Clients Clustering Using K-means

Daniel Njoroge

7/7/2021

Executive Summary

In this Data Science project, we are going to apply machine leaning techniques to implement clients clustering. To find your best client, client clustering is the ideal methodology. We are going to use dataset obtained from kaggle.com. We will explore the data upon which we will build cluster models. This project implements several K-means algorithm versions. Clients clustering is one of the application of unsupervised learning - a type of machine learning in which models are trained using unlabeled datasets and are allowed to run without any supervision.

Introduction

Clients clustering involves categorizing of customers into individual groups with a similarity relevant to marketing such as as age, source of income, interests, and gender. This technique assumes that different clients have different preferences and requires specific unique marketing strategy to address them appropriately. Through the data collected, companies can gain a deeper understanding of customer preferences and the requirements for discovering valuable clusters that would reap them maximum profit. The key differentiators that divide clients into unique groups to be targeted are: Data related to economic status, demographics, geography, and behavioral patterns.

Data Exploration and Analysis

```
#importing essential packages
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
if(!require(plotrix)) install.packages("plotrix", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: plotrix
```

```
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(purrr)) install.packages("purrr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: purrr

##
## Attaching package: 'purrr'

## The following object is masked from 'package:caret':
##
##     lift
```

```
if(!require(cluster)) install.packages("cluster", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: cluster
```

```
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: gridExtra
```

```
if(!require(grid)) install.packages("grid", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: grid
```

```
setwd("D:/GitHub/clients_clustering/clients_dataset")

customer_data = read.csv("Mall_Customers.csv") #reading data from file
```

```
str(customer_data) #structure of the data frame
```

```
## 'data.frame':    200 obs. of  5 variables:
##  $ CustomerID            : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Gender                : chr  "Male" "Male" "Female" "Female" ...
##  $ Age                   : int  19 21 20 23 31 22 35 23 64 30 ...
##  $ Annual.Income..k..    : int  15 15 16 16 17 17 18 18 19 19 ...
##  $ Spending.Score..1.100.: int  39 81 6 77 40 76 6 94 3 72 ...
```

```
names(customer_data) #row titles only
```

```
## [1] "CustomerID"             "Gender"                 "Age"
## [4] "Annual.Income..k.."     "Spending.Score..1.100."
```

We will now display the first six rows of our dataset using the head() function and use the summary() function to display its summary.

```
head(customer_data)
```

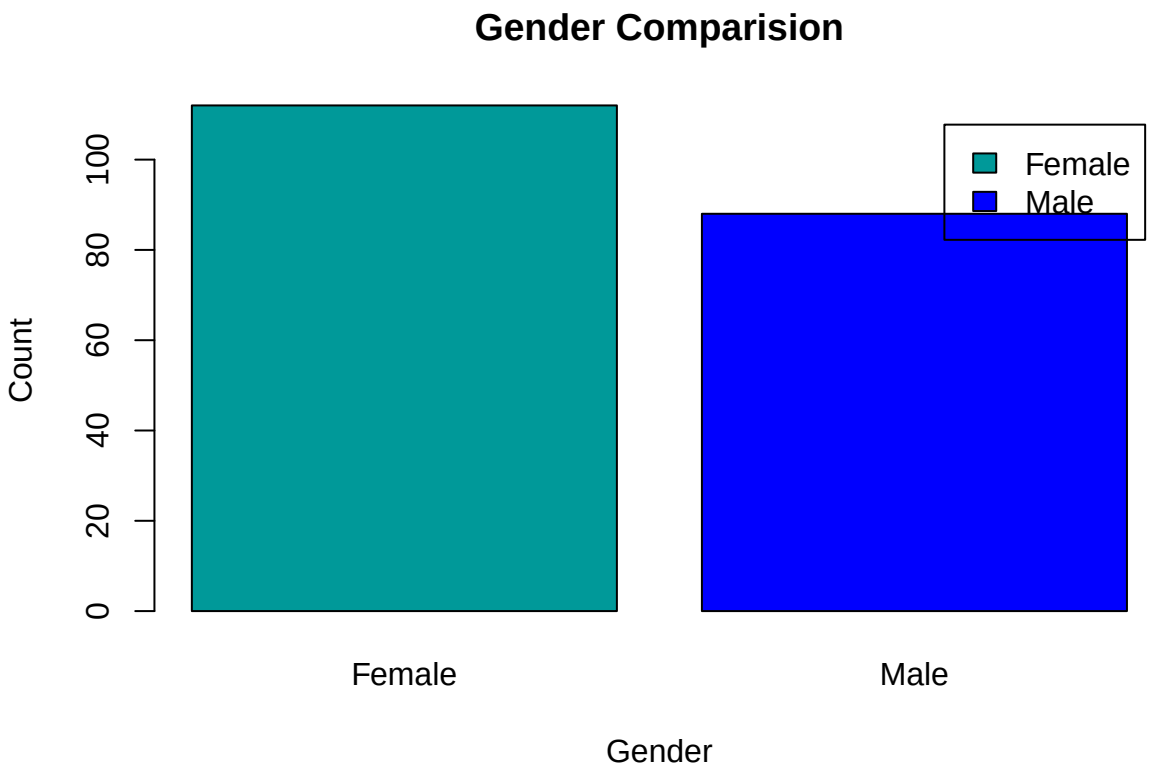
```
##   CustomerID Gender Age Annual.Income..k.. Spending.Score..1.100.
## 1          1   Male  19                 15                     39
## 2          2   Male  21                 15                     81
## 3          3 Female  20                 16                      6
## 4          4 Female  23                 16                     77
## 5          5 Female  31                 17                     40
## 6          6 Female  22                 17                     76
```

```
summary(customer_data)
```

```
##    CustomerID        Gender               Age        Annual.Income..k..
##  Min.   :  1.00   Length:200         Min.   :18.00   Min.   : 15.00
##  1st Qu.: 50.75   Class :character   1st Qu.:28.75   1st Qu.: 41.50
##  Median :100.50   Mode  :character   Median :36.00   Median : 61.50
##  Mean   :100.50                      Mean   :38.85   Mean   : 60.56
##  3rd Qu.:150.25                      3rd Qu.:49.00   3rd Qu.: 78.00
##  Max.   :200.00                      Max.   :70.00   Max.   :137.00
##  Spending.Score..1.100.
##  Min.   : 1.00
##  1st Qu.:34.75
##  Median :50.00
##  Mean   :50.20
##  3rd Qu.:73.00
##  Max.   :99.00
```

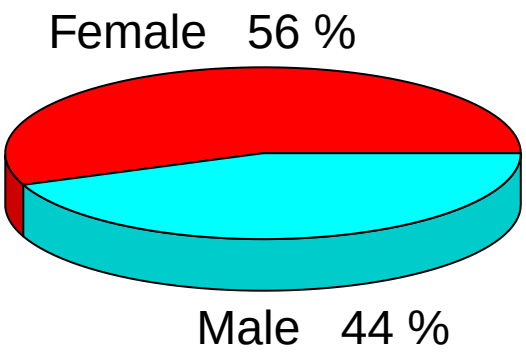
Gender Distribution We will visualize the gender distribution across our data using barplot and a pie chart.

```
#barplot of gender distribution
a = table(customer_data$Gender) #fetch from Gender column only
barplot(a, main="Gender Comparision",
        ylab = "Count",
        xlab = "Gender",
        col = c("#009999", "#0000FF"),
        legend = rownames(a))
```



We observe that the number of females is higher than that of the males. Now, let us visualize a pie chart to observe the ratio of male and female distribution.

```
pct = round(a/sum(a)*100)
lbs = paste(c("Female", "Male"), " ", pct, "%", sep = " ")
pie3D(a, labels = lbs,
      main = "Ratio of Female and Male")
```



The observation is that the percentage of females is 56%, whereas the percentage of male in the customer dataset is 44%.

Age Distribution We will visualize clients age distribution using histogram to show the frequency. Let's have a look at the age summary first.

```
summary(customer_data$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   28.75   36.00   38.85   49.00   70.00
```

```
hist(customer_data$Age,
     col = "grey",
     main = "Count of Age Class",
     xlab = "Age Class",
     ylab = "Frequency",
     labels = TRUE #adding frequency label to individual bars
     )
```

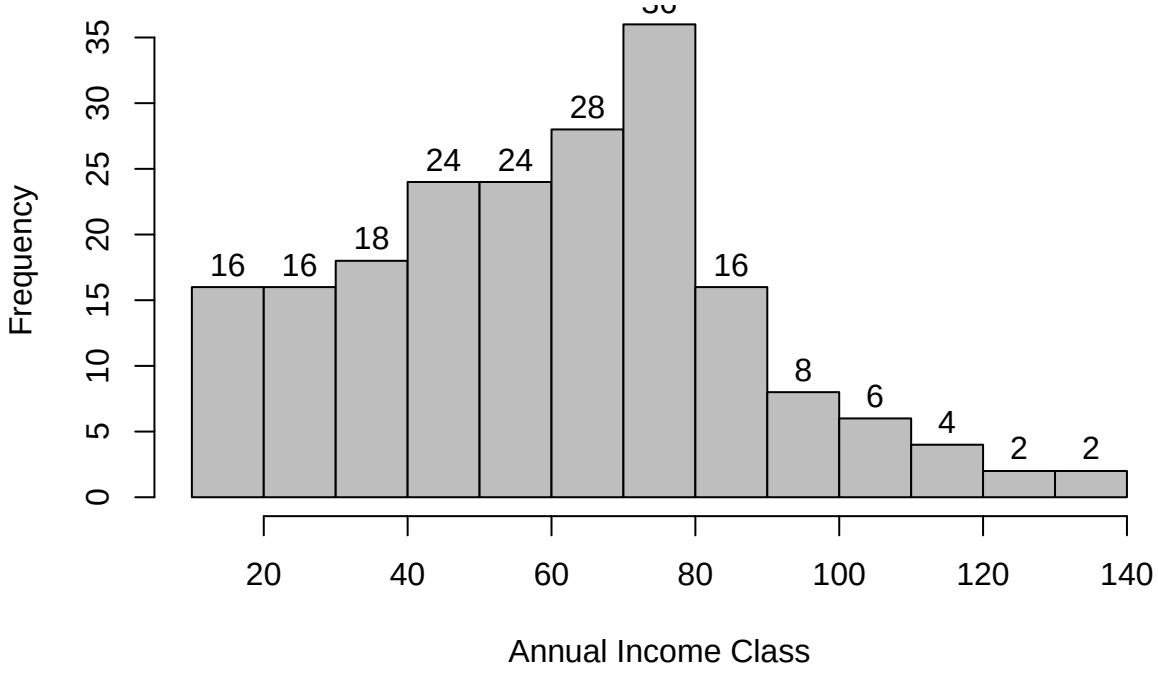
From the summary and histogram above, we can conclude that the majority of customer are aged between 30 and 35. The minimum age of customers is 18, whereas, the maximum age is 70.

Clients Annual Income Analysis We are going to use a histogram and a density plot to visualize and analyze clients' annual income. This is an important aspect to consider during clustering.

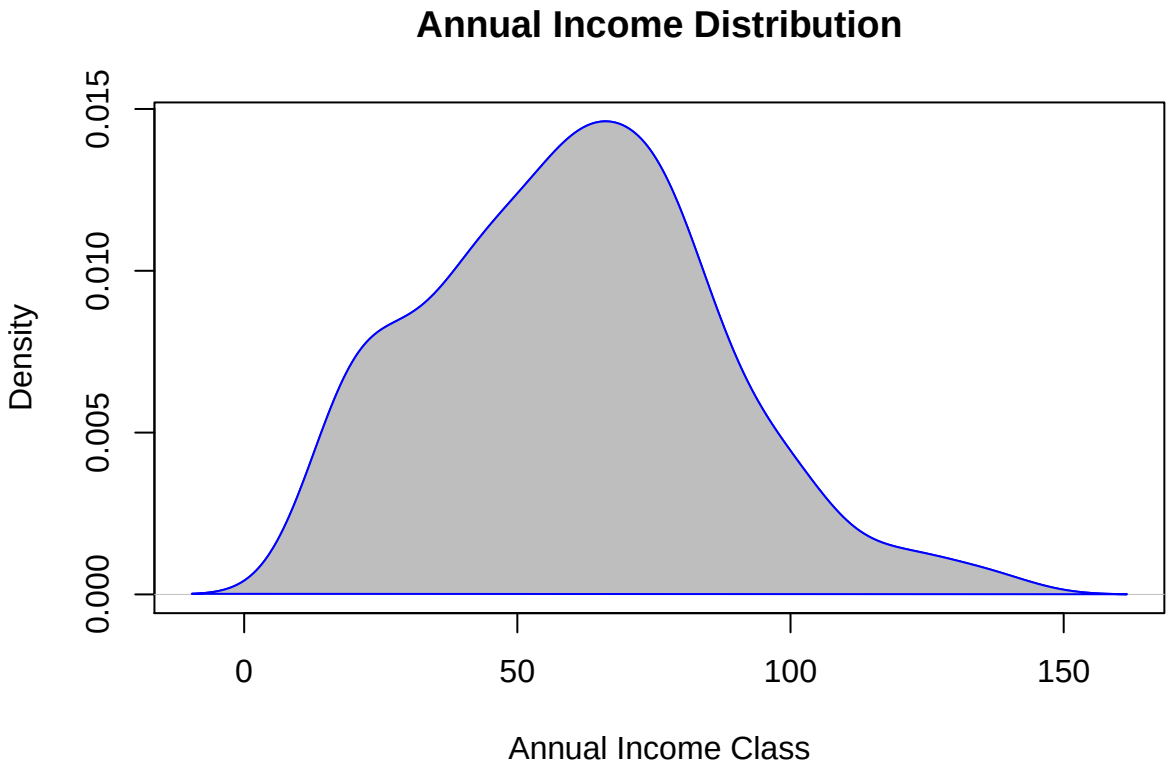
```
summary(customer_data$Annual.Income..k..)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.00   41.50   61.50   60.56   78.00  137.00
```

```
#annual income histogram
hist(customer_data$Annual.Income..k..,
     col = "grey",
     main = " Annual Income Distribution",
     xlab = "Annual Income Class",
     ylab = "Frequency",
     labels = TRUE
     )
```




```
#the density plot
#the density plot
plot(density(customer_data$Annual.Income..k..),
     col="blue",
     main="Annual Income Distribution",
     xlab="Annual Income Class",
     ylab="Density")
#filled density plot
polygon(density(customer_data$Annual.Income..k..),
        col="grey", border = "blue")
```



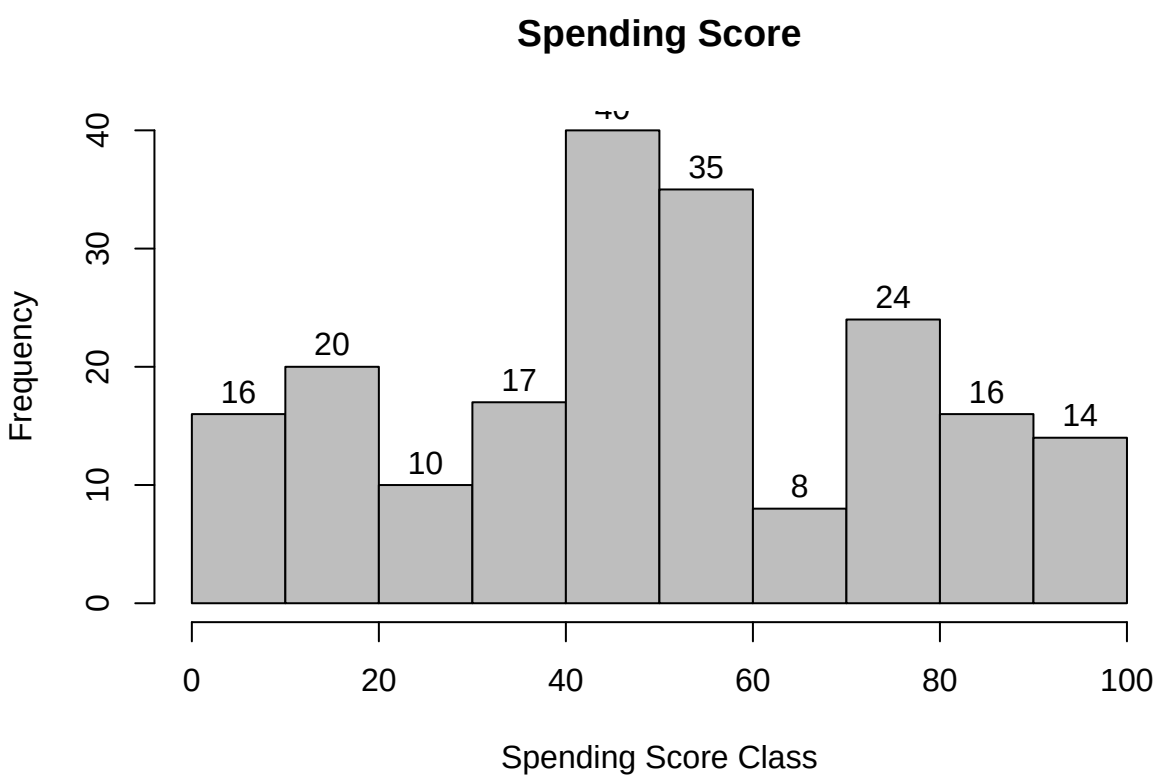
From the annual income summary, we see that minimum annual income of the clients from our data is 15 and the maximum income is 137. Also, people with an average income of 70 have th highest frequency count on our histogram. We also find that the average salary of all the customers is 60.56. From the density plot, we observe a normal distribution of the annual income.

Clients Spending Score

```
summary(customer_data$Spending.Score..1.100.)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   34.75   50.00   50.20   73.00   99.00
```

```
hist(customer_data$Spending.Score..1.100.,
     main="Spending Score",
     xlab="Spending Score Class",
     ylab="Frequency",
     col="grey",
     labels = TRUE)
```



From the above analysis we see that the minimum spending score is 1 and the maximum is 99. The average spending score is 50.20. From the histogram we conclude that, people between the class of 40 and 50 have the highest spending score in terms of frequency.

K-means Clustering

The Algorithm

While using the k-means clustering algorithm, the first step is to indicate the number of clusters (k) that we wish to produce in the final output. The algorithm starts by selecting k objects from dataset randomly that will serve as the initial centers for our clusters. These selected objects are the cluster means, also known as centroids. Then, the remaining objects have an assignment of the closest centroid. This centroid is defined by the Euclidean Distance present between the object and the cluster mean. We refer to this step as "cluster assignment". When the assignment is complete, the algorithm proceeds to calculate new mean value of each cluster present in the data. After the recalculation of the centers, the observations are checked if they are closer to a different cluster. Using the updated cluster mean, the objects undergo reassignment. This goes on repeatedly through several iterations until the cluster assignments stop altering. The clusters that are present in the current iteration are the same as the ones obtained in the previous iteration.

Summing up the K-means clustering:

- We specify the number of clusters that we need to create.
- The algorithm selects k objects at random from the dataset. This object is the initial cluster or mean.
- The closest centroid obtains the assignment of a new observation. We base this assignment on the Euclidean Distance between object and the centroid.

- k clusters in the data points update the centroid through calculation of the new mean values present in all the data points of the cluster.
- The kth cluster's centroid has a length of p that contains means of all variables for observations in the k-th cluster. We denote the number of variables with p.
- Iterative minimization of the total within the sum of squares. Then through the iterative minimization of the total sum of the square, the assignment stop wavering when we achieve maximum iteration. The default value is 10 that the R software uses for the maximum iterations.

Determining Optimal Clusters

1. The Elbow Method The main goal behind cluster partitioning methods like k-means is to define the clusters such that the intra-cluster variation stays minimum.

minimize(sum W(Ck)), k=1...k

Where Ck represents the kth cluster and W(Ck) denotes the intra-cluster variation. With the measurement of the total intra-cluster variation, one can evaluate the compactness of the clustering boundary. We can then proceed to define the optimal clusters as follows:

- We begin by calculating the clustering algorithm for several values of k by creating a variation within k from 1 to 10 clusters.
- We will then calculate the total intra-cluster sum of square (iss).
- Then, we proceed to plot iss based on the number of k clusters. This plot denotes the appropriate number of clusters required in our model. In the plot, the location of a bend or a knee is the indication of the optimum number of clusters.

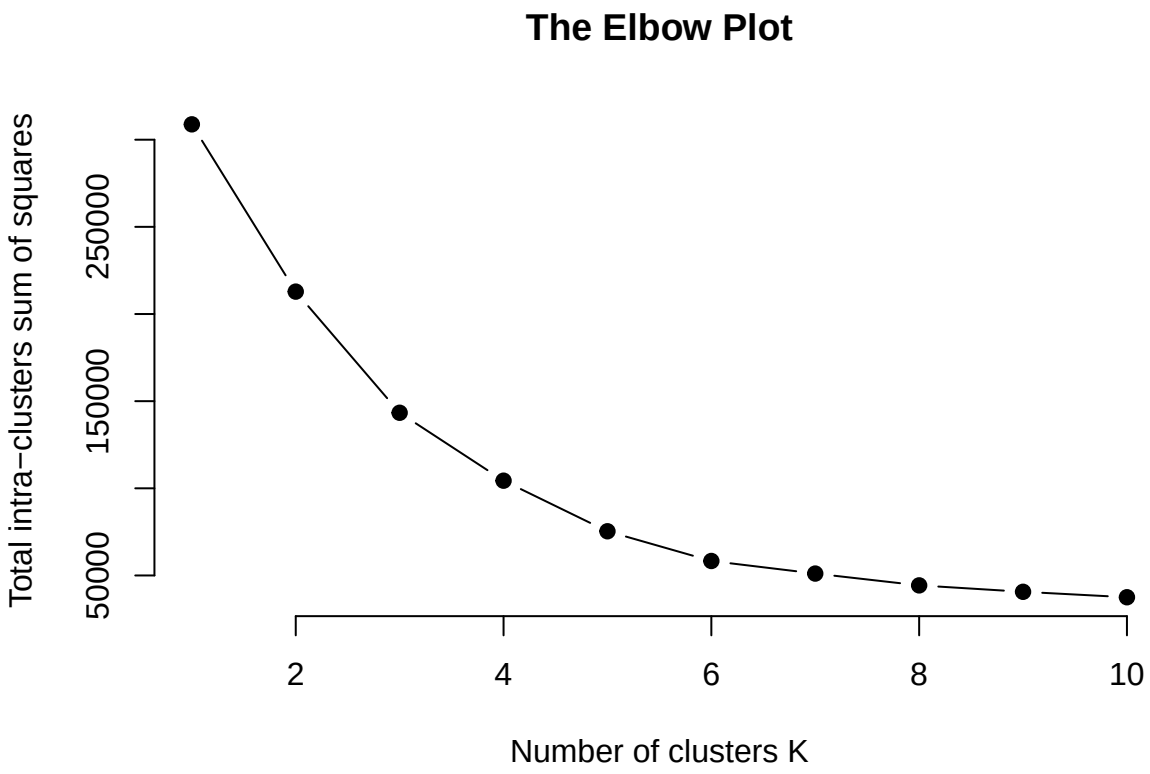
```
set.seed(123)
```

```
iss <- function(k) {
  kmeans(customer_data[,3:5], k, iter.max=100, nstart=100, algorithm = "Lloyd" )$tot.withinss
}
```

```
k.values <- 1:10 #Number of clusters K

iss_values <- map_dbl(k.values, iss) #Total intra-clusters sum of squares
```

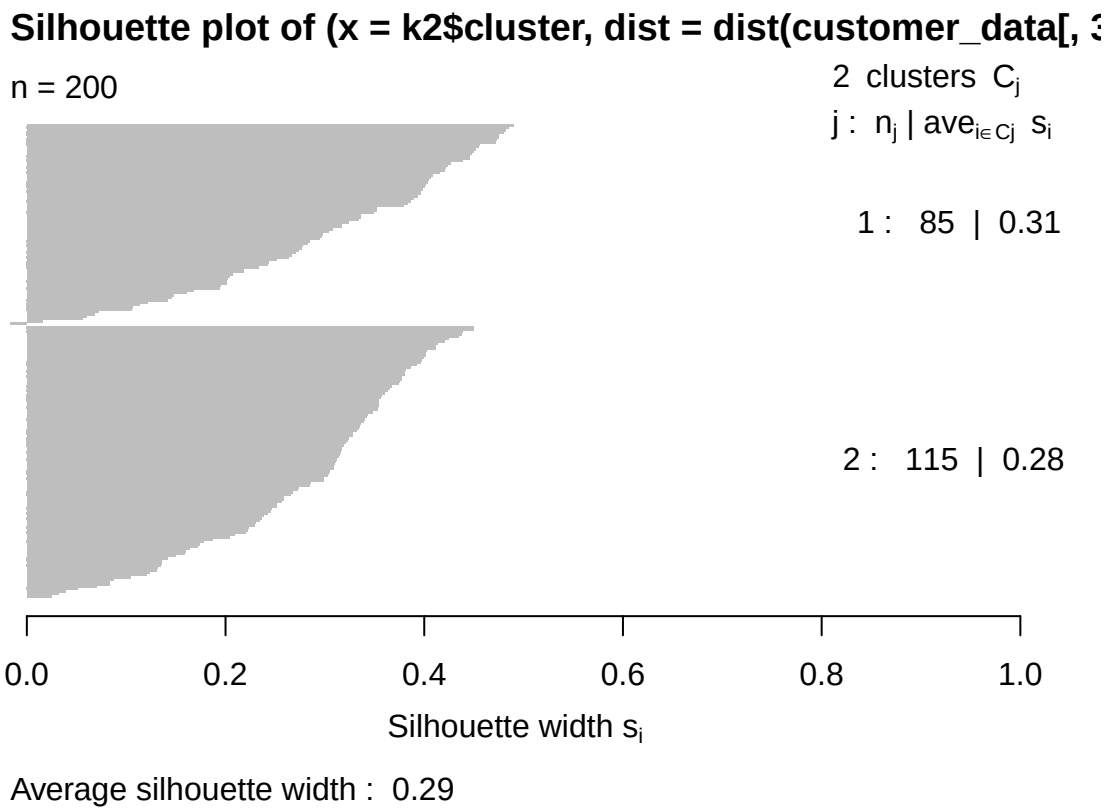
```
plot(k.values, iss_values,
     type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Total intra-clusters sum of squares",
     main = "The Elbow Plot")
```



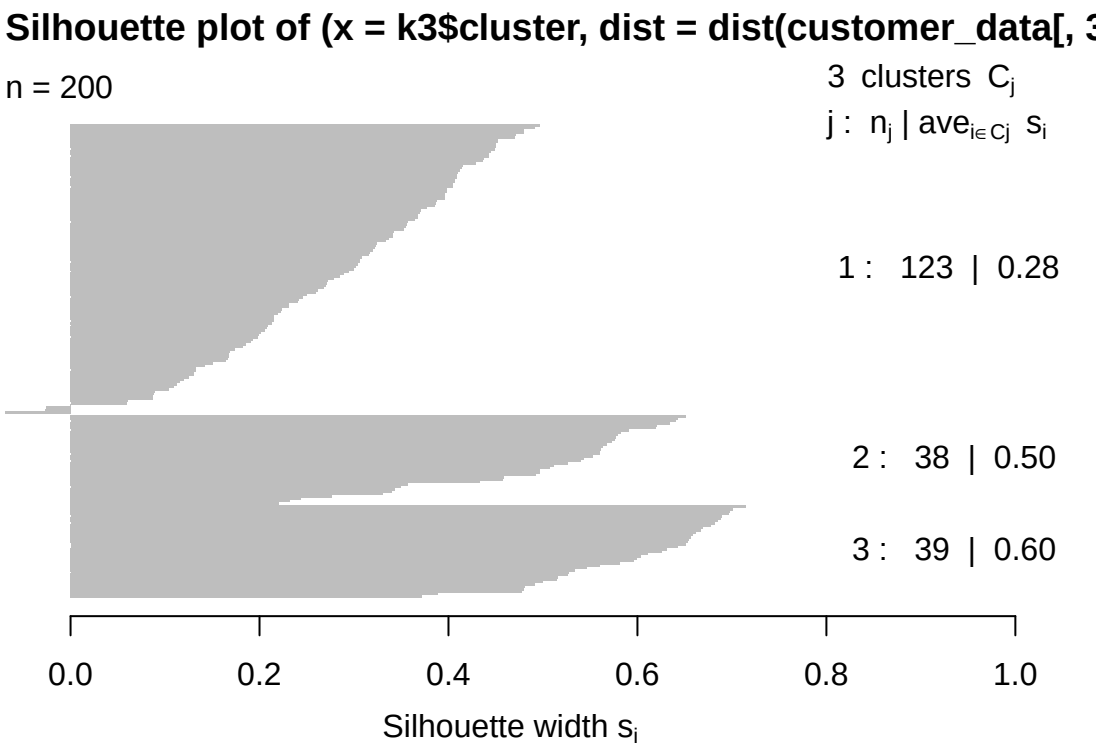
With 4 appearing at the bend in the elbow plot, we can conclude that it is the approporiate number of clusters.

2. The Silhouette Method With silhouette method, we can measure the quality of our clustering operation and determine how well within the cluster the data object is. If we obtain a high average silhouette width, it means that we have good clustering. The average silhouette method calculates the mean of silhouette observations for different k values. With the optimal number of k clusters, one can maximize the average silhouette over significant values for k clusters. Here, the optimal cluster will possess the highest average.

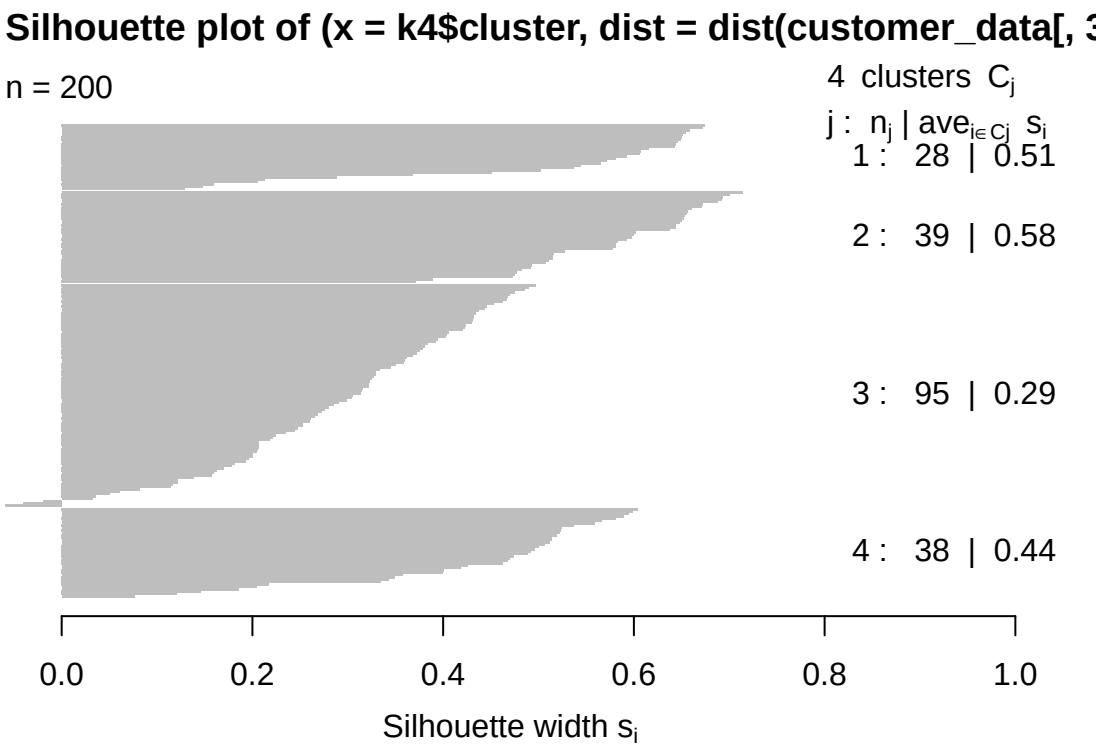
```
k2 <- kmeans(customer_data[, 3:5], 2, iter.max = 100, nstart = 50, algorithm = "Lloyd")
s2 <- plot(silhouette(k2$cluster, dist(customer_data[, 3:5], "euclidean")))
```




```
k3 <- kmeans(customer_data[, 3:5], 3, iter.max = 100, nstart = 50, algorithm = "Lloyd")
s3 <- plot(silhouette(k3$cluster, dist(customer_data[, 3:5], "euclidean")))
```

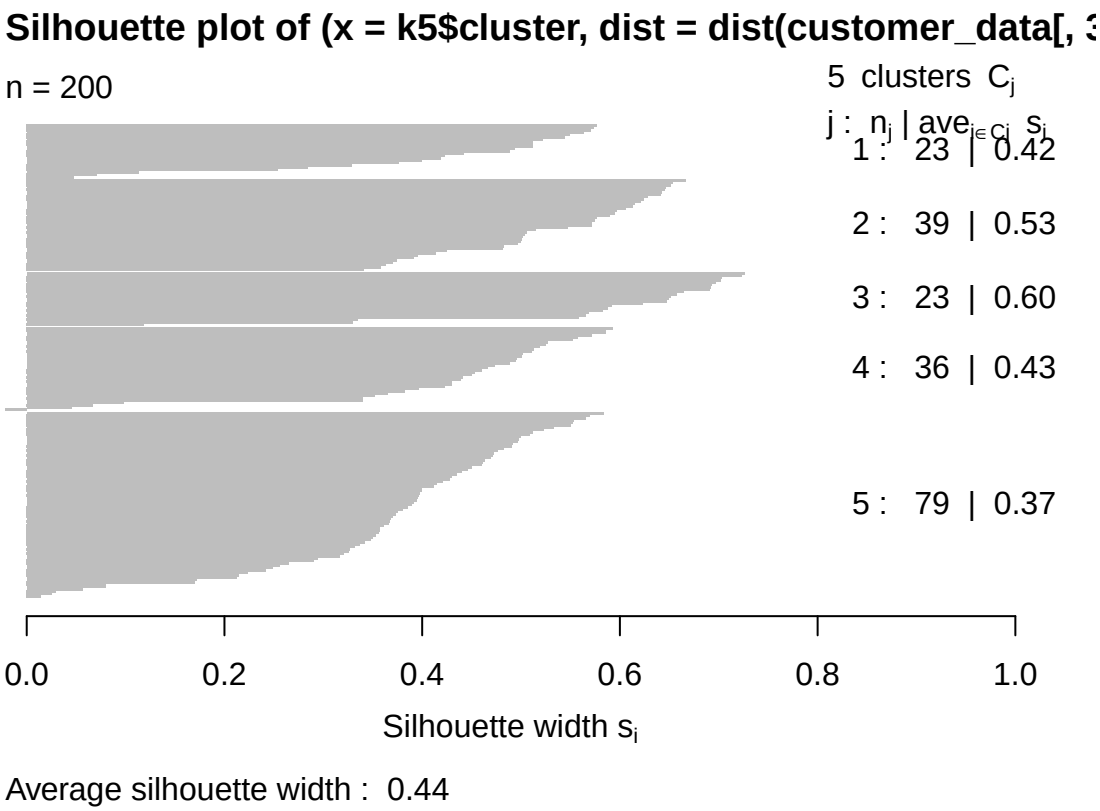



```
k4 <- kmeans(customer_data[, 3:5], 4, iter.max = 100, nstart = 50, algorithm = "Lloyd")
s4 <- plot(silhouette(k4$cluster, dist(customer_data[, 3:5], "euclidean")))
```

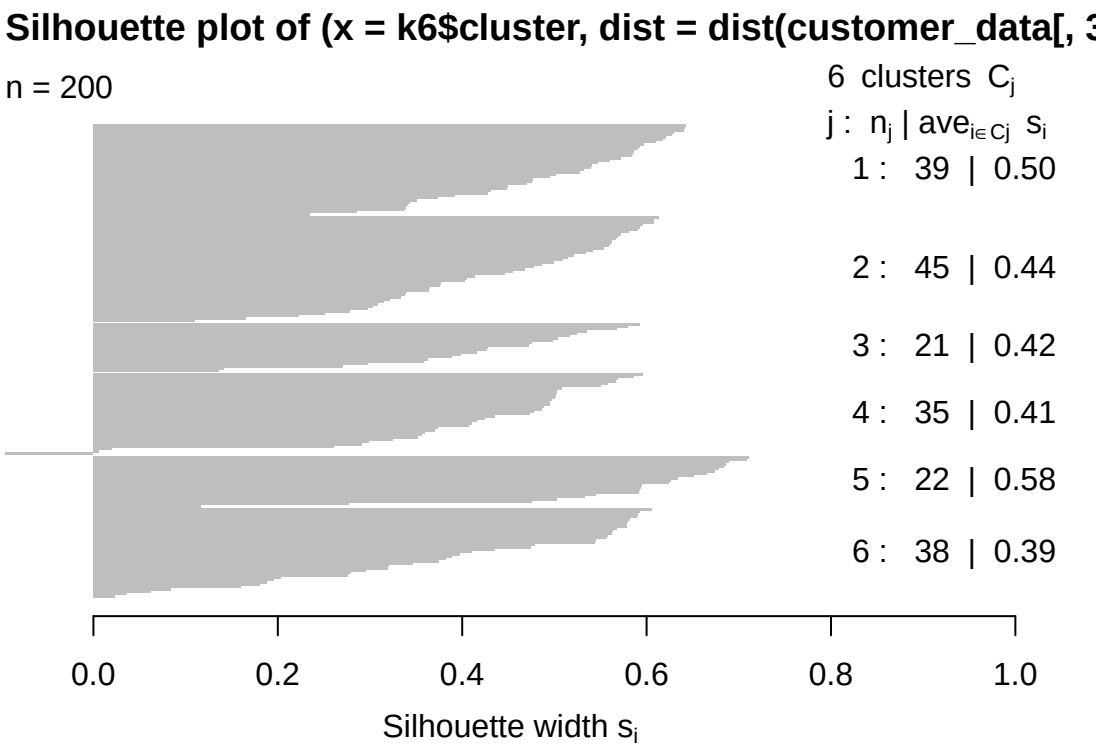


Average silhouette width : 0.41

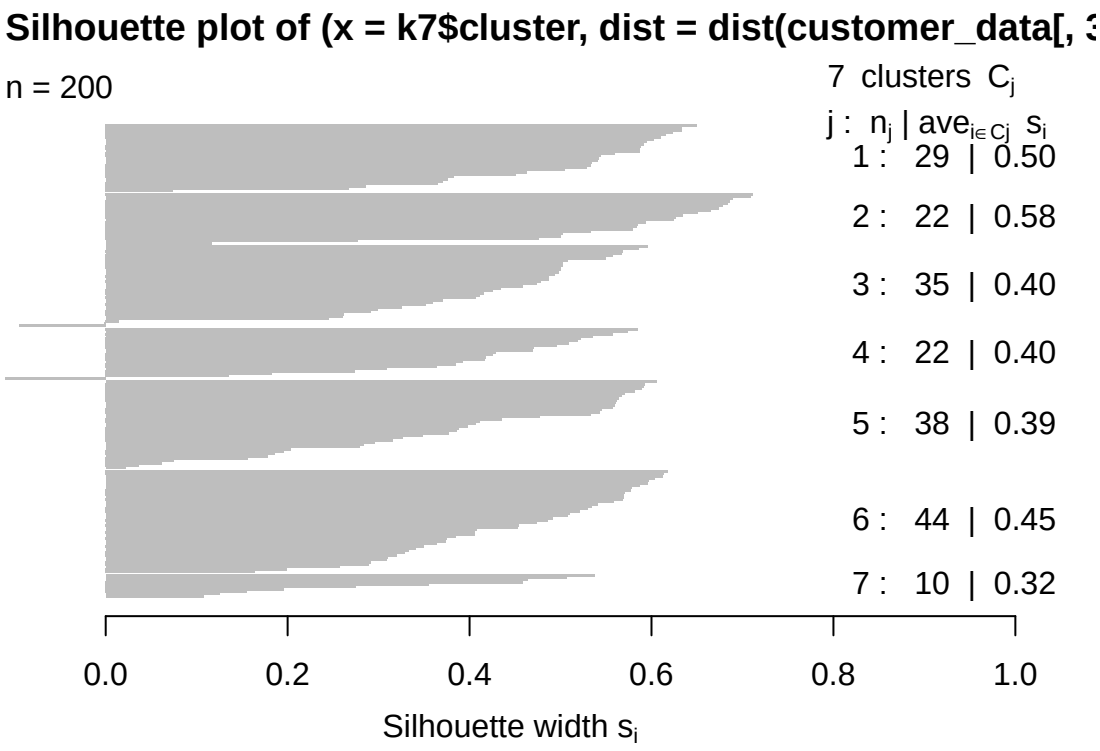
```
k5 <- kmeans(customer_data[, 3:5], 5, iter.max = 100, nstart = 50, algorithm = "Lloyd")
s5 <- plot(silhouette(k5$cluster, dist(customer_data[, 3:5], "euclidean")))
```



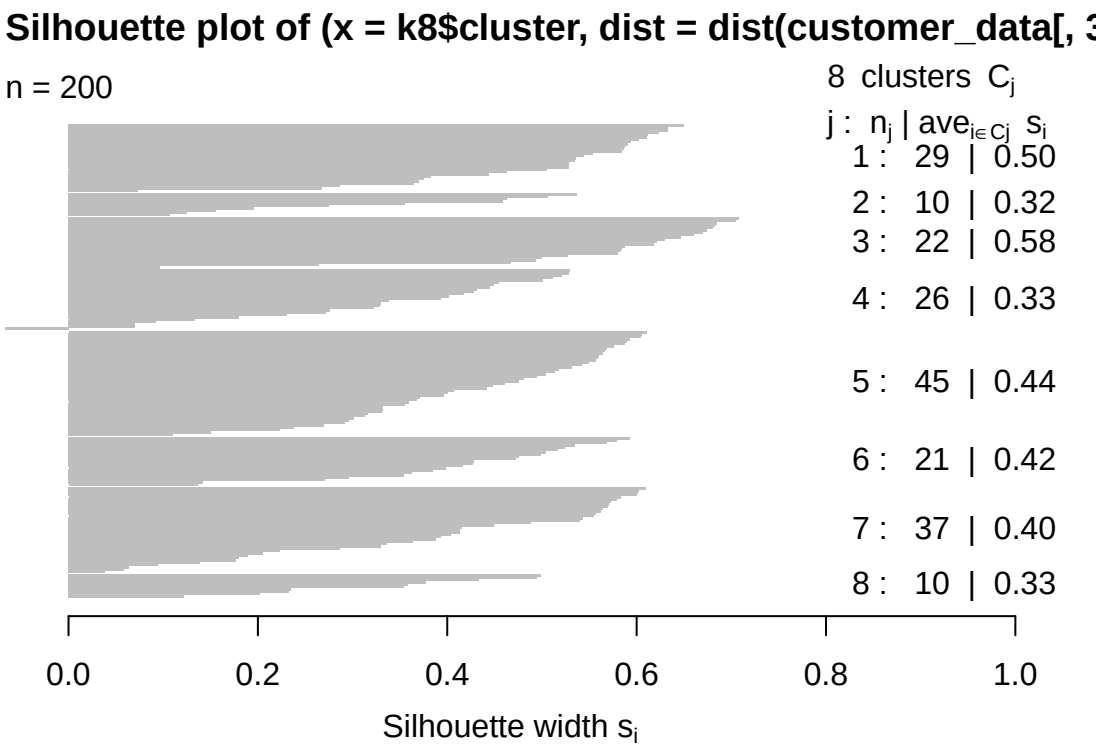

```
k6 <- kmeans(customer_data[, 3:5], 6, iter.max = 100, nstart = 50, algorithm = "Lloyd")
s6 <- plot(silhouette(k6$cluster, dist(customer_data[, 3:5], "euclidean")))
```




```
k7 <- kmeans(customer_data[, 3:5], 7, iter.max = 100,nstart = 50,algorithm = "Lloyd")
s7 <- plot(silhouette(k7$cluster, dist(customer_data[, 3:5], "euclidean")))
```

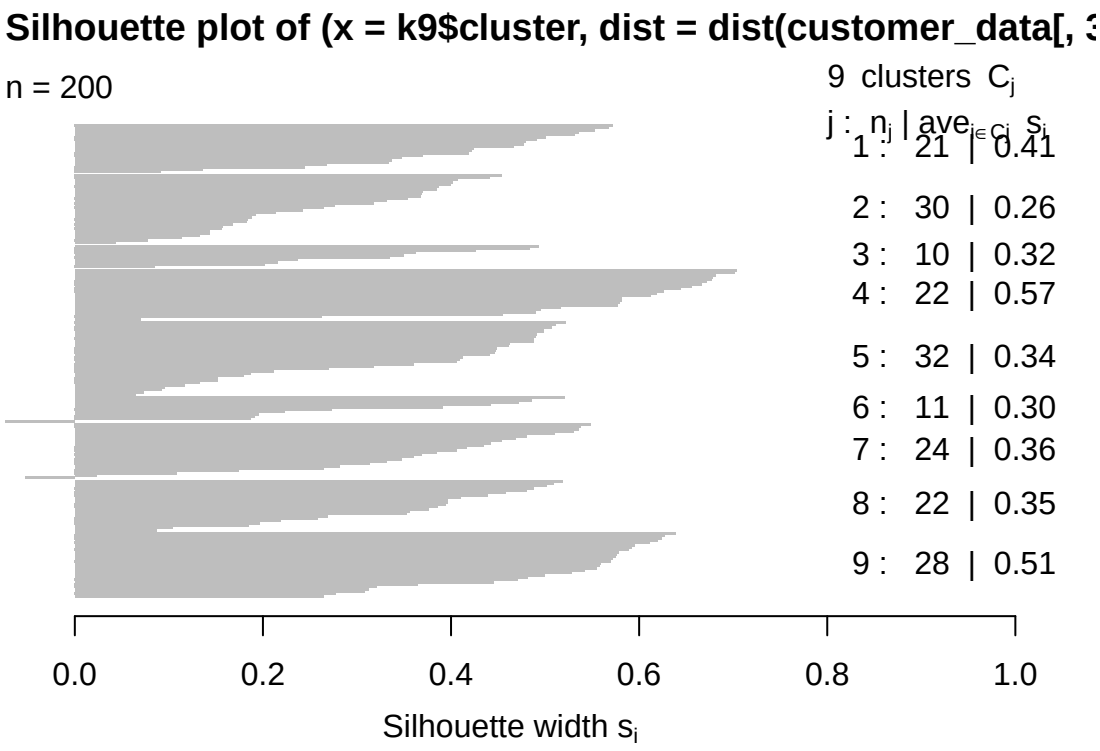



```r
k8 <- kmeans(customer_data[, 3:5], 8, iter.max = 100, nstart = 50, algorithm = "Lloyd")
s8 <- plot(silhouette(k8$cluster, dist(customer_data[, 3:5], "euclidean")))
```



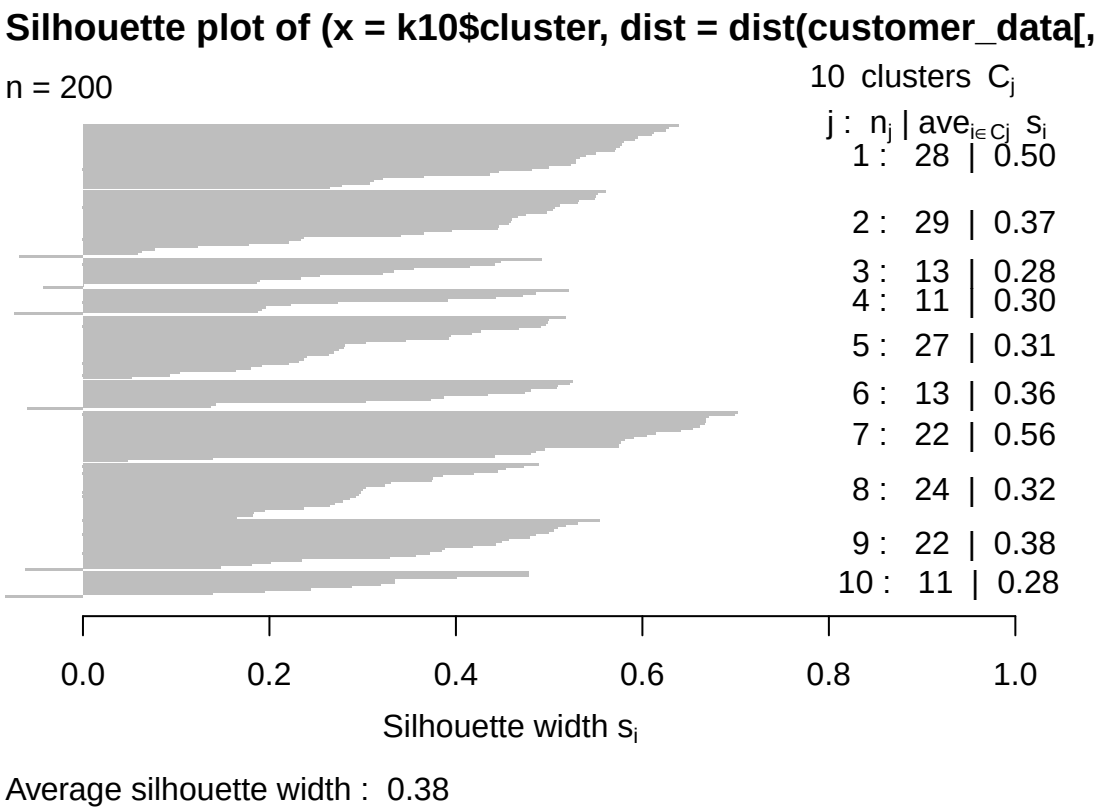
Average silhouette width : 0.43

```
k9 <- kmeans(customer_data[, 3:5], 9, iter.max = 100, nstart = 50, algorithm = "Lloyd")
s9 <- plot(silhouette(k9$cluster, dist(customer_data[, 3:5], "euclidean")))
```



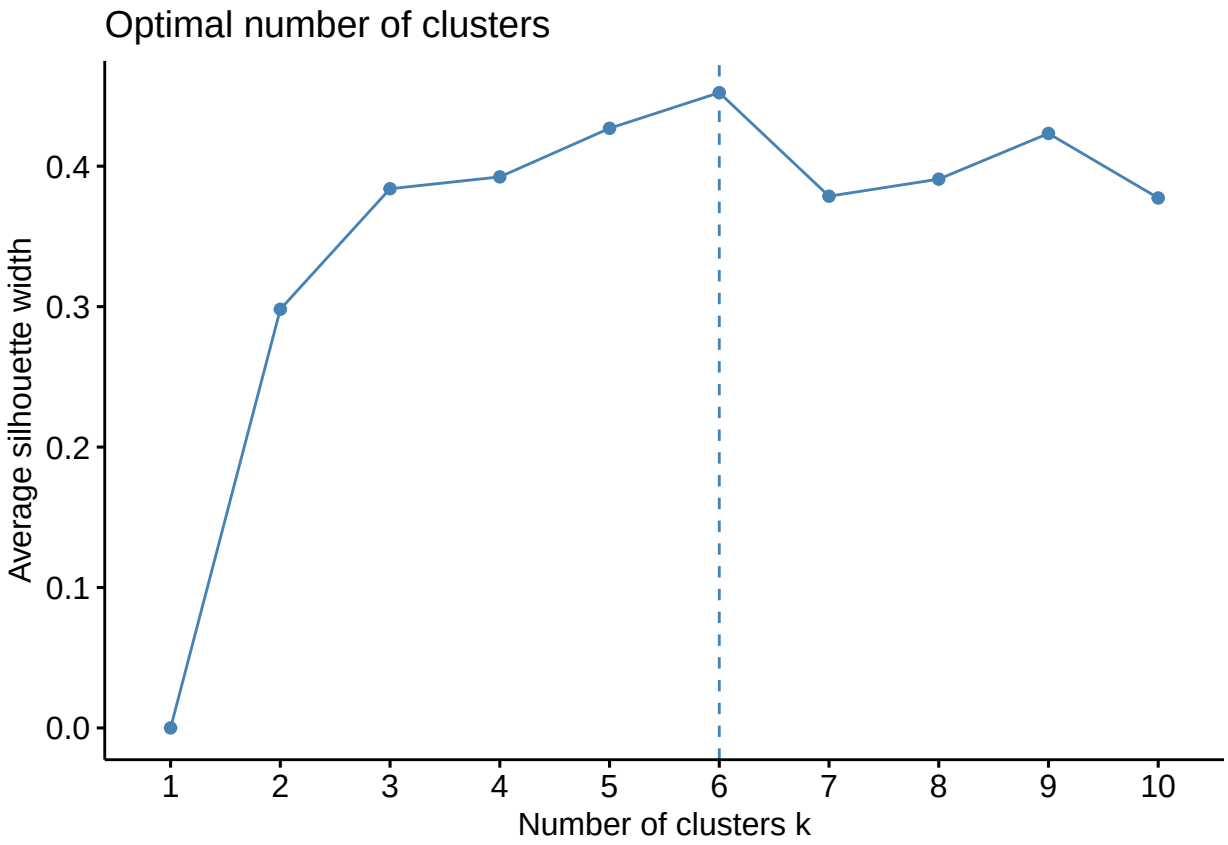
Average silhouette width : 0.39

```
k10 <- kmeans(customer_data[, 3:5], 10, iter.max = 100, nstart = 50, algorithm = "Lloyd")
s10 <- plot(silhouette(k10$cluster, dist(customer_data[, 3:5], "euclidean")))
```



Now, we make use of the fviz_nbclust() function to determine and visualize the optimal number of clusters as follows

```
fviz_nbclust(customer_data[,3:5], kmeans, method = "silhouette")
```



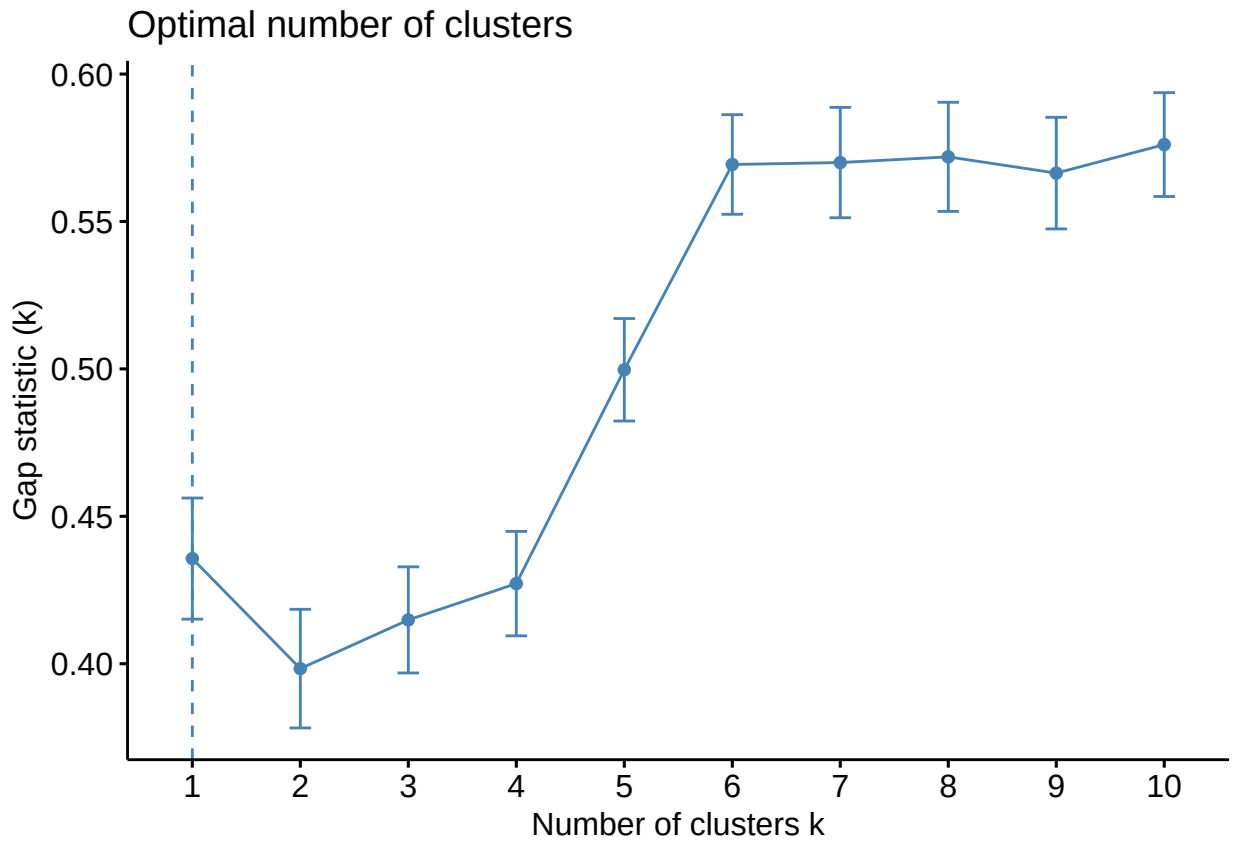
Cluster 6 (k6) has the highest averagae of 45.

3. The Gap Statistic Method Using the gap statistic, one can compare the total intracluster variation for different values of k along with their expected values under the null reference distribution of data. With the help of Monte Carlo simulations, one can produce the sample dataset. For each variable in the dataset, we can calculate the range between min(xi) and max (xj) through which we can produce values uniformly from interval lower bound to upper bound.

We will use the clusGap function for providing gap statistic as well as standard error for a given output.

```
set.seed(125)
stat_gap <- clusGap(customer_data[,3:5], FUN = kmeans, nstart = 25,
            K.max = 10, B = 50)
```

```
fviz_gap_stat(stat_gap)
```



We are now going to take k = 6 since it is our optimal cluster and display its output.

```
k6<-kmeans(customer_data[,3:5],6,iter.max=100,nstart=50,algorithm="Lloyd")
k6
```

```
## K-means clustering with 6 clusters of sizes 45, 22, 21, 38, 35, 39
##
## Cluster means:
##        Age Annual.Income..k.. Spending.Score..1.100.
## 1 56.15556           53.37778               49.08889
## 2 25.27273           25.72727               79.36364
## 3 44.14286           25.14286               19.52381
## 4 27.00000           56.65789               49.13158
## 5 41.68571           88.22857               17.28571
## 6 32.69231           86.53846               82.12821
##
## Clustering vector:
##   [1] 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3
##  [38] 2 3 2 1 2 1 4 3 2 1 4 4 4 1 4 4 1 1 1 1 1 4 1 1 4 1 1 1 4 1 1 4 4 1 1 1 1
##  [75] 1 4 1 4 4 1 1 4 1 1 4 1 1 4 4 1 1 4 1 4 4 4 1 4 1 4 4 1 1 4 1 4 1 1 1 1 1
## [112] 4 4 4 4 4 1 1 1 1 4 4 4 6 4 6 5 6 5 6 5 6 4 6 5 6 5 6 5 6 5 6 4 6 5 6 5 6
## [149] 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5
## [186] 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6
##
## Within cluster sum of squares by cluster:
## [1]  8062.133  4099.818  7732.381  7742.895 16690.857 13972.359
```

```
##  (between_SS / total_SS =  81.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

From the output of our k-means operation, we observe a list with several key information. From this, we conclude the useful information being:

cluster - This is a vector of several integers that denote the cluster which has an allocation of each point. **venters** - Matrix comprising of several cluster centers **totss** - This represents the total sum of squares. **withinss** - This is a vector representing the intra-cluster sum of squares having one component per cluster. **tot.withinss** - This denotes the total intra-cluster sum of squares. **betweenss** - This is the sum of between-cluster squares. **size** - The total number of points that each cluster holds.

Plotting the Clustering Results

We are going to visualize using the Principle Component Analysis

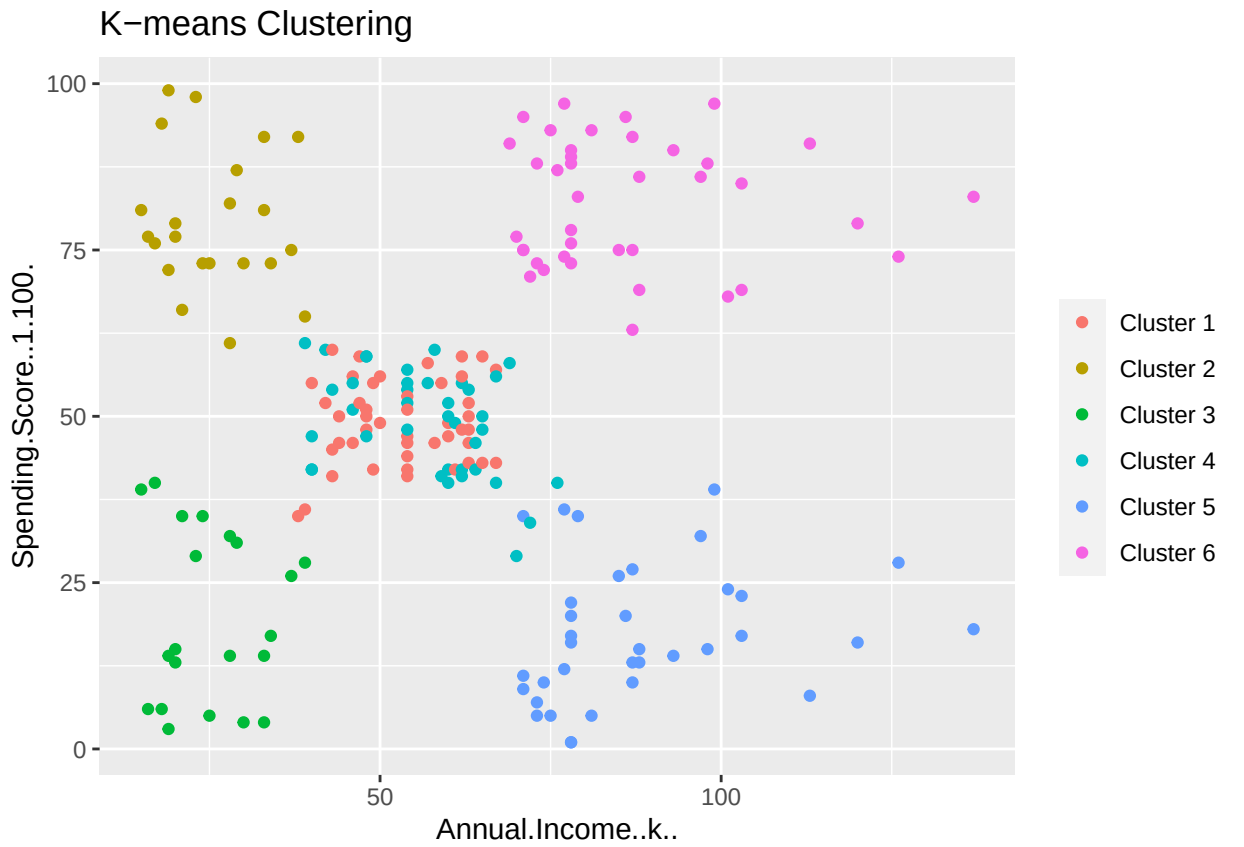
```
pcclust = prcomp(customer_data[, 3:5], scale = FALSE) #principal component analysis
summary(pcclust)
```

```
## Importance of components:
##                             PC1     PC2     PC3
## Standard deviation     26.4625 26.1597 12.9317
## Proportion of Variance  0.4512  0.4410  0.1078
## Cumulative Proportion   0.4512  0.8922  1.0000
```

```
pcclust$rotation[, 1:2]
```

```
##                                PC1        PC2
## Age                     0.1889742 -0.1309652
## Annual.Income..k..     -0.5886410 -0.8083757
## Spending.Score..1.100. -0.7859965  0.5739136
```

```
set.seed(1)
ggplot(customer_data, aes(x = Annual.Income..k.., y = Spending.Score..1.100.)) +
  geom_point(stat = "identity", aes(color = as.factor(k6$cluster))) +
  scale_color_discrete(name = " ",
                       breaks=c("1", "2", "3", "4", "5","6"),
                       labels=c("Cluster 1", "Cluster 2", "Cluster 3", "Cluster 4", "Cluster 5", "Cluster 6")) +
  ggtitle("K-means Clustering")
```



From the above visualization, we observe that there is a distribution of 6 clusters as follows –

Cluster 1 – This cluster represents the customer data having a high annual income as well as a high annual spend.

Cluster 2 – This cluster denotes a high annual income and low yearly spend.

Cluster 3 – This cluster denotes the customer data with low annual income as well as low yearly spend of income.

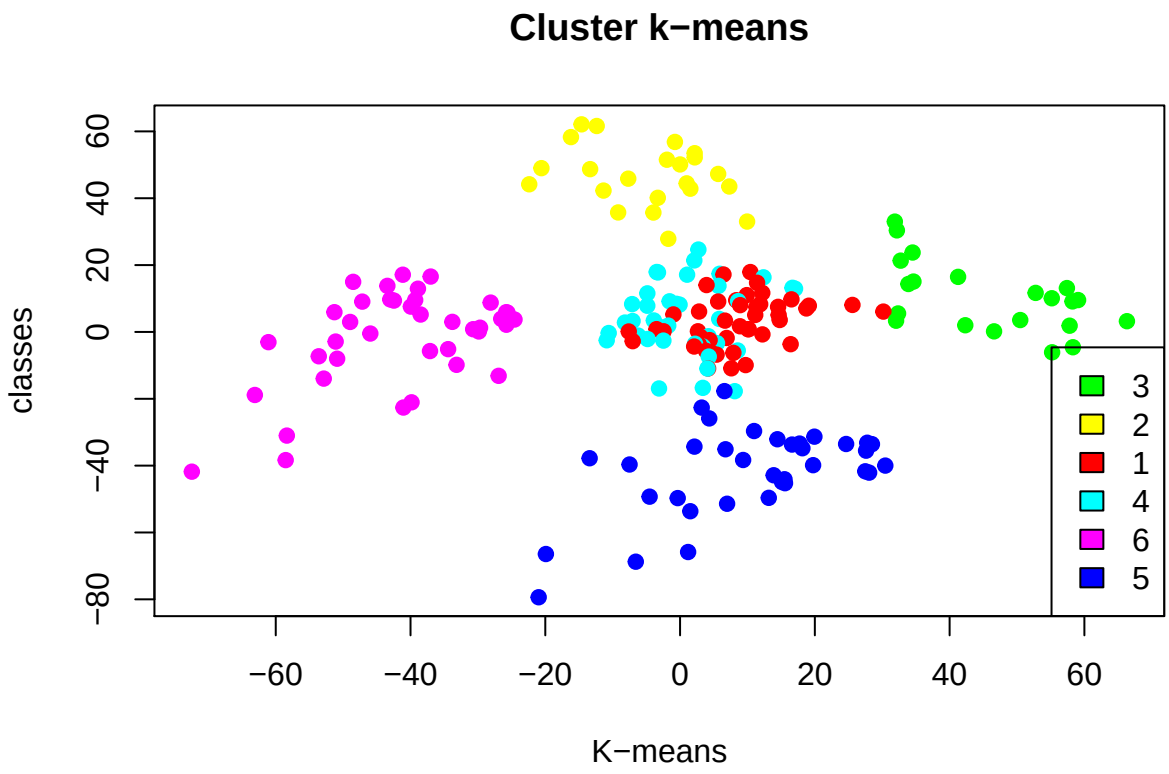
Cluster 5 – This cluster represents a low annual income but its high yearly expenditure.

Cluster 6 and 4 – These clusters represent the customer data with the medium income salary as well as the medium annual spend of salary.

```
#ploting k-means against the clusters
kCols = function(vec){
  cols = rainbow (length (unique (vec)))
return (cols[as.numeric(as.factor(vec))])
  }
```

```
digCluster <- k6$cluster;
dignm <- as.character(digCluster); # K-means clusters
```

```
plot(pcclust$x[,1:2], #the principle component algorithm
     col = kCols(digCluster), pch = 19, xlab = "K-means", ylab = "classes",
     main = "Cluster k-means")
legend("bottomright", unique(dignm), fill=unique(kCols(digCluster)))
```



Principal Component Analysis (PCA) is a useful technique for exploratory data analysis allowing you to better visualize the variation present in a dataset.

From the graph above we can see that:

Cluster 1 and 4 – These clusters consist of customers with medium PCA1 and medium PCA2 score.

Cluster 2 – This comprises of customers with a high PCA2 and a medium annual spend of income.

Cluster 3 – This cluster comprises of customers with a high PCA1 income and a high PCA2.

Cluster 5 – In this cluster, there are customers with a medium PCA1 and a low PCA2 score.

Cluster 6 – This cluster represents customers having a high PCA2 and a low PCA1.

Conclusion

In this data science project, we went through the clients clustering model. We developed this using a class of machine learning known as unsupervised learning. Specifically, we made use of a clustering algorithm called K-means clustering. We analyzed and visualized the data and then proceeded to implement our algorithm.

The dataset used throughpout this project was obta from https://www.kaggle.com/datasets.