ash Ash_Alcanity   magnesium Total_Phenols
## 1 -0.06445760  0.6581961  0.1901384    0.5091125 -0.14456797    -0.9355765
## 2  0.03740843 -0.3819888 -0.1103482   -0.2954671  0.08390105     0.5429685
##   Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity        Hue
## 1 -1.0380706            0.8305442      -0.7082478        0.541918 -0.8753923
## 2  0.6024517           -0.4820123       0.4110367       -0.314506  0.5080402
##        OD280    Proline
## 1 -1.0632184 -0.4462323
## 2  0.6170464  0.2589741
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 1 2 1 2 2 2 2
##  [75] 2 2 1 2 2 2 2 2 1 2 2 2 2 1 2 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2
## [112] 1 2 2 2 2 2 1 2 2 2 1 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1]  511.6897 1130.4400
##  (between_SS / total_SS =  28.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Interpretation of Model:

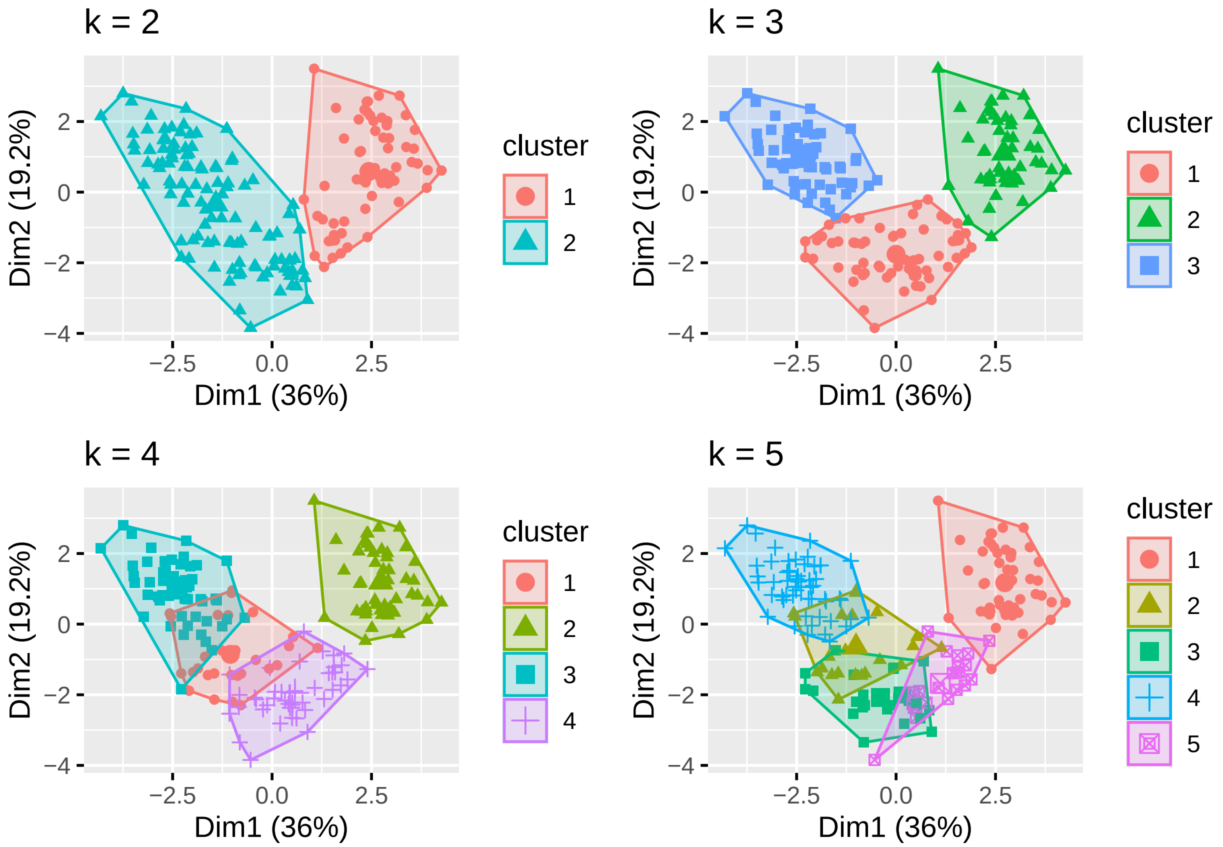
We can view our results by using *fviz_cluster*. This provides a nice illustration of the clusters. If there are more than two dimensions (variables) *fviz_cluster* will perform principal component analysis (PCA) and plot the data points according to the first two principal components that explain the majority of the variance.

```
fviz_cluster(k2,data=winesNorm)
```



Because the number of clusters (k) must be set before we start the algorithm, it is often advantageous to use several different values of k and examine the differences in the results. We can execute the same process for 3, 4, and 5 clusters, and the results are shown in the figure:

```
k3 <- kmeans(winesNorm, centers = 3, nstart = 25)
k4 <- kmeans(winesNorm, centers = 4, nstart = 25)
k5 <- kmeans(winesNorm, centers = 5, nstart = 25)

# plots to compare
p1 <- fviz_cluster(k2, geom = "point",  data = winesNorm) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = winesNorm) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = winesNorm) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = winesNorm) + ggtitle("k = 5")

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



This visual assessment tells us where true delineations occur between clusters; however, it does not tell us what the optimal number of clusters is. To determine that we will look at a few different ways in the next section. However, what you can see from the cluster visualization above that k=2 and k=3 are the only values with the minimal cross-section of the clusters. Each data point in those clusters strictly falls under one cateegory or the other and isn't ambiguous.

(4) MODEL EVALUATION

Model Summary and Assessment

In this essay so far, we have been able to successfully create a k-means clustering model that will cluster chemically similar wines together that are grown in the same region of Italy but by three different cultivars. To start with, we visualized our wines data set and normalized it so that we don't have to depend on an arbitrary variable unit. After that, we successfully developed the k-means clustering model on the normalized data set using different values of 'k'.

There are different methods of evaluating how our model performs like Elbow method, Silhouette method and Gap Statistics. However, for this essay we will just be using one of these methods to analyze which value of 'k' is the most optimal value for our data set.

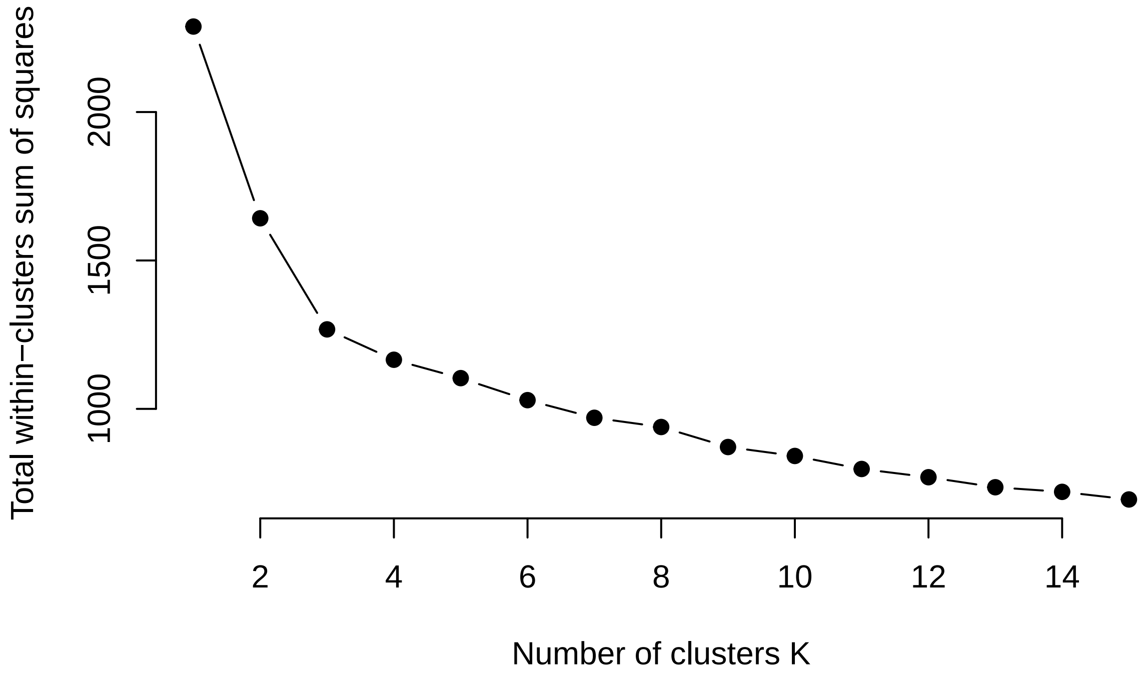
In the next section, we will discuss about the optimal value of 'k' that will help in minimizing the total within-cluster variation using a method called the 'Elbow method'.

Prediction and Model accuracy

To calculate the optimal value of 'k', we will perform the following 'Elbow method' :

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 15 clusters.
2. For each k, calculate the total within-cluster sum of square (wss)
3. Plot the curve of wss according to the number of clusters k.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

```r
set.seed(123)

# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(winesNorm, 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")
```



```
#  we use the wrapper function **fviz_nbclust()** that computes the 'Elbow method'
# using just one line of code. We can see the knee (bend) in the plot being at k=3.

# set.seed(123)
#
# fviz_nbclust(winesNorm, kmeans, method = "wss")
```

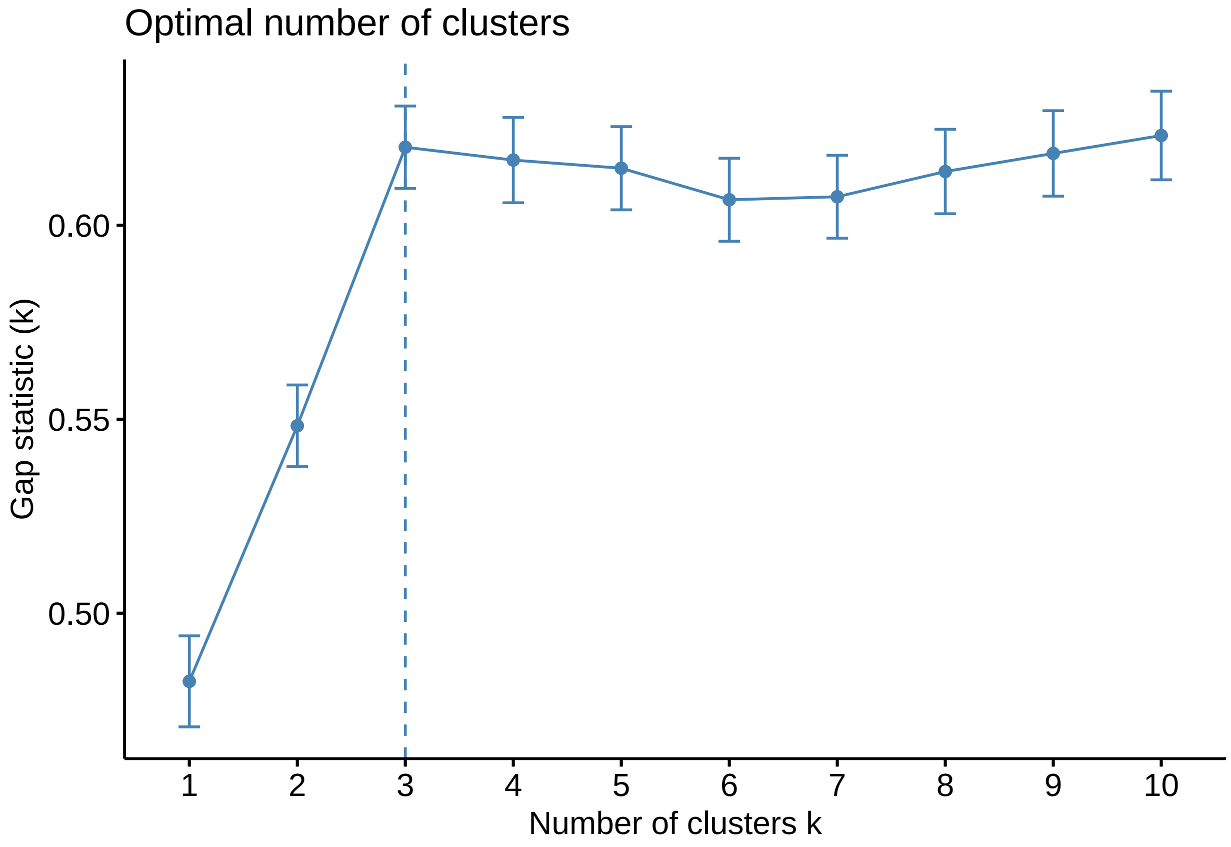
Looking at the plot above, we can see that **k=3 is the optimal value of k** since from this point onwards the total WCSS (Within Cluster Sum of Squares) doesn't decrease significantly. Hence, for our essay we will be using k=3 (3-means clustering method) to cluster similar chemically similar wines together based on the initial data set.

Gap Statistic

We can also use the Gap statistic measure to determine the optimal k value. he gap statistic compares the total intracluster variation for different values of k with their expected values under null reference distribution of the data (i.e. a distribution with no obvious clustering).

To compute the gap statistic method we can use the `clusGap` function which provides the gap statistic and standard error for an output. We can visualize the results with fviz_gap_stat which also suggests 3 as the optimal number of clusters.

```
set.seed(123)
gap_stat <- clusGap(winesNorm, FUN = kmeans, nstart = 25,
                    K.max = 10, B = 50)

fviz_gap_stat(gap_stat)
```

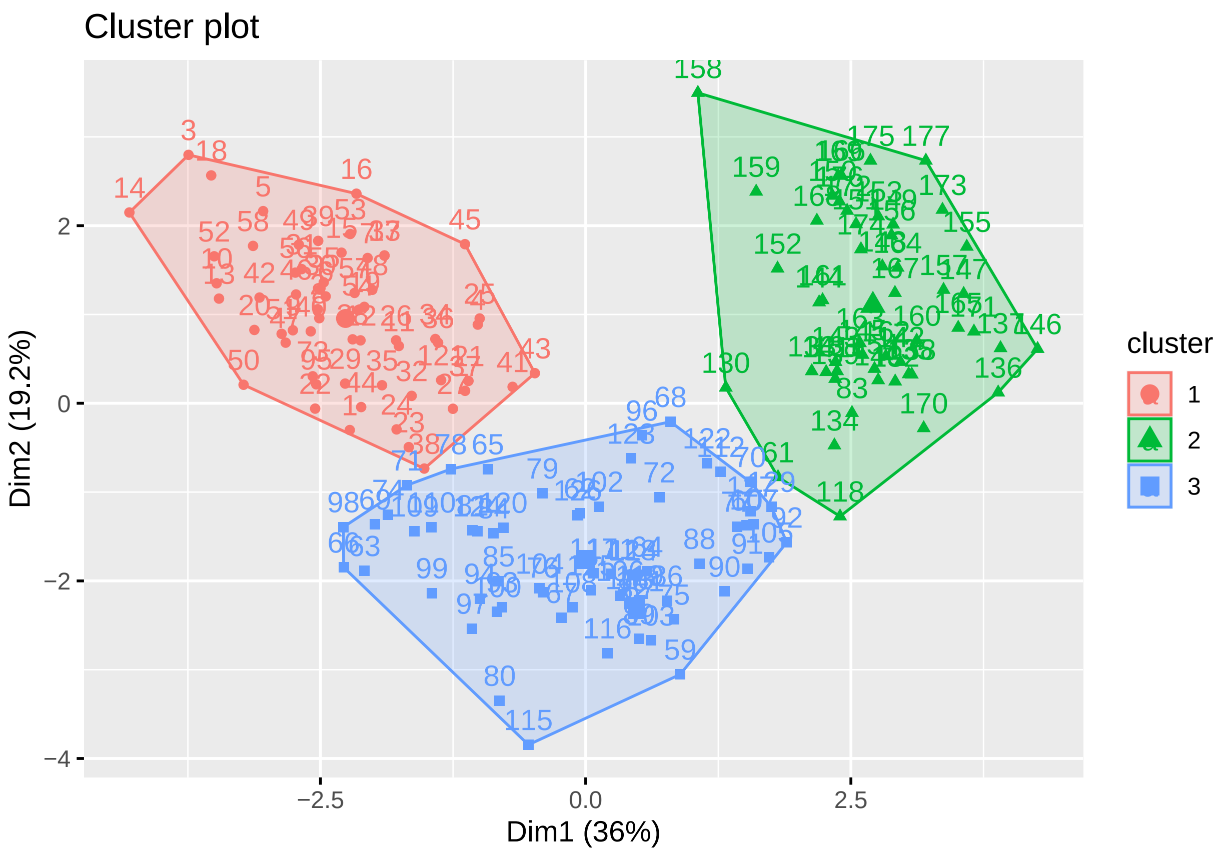


Finally, just to visualize the case where k=3, we get the following clustering using our data set.

```
k3 <- kmeans(winesNorm, centers = 3, nstart = 25)
k3
```

```
## K-means clustering with 3 clusters of sizes 61, 51, 65
##
## Cluster means:
##      Alcohol Malic_Acid        Ash Ash_Alcanity   magnesium Total_Phenols
## 1  0.8333649 -0.3013131  0.3661731   -0.6065538  0.56922228    0.88768039
## 2  0.1736447  0.8642504  0.1871775    0.5168437 -0.06497127   -0.97106500
## 3 -0.9183253 -0.3953334 -0.4905017    0.1637039 -0.48321576   -0.07114136
##    Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity        Hue
## 1  0.98016451          -0.56173008      0.57583669       0.1702296  0.4753467
## 2 -1.20624204           0.71915195     -0.77171004       0.9378162 -1.1566204
## 3  0.02658937          -0.03709561      0.06509498      -0.8955790  0.4614076
##        OD280    Proline
## 1  0.7753334  1.1296451
## 2 -1.2872265 -0.4002655
## 3  0.2823571 -0.7460740
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 2 3 3 3 3 3 3 3 3 3 3 3 1 3
##  [75] 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 2 3 3 1 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 382.1859 326.3534 559.2978
##  (between_SS / total_SS =  44.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
fviz_cluster(k3, data = winesNorm)
```



Finally, we can extract the clusters and add to our initial data to do some descriptive statistics at the cluster level:

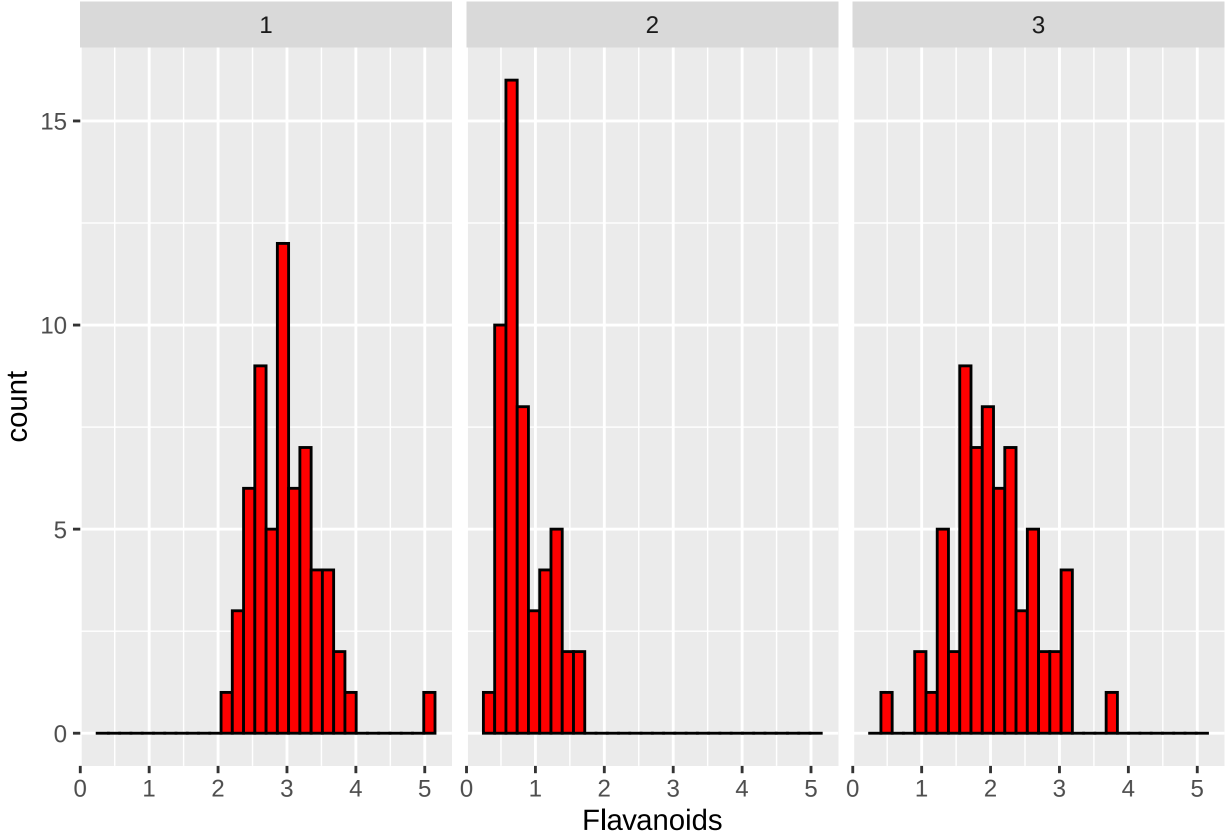
```
wine %>%
  mutate(Cluster = as.factor(k3$cluster)) %>%
  group_by(Cluster) %>%
  summarise_all("mean")
```

```
## # A tibble: 3 x 14
##   Cluster Alcohol Malic_Acid   Ash Ash_Alcanity magnesium Total_Phenols
##   <fct>     <dbl>      <dbl> <dbl>        <dbl>     <dbl>         <dbl>
## 1 1          13.7       2.00  2.47         17.5     108.           2.85
## 2 2          13.1       3.31  2.42         21.2      98.7          1.68
## 3 3          12.3       1.90  2.23         20.1      92.7          2.25
## # ... with 7 more variables: Flavanoids <dbl>, Nonflavanoid_Phenols <dbl>,
## #   Proanthocyanins <dbl>, Color_Intensity <dbl>, Hue <dbl>, OD280 <dbl>,
## #   Proline <dbl>
```

Below we can visualize the difference between the attributes under each cluster, we use Flavanoids and OD280 as examples for visualization:

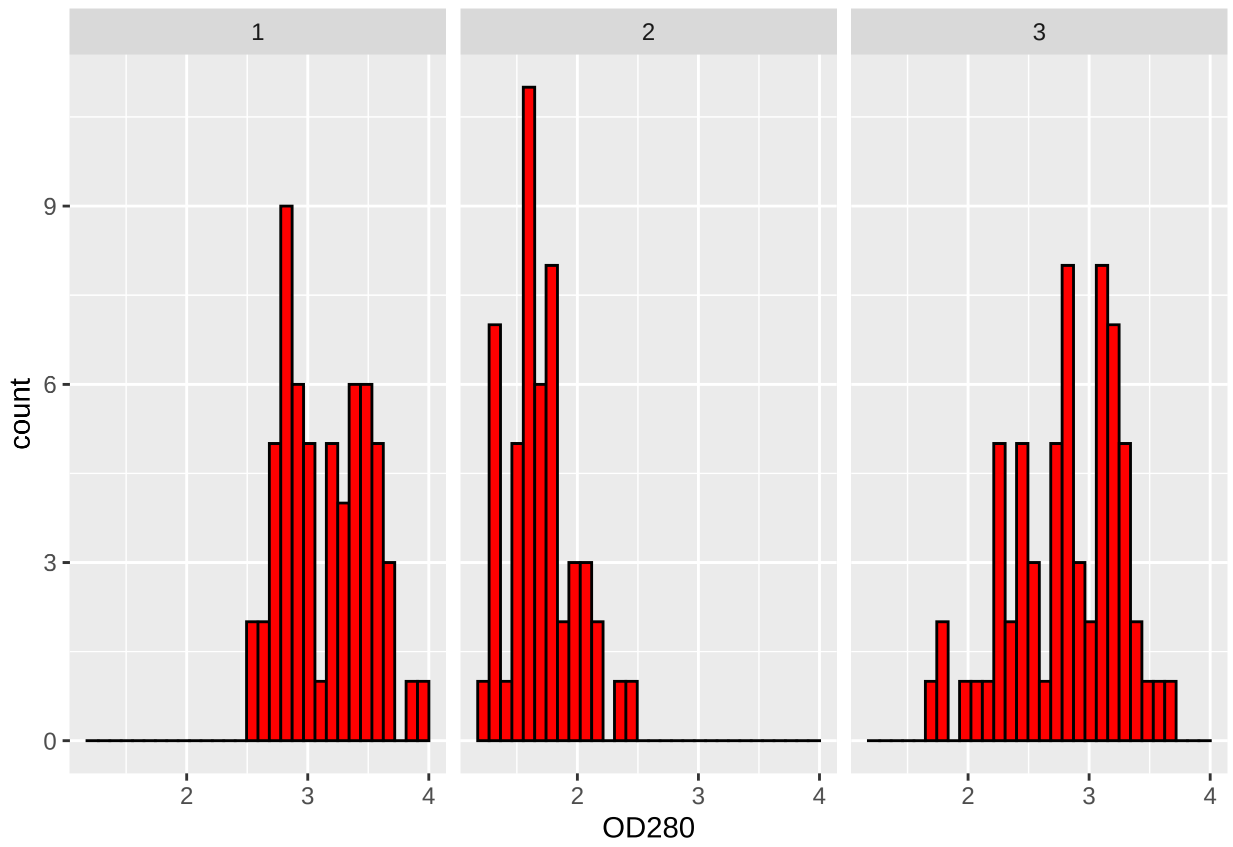
```
wine_cluster <- wine %>%
  mutate(Cluster = k3$cluster)

wine_cluster %>%
  ggplot(aes(x=Flavanoids, color = Cluster)) +
  geom_histogram(color="black", fill="red") +
  facet_wrap(~Cluster)
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



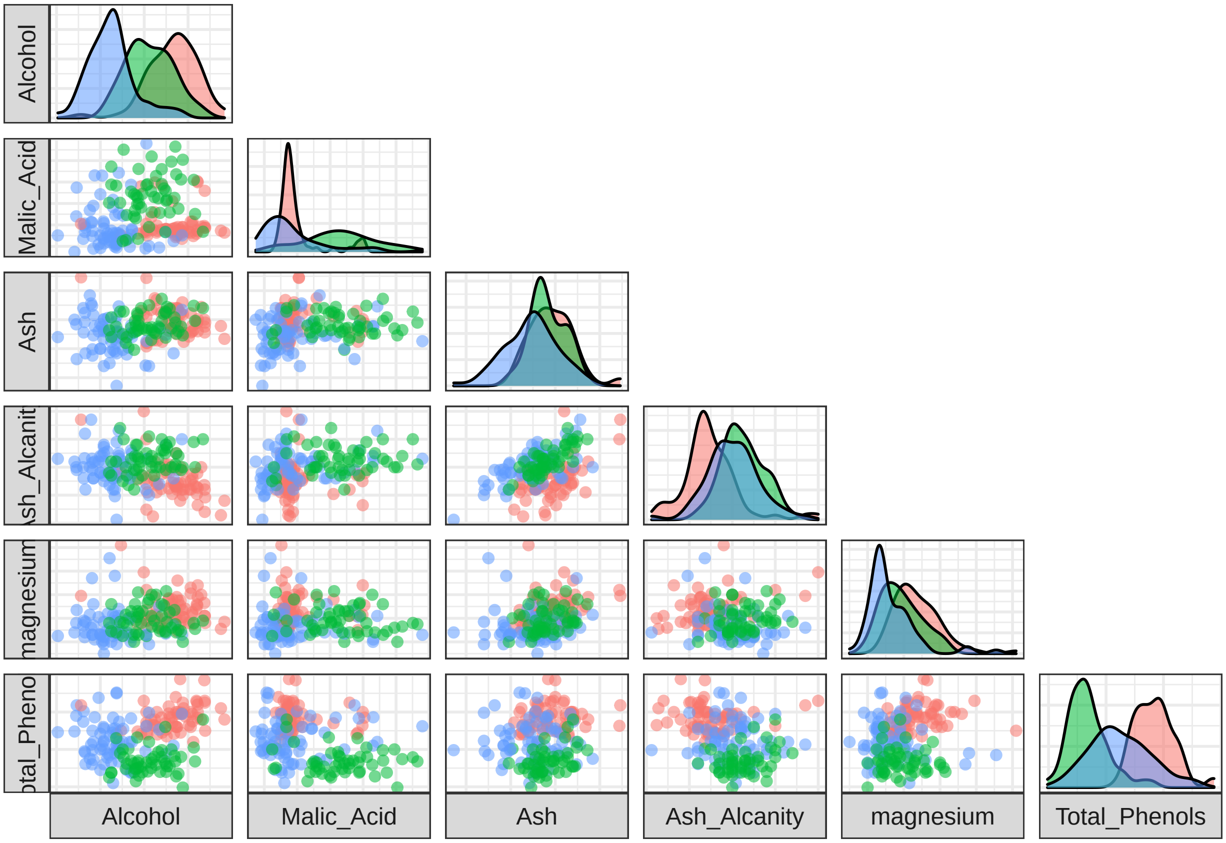
```
wine_cluster %>%
  ggplot(aes(x=OD280, color = Cluster)) +
  geom_histogram(color="black", fill="red") +
  facet_wrap(~Cluster)
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



Below we can visualize the densities of each cluster within each attribute and how they differ.

```
ggpairs(cbind(wine, Cluster=as.factor(k3$cluster)),
        columns=1:6, aes(colour=Cluster, alpha=0.5),
        lower=list(continuous="points"),
        upper=list(continuous="blank"),
        axisLabels="none", switch="both") +
        theme_bw()
```



(5) CONCLUSION

Summary

In this article we perform unsupervised learning through k-means clustering on the wine data set consisting of 13 attributes. We first do madel analysis and try to determine how many clusters we should pick by try out different values of k. We determined that k=3 works the best. So we define 3 centroids which are determined recursively until no more changes are observed and we group each data point to a given cluster by calculating the distance between the data point and the centroids of the cluster, assigning it to the one with the smallest distance. Finally, we do a model evaluation to confirm which value of k is the most optimal through the 'Elbow Method' and 'Gap statistic' plots which confirm that k=3 is indeed the most optimal. We also visualize the clusters and the difference of the attributes under each cluster.

K-means clustering is a simple and fast algorithm. It is also quite efficient for larger datasets; however, it has a few shortcomings:

- It assumes prior knowledge of the data and requires the analyst to choose the appropriate number of cluster (k) in advance
- The final results obtained is sensitive to the initial random selection of cluster centers. Why is it a problem? Because, for every different run of the algorithm on the same dataset, you may choose different set of initial centers. This may lead to different clustering results on different runs of the algorithm.
- It's sensitive to outliers.
- If you rearrange your data, it's very possible that you'll get a different solution every time you change the ordering of your data.

(6) REFERENCES

https://stat.ethz.ch/R-manual/R-patched/library/datasets/html/USArrests.html

https://www.kaggle.com/xvivancos/tutorial-clustering-wines-with-k-means

https://www.kaggle.com/rodrigofragoso/explained-k-means-pca-visualization

https://uc-r.github.io/kmeans_clustering

https://www.guru99.com/r-k-means-clustering.html