Report - Customer Segmentation - Capstone Project

Nitesh Kumar Gupta - Harvard Data Science Professional

11/29/2020

Contents

1 Requirement/Problem Statement

For this project, you will be applying machine learning techniques that go beyond standard linear regression. You will have the opportunity to use a publicly available dataset to solve the problem of your choice. You are strongly discouraged from using well-known datasets, particularly ones that have been used as examples in previous courses or are similar to them (such as the iris, titanic, mnist, or movielens datasets, among others) - this is your opportunity to branch out and explore some new data! The UCI Machine Learning Repository and Kaggle are good places to seek out a dataset. Kaggle also maintains a curated list of datasets that are cleaned and ready for machine learning analyses. Your dataset must be automatically downloaded in your code or included with your submission. You may not submit the same project for both the MovieLens and Choose Your Own project submissions.

2 Executive Summary

In this machine learning project, we made use of **K-means** clustering algorithm which is the essential algorithm for clustering unlabeled dataset. We went through mall customer data and implemented **unsupervised clustering algorithm K-Means** to group customers into **six clusters** based on three features i.e. Age, Annual Income, and Spending Score. Overall, interesting outcome of this project was that we successfully identified group of customers whom business can target using different **marketing techniques** such as special offers or discounts to **increase the mall's revenue**.

3 Introduction to Customer Segmentation

Customer Segmentation is the process of dividing customer base into several groups of individuals that share a similarity in different ways that are relevant to marketing such as gender, age, interests, and miscellaneous spending habits.

Businesses that deploy customer segmentation are under the notion that every customer has different requirements and require a specific marketing effort to address them appropriately. They aim to gain a deeper approach of the customer they are targeting. Therefore, their aim has to be specific and should be tailored to address the requirements of each and every individual customer. Furthermore, through the data collected, business can gain a deeper understanding of customer preferences as well as the requirements for discovering valuable segments that would reap them maximum profit. This way, they can strategize their marketing techniques more efficiently and minimize the possibility of risk to their investment.

The technique of customer segmentation is dependent on several key differentiators that divide customers into groups to be targeted. Data related to demographics, geography, economic status as well as behavioral patterns play a crucial role in determining the company direction towards addressing the various segments.

4 EDA - Exploratory Data Analysis

4.1 Steps to implement Customer Segmentation

In the first step of this data science project, we will perform data exploration. We will import the essential packages required for this role and then read our data. Finally, we will go through the input data to gain necessary insights about it.

Installing and loading required R packages

```
################################################################################
# Install required packages if not installed already
# Note: this process could take a couple of minutes
################################################################################

if(!require(readr)) install.packages("readr")
if(!require(purrr)) install.packages("purrr")
if(!require(cluster)) install.packages("cluster")
if(!require(gridExtra)) install.packages("gridExtra")
if(!require(grid)) install.packages("grid")
if(!require(NbClust)) install.packages("NbClust")
if(!require(factoextra)) install.packages("factoextra")
if(!require(kableExtra)) install.packages("kableExtra")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(dplyr)) install.packages("dplyr")

################################################################################
# Load required libraries
################################################################################

library (readr)
library(purrr)
library(cluster)
library(gridExtra)
library(grid)
library(NbClust)
library(factoextra)
library(kableExtra)
library(tidyverse)
library(dplyr)
```

Loading customer data into R data frame.

Note: To make script work on any machine, we have removed input data location dependency by making data available on internet using Github repository. Raw Github URL to the repository file has been used to load data.

```
# Customer data file url
customer_file_url="https://raw.githubusercontent.com/niteshn99/Edx-HarvardX-CustomerSegment/main/data/Ma
# read csv file into dataframe
customer_data <- read_csv(url(customer_file_url))
```

```
## Parsed with column specification:
## cols(
##   CustomerID = col_double(),
##   Gender = col_character(),
##   Age = col_double(),
##   `Annual Income (k$)` = col_double(),
##   `Spending Score (1-100)` = col_double()
## )
```

4.2 Initial data exploration

Lets have a look at first six rows of data followed by data frame structure and features name to learn more about features within it.

Structure of customer data

```
str(customer_data)
```

```
## tibble [200 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ CustomerID            : num [1:200] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Gender                : chr [1:200] "Male" "Male" "Female" "Female" ...
##  $ Age                   : num [1:200] 19 21 20 23 31 22 35 23 64 30 ...
##  $ Annual Income (k$)    : num [1:200] 15 15 16 16 17 17 18 18 19 19 ...
##  $ Spending Score (1-100): num [1:200] 39 81 6 77 40 76 6 94 3 72 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   CustomerID = col_double(),
##   ..   Gender = col_character(),
##   ..   Age = col_double(),
##   ..   `Annual Income (k$)` = col_double(),
##   ..   `Spending Score (1-100)` = col_double()
##   .. )
```

From the customer data structure, we can conclude that input data contains 200 observation of 5 variables.

Features in customer data

```
# Columns/Features names
names(customer_data)
```

```
## [1] "CustomerID"             "Gender"                 "Age"
## [4] "Annual Income (k$)"     "Spending Score (1-100)"
```

Customer data contains following features

- **CustomerID** - is a numeric column containing unique customer id to distinguish each customer.
- **Gender** - is a character column containing gender of respective customer.
- **Age** - is a numeric column containing age of respective customer.
- **Annual Income (k$)** - is a numeric column containing annual income of respective customer in currency dollars.
- **Spending Score (1-100)** - is a numeric column containing respective customer spending score between 1 to 100.

Missing values in customer data

```
sapply(customer_data, function(x) sum(is.na(x))) %>%
kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

	x
CustomerID	0
Gender	0
Age	0
Annual Income (k$)	0
Spending Score (1-100)	0

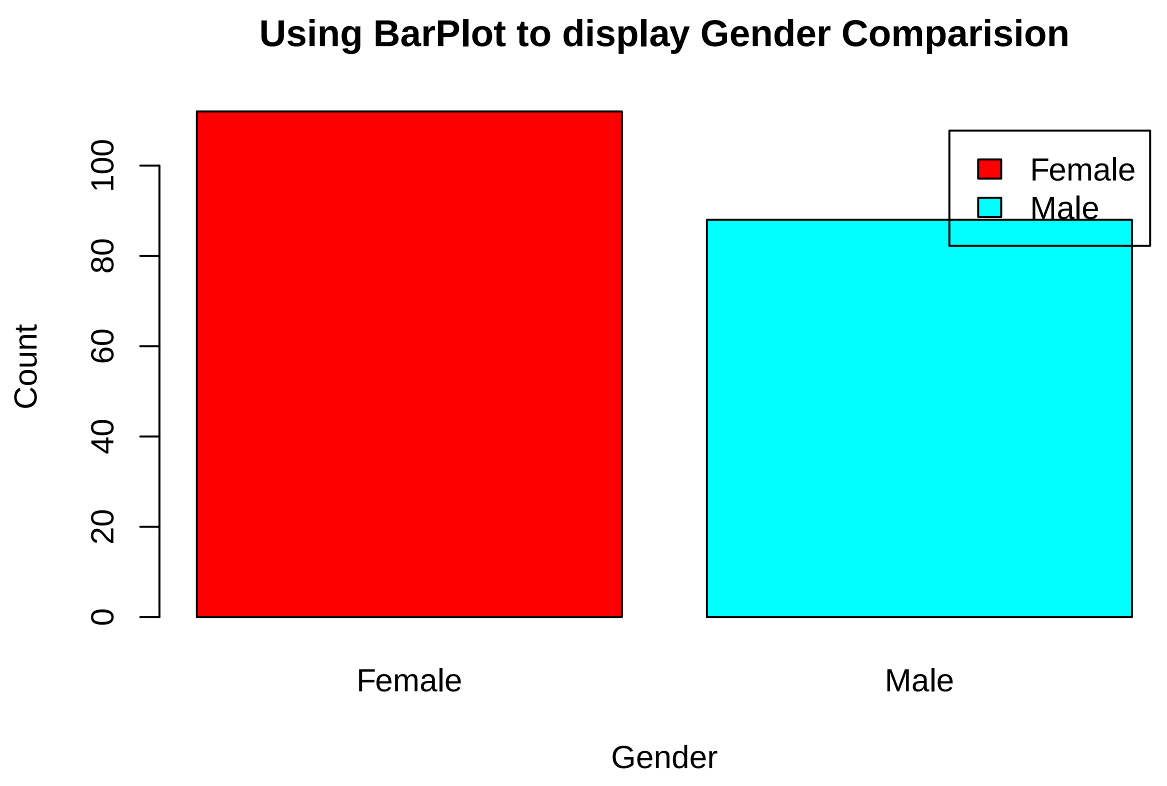
First six rows of customer data.

```
head(customer_data) %>%
  kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                  position = "center",
                  font_size = 10,
                  full_width = FALSE)
```

CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40
6	Female	22	17	76

4.3 Customer Gender Distribution

```
a=table(customer_data$Gender)
barplot(a,main="Using BarPlot to display Gender Comparision",
        ylab="Count",
        xlab="Gender",
        col=rainbow(2),
        legend=rownames(a))
```



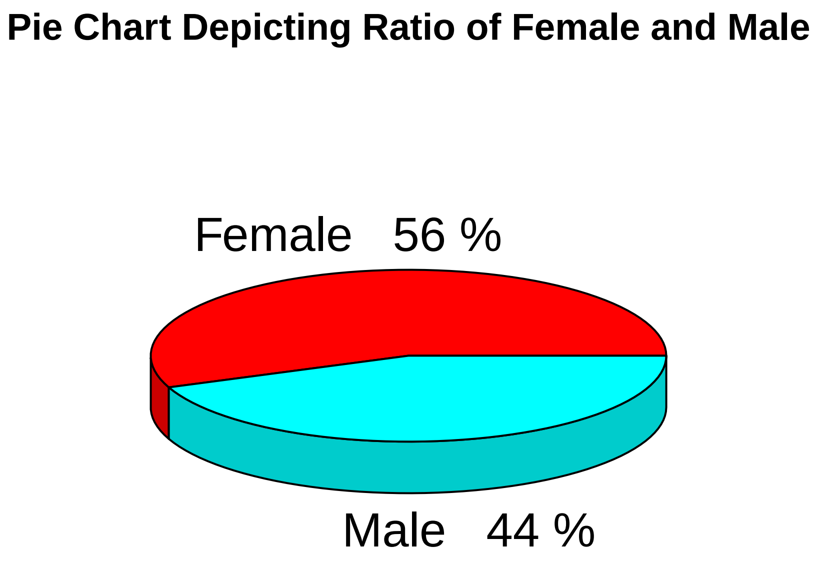
Above bar plot shows gender distribution and it is clear from graph is that number of female shopper is higher than male shopper. Lets have a look at that in ratio proportion.

4.3.1 Ratio of male and female distribution

```
pct=round(a/sum(a)*100)
lbs=paste(c("Female","Male")," ",pct,"%",sep=" ")
library(plotrix)
```

```
## Warning: package 'plotrix' was built under R version 4.0.3
```

```
pie3D(a,labels=lbs,
      main="Pie Chart Depicting Ratio of Female and Male")
```



From the above graph, we conclude that percentage of females is 56% whereas percentage of male in customer dataset is 44%.

4.4 Customer Age Distribution

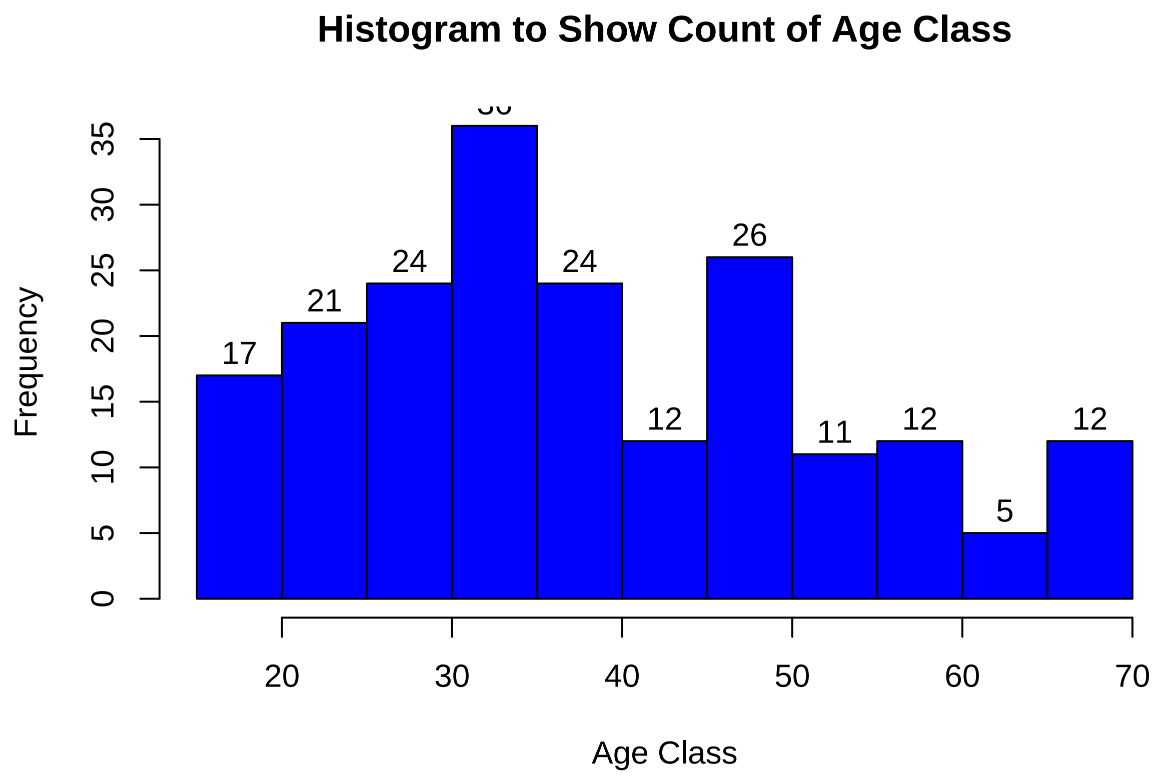
```
sd(customer_data$Age)
```

```
## [1] 13.96901
```

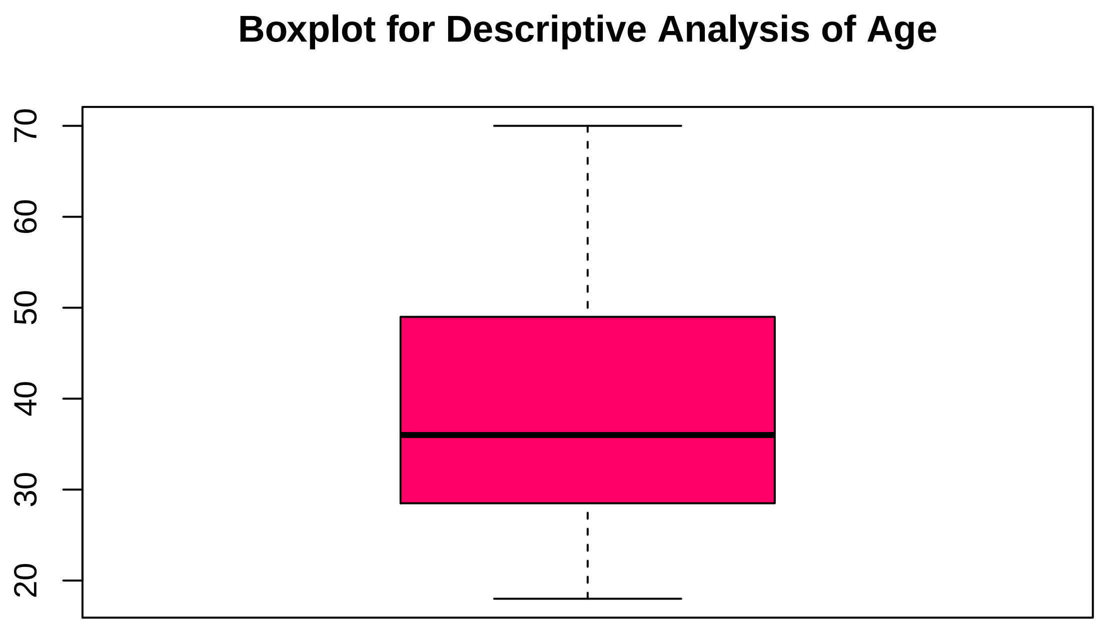
```
summary(customer_data$Age)
```

```
##    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="blue",
     main="Histogram to Show Count of Age Class",
     xlab="Age Class",
     ylab="Frequency",
     labels=TRUE)
```



```
boxplot(customer_data$Age,
        col="#ff0066",
        main="Boxplot for Descriptive Analysis of Age")
```



From above two graphs, we conclude that maximum number of customers are in age group between 30 and 50 years. Minimum age of customer is 18 whereas maximum age is 70.

4.5 Customer Annual Income Distribution

```
sd(customer_data$`Annual Income (k$)`)
```

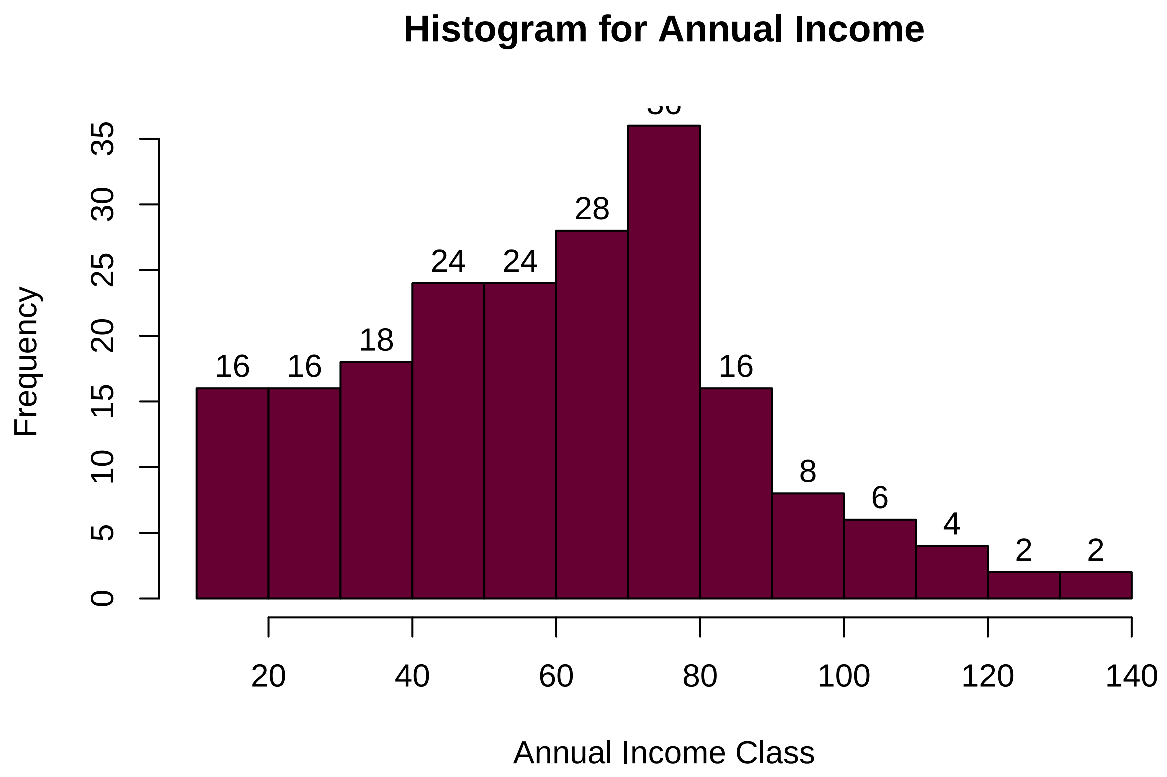
```
## [1] 26.26472
```

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

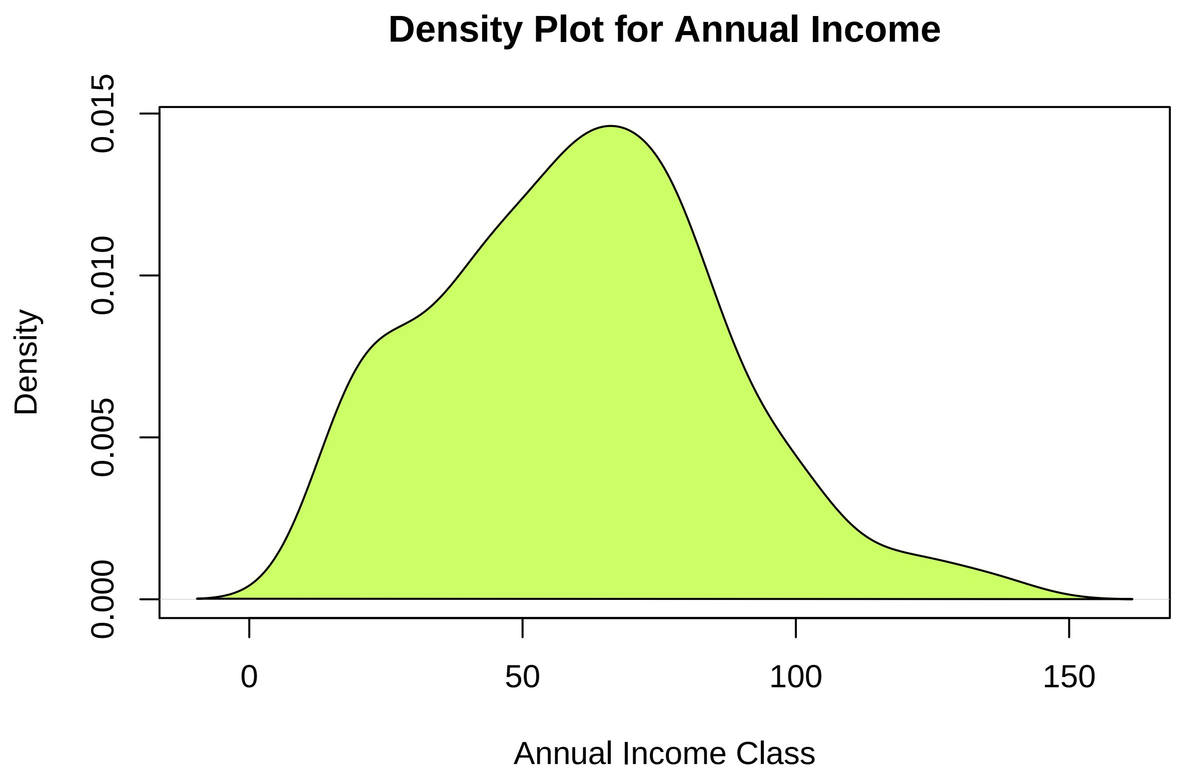
```
## Warning: Unknown or uninitialised column: `Annual.Income..k..`.
```

```
## Length  Class   Mode
##      0   NULL   NULL
```

```
hist(customer_data$`Annual Income (k$)`,
     col="#660033",
     main="Histogram for Annual Income",
     xlab="Annual Income Class",
     ylab="Frequency",
     labels=TRUE)
```



```
## Annual income density plot
plot(density(customer_data$`Annual Income (k$)`),
     col="yellow",
     main="Density Plot for Annual Income",
     xlab="Annual Income Class",
     ylab="Density")
polygon(density(customer_data$`Annual Income (k$)`),
        col="#ccff66")
```



From the above descriptive analysis, we conclude that the minimum annual income of the customer is 15 and maximum income is approximately 140. Average income having highest frequency count based histogram distribution is 70; however, average average income of all customer is approximately equal to 60 or 61. Finally, based on Kernel distribution plot we can conclude that annual income has a normal distribution.

4.6 Customer Spending Score

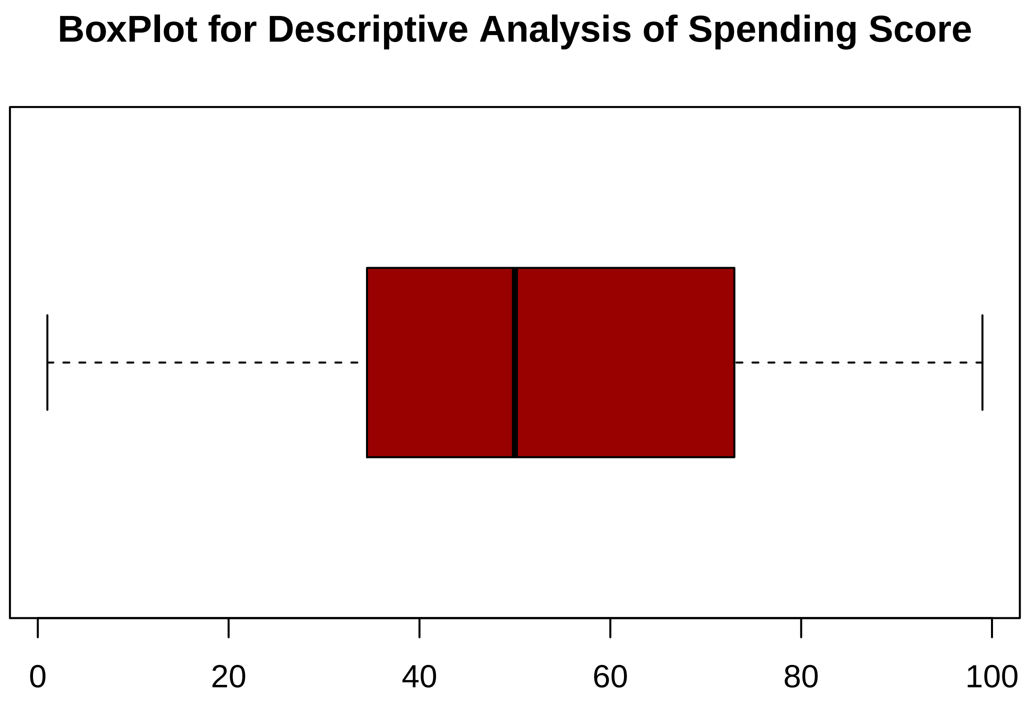
```r
sd(customer_data$`Spending Score (1-100)`)
```

```
## [1] 25.82352
```

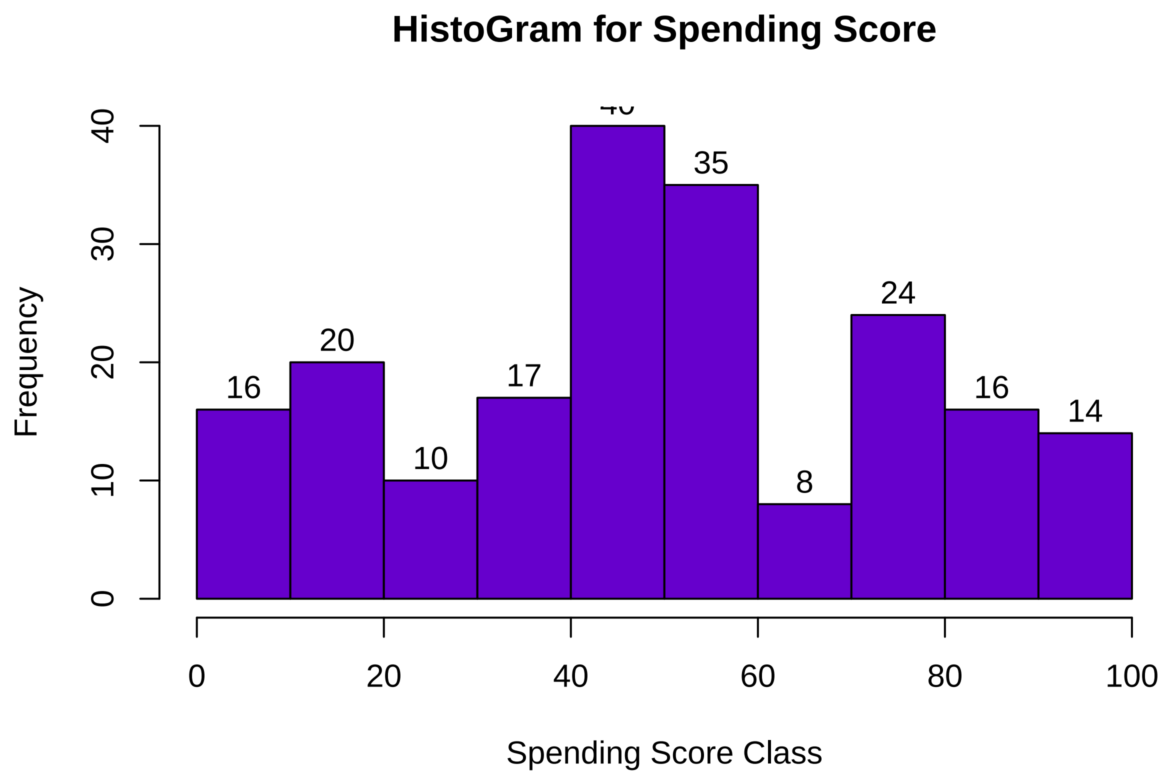
```r
summary(customer_data$`Spending Score (1-100)`)
```

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

```r
boxplot(customer_data$`Spending Score (1-100)`,
        horizontal=TRUE,
        col="#990000",
        main="BoxPlot for Descriptive Analysis of Spending Score")
```



```
hist(customer_data$`Spending Score (1-100)`,
     main="HistoGram for Spending Score",
     xlab="Spending Score Class",
     ylab="Frequency",
     col="#6600cc",
     labels=TRUE)
```



Descriptive analysis on spending score shows that minimum score is 1, maximum is 99, and average score is 50.20. From the histogram plot, we can conclude that customers between class 40 and 50 have highest spending score among all the classes.

5 Customer Segmentation using K-means 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. Below is a graphical flow of k-means algorithm flow which will further help us understand it better.

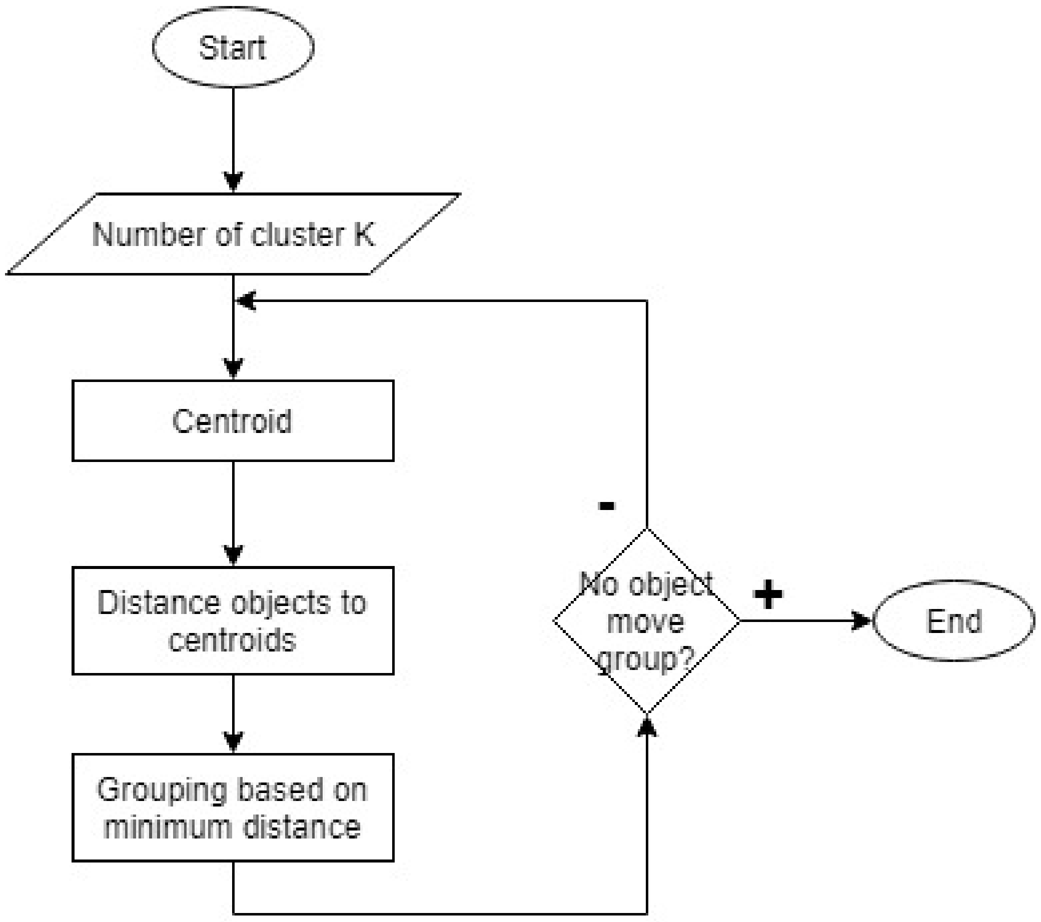

Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
End

Figure 1: Algorithm flow

5.1 Determining Optimal Clusters

While working with clusters, we need to specify the number of clusters to use. We would like to utilize the optimal number of clusters. To help us in determining the optimal clusters, there are three popular methods

- Elbow method
- Silhouette method
- Gap statistic

5.1.1 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.

$$\text{minimize(sum W(Ck))}, k = 1 \ldots k$$

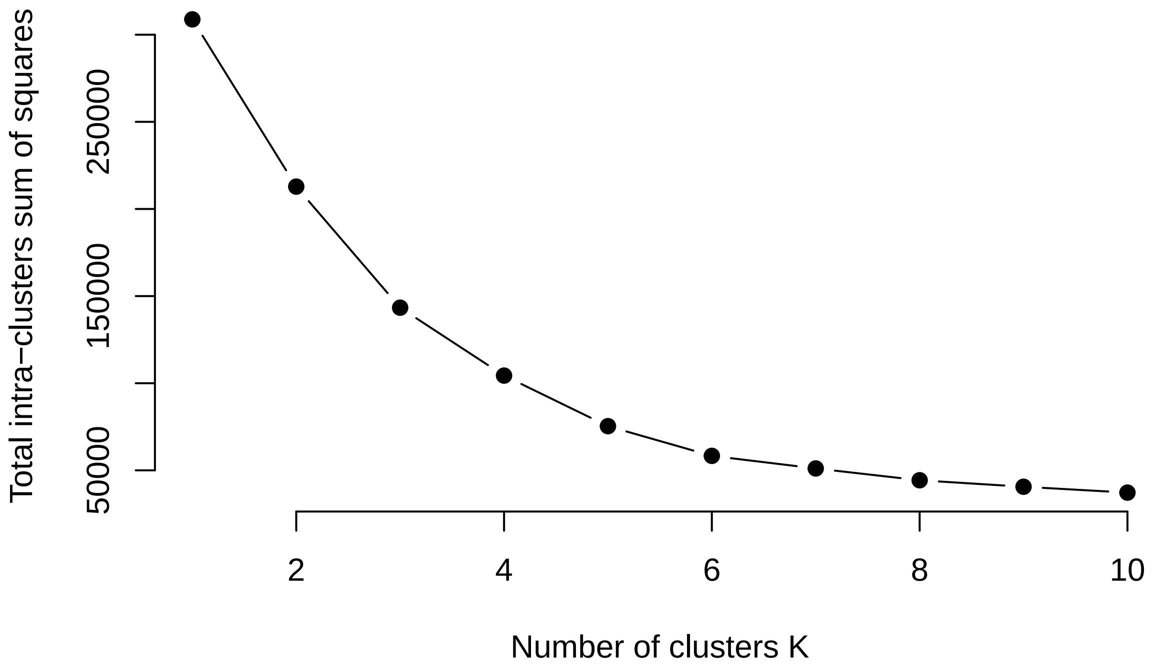
Where Ck represents the kth cluster and W(Ck) denotes the intra-cluster variation. With the measurement of the total intra-cluster variation, one can evaluate the compactness of the clustering boundary. We can then proceed to define the optimal clusters as follows –

First, we calculate the clustering algorithm for several values of k. This can be done by creating a variation within k from 1 to 10 clusters. We 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, sample.kind = "Rounding")
```

```
## Warning in set.seed(123, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
# Function to calculate total intra-cluster sum of square
iss <- function(k) {
  kmeans(customer_data[,3:5],k,iter.max=100,nstart=100,algorithm="Lloyd" )$tot.withinss
}
# Range of K values out of which we want to determine best optimal k value
k.values <- 1:10
#Intra-cluster sum of square for each K values
iss_values <- map_dbl(k.values, iss)
# Elbow curve plot to determine best optimal K value
plot(k.values, iss_values,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total intra-clusters sum of squares")
```

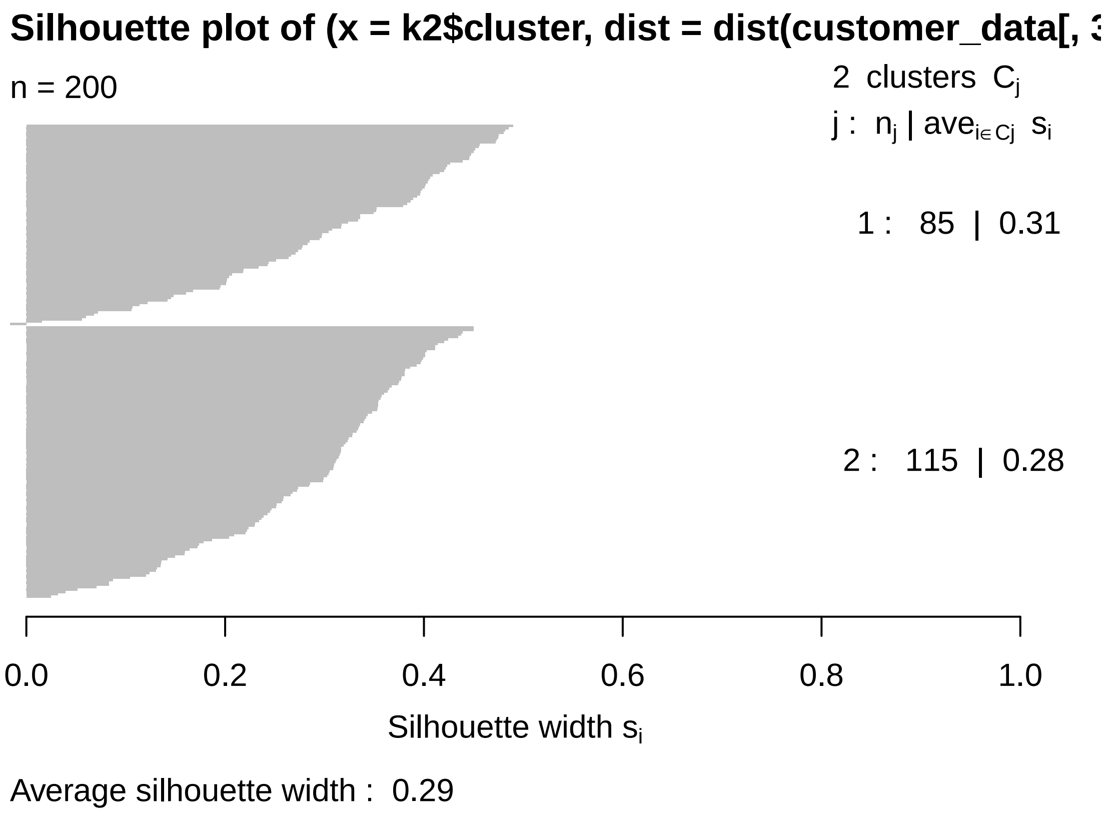


From above plot, we can conclude that K = 4 is the appropriate number of clusters since it seems to be appearing at the bend in the elbow curve plot.

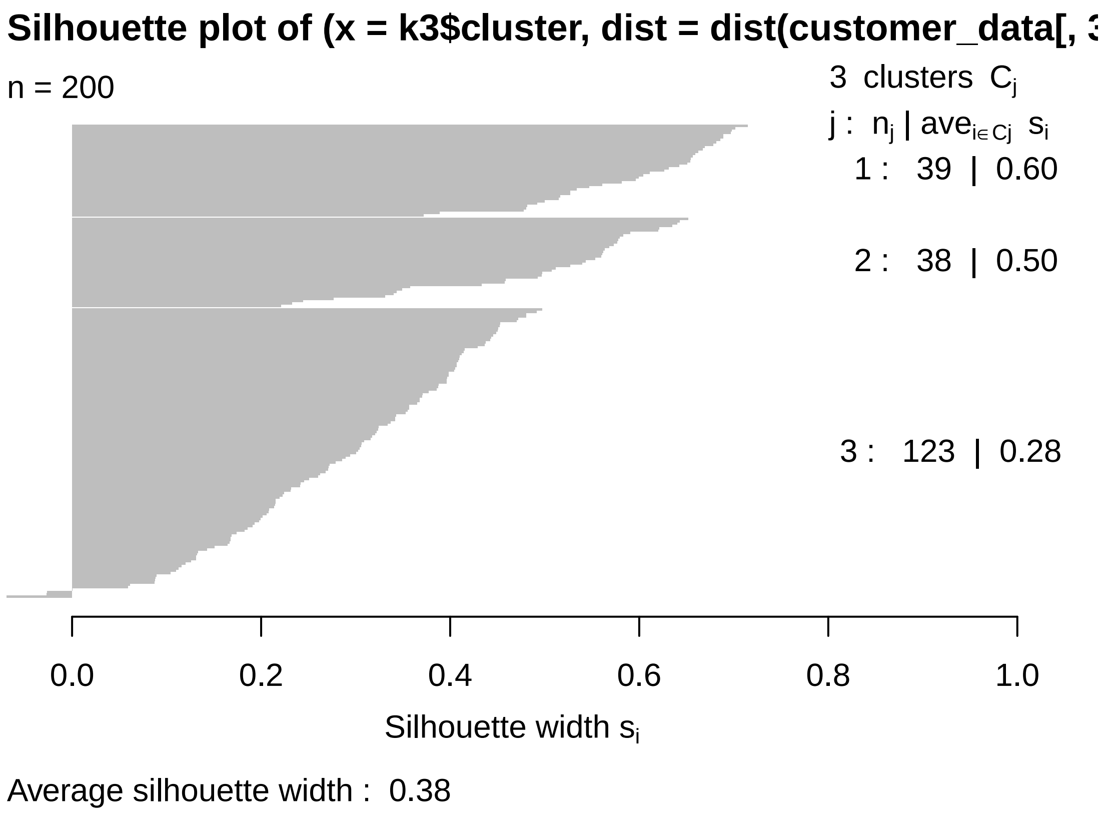
5.1.2 Average Silhouette Method

With the help of the average silhouette method, we can measure the quality of our clustering operation. With this, we can determine how well within the cluster is the data object. 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. Using the silhouette function in the cluster package, we can compute the average silhouette width using the kmean function.

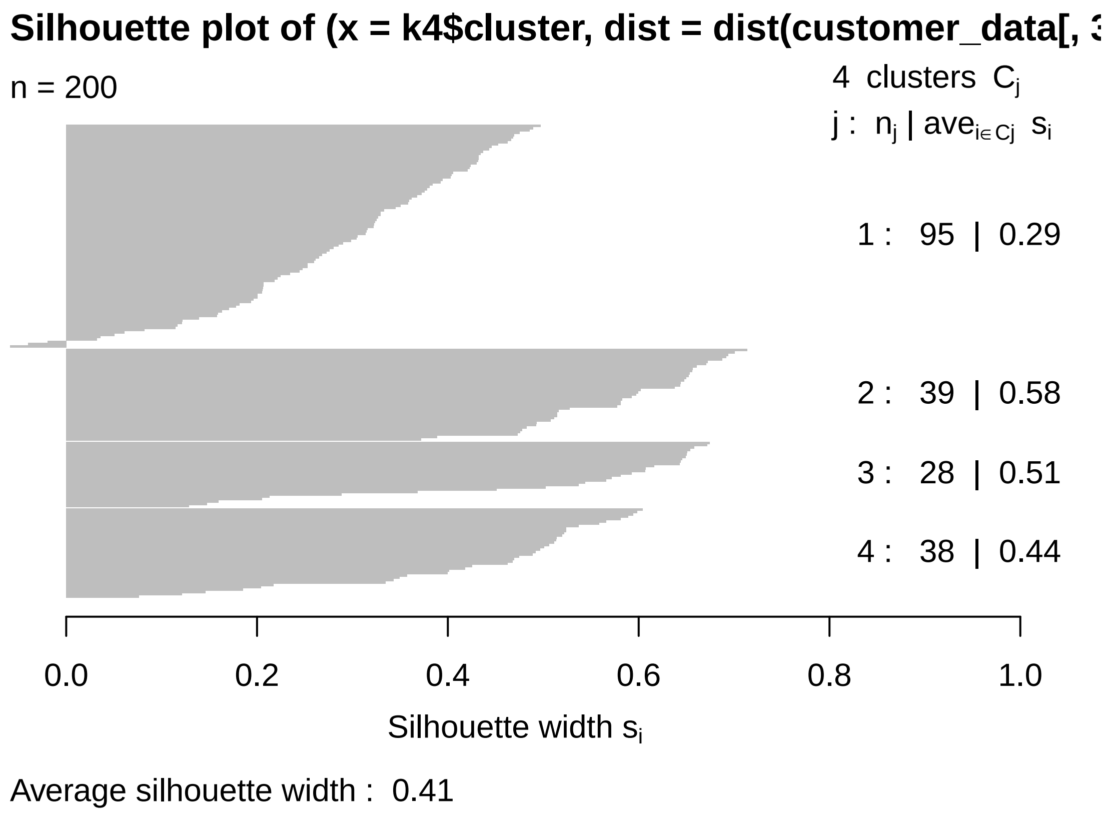
```r
# K = 2 clusters
k2<-kmeans(customer_data[,3:5],2,iter.max=100,nstart=50,algorithm="Lloyd")
s2<-plot(silhouette(k2$cluster,dist(customer_data[,3:5],"euclidean")))
```



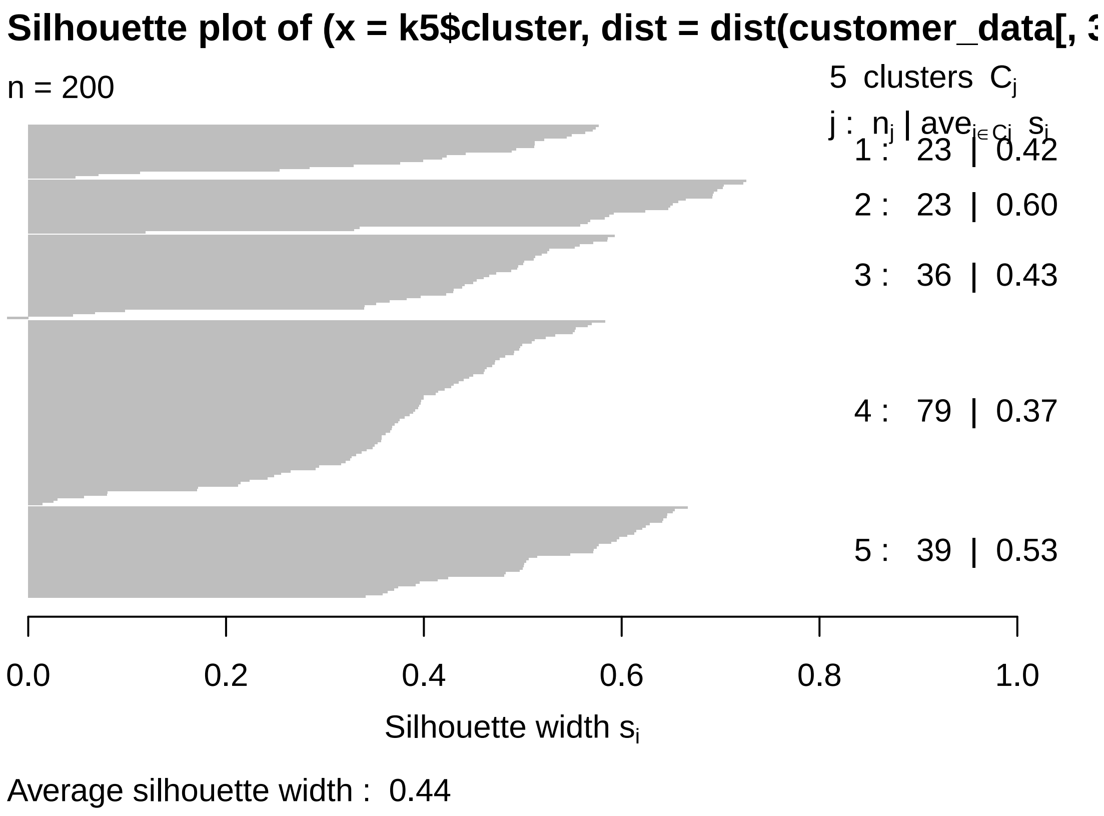
```r
# K = 3 clusters
k3<-kmeans(customer_data[,3:5],3,iter.max=100,nstart=50,algorithm="Lloyd")
s3<-plot(silhouette(k3$cluster,dist(customer_data[,3:5],"euclidean")))
```



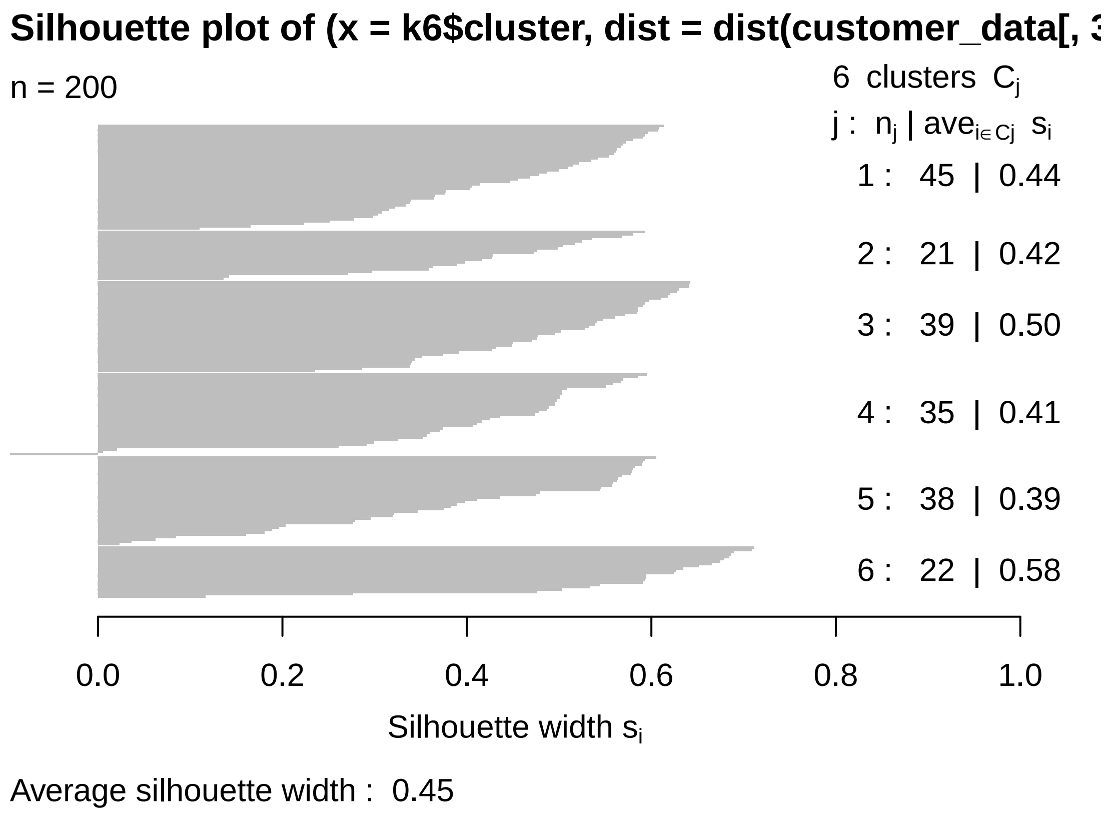
```
# K = 4 clusters
k4<-kmeans(customer_data[,3:5],4,iter.max=100,nstart=50,algorithm="Lloyd")
s4<-plot(silhouette(k4$cluster,dist(customer_data[,3:5],"euclidean")))
```



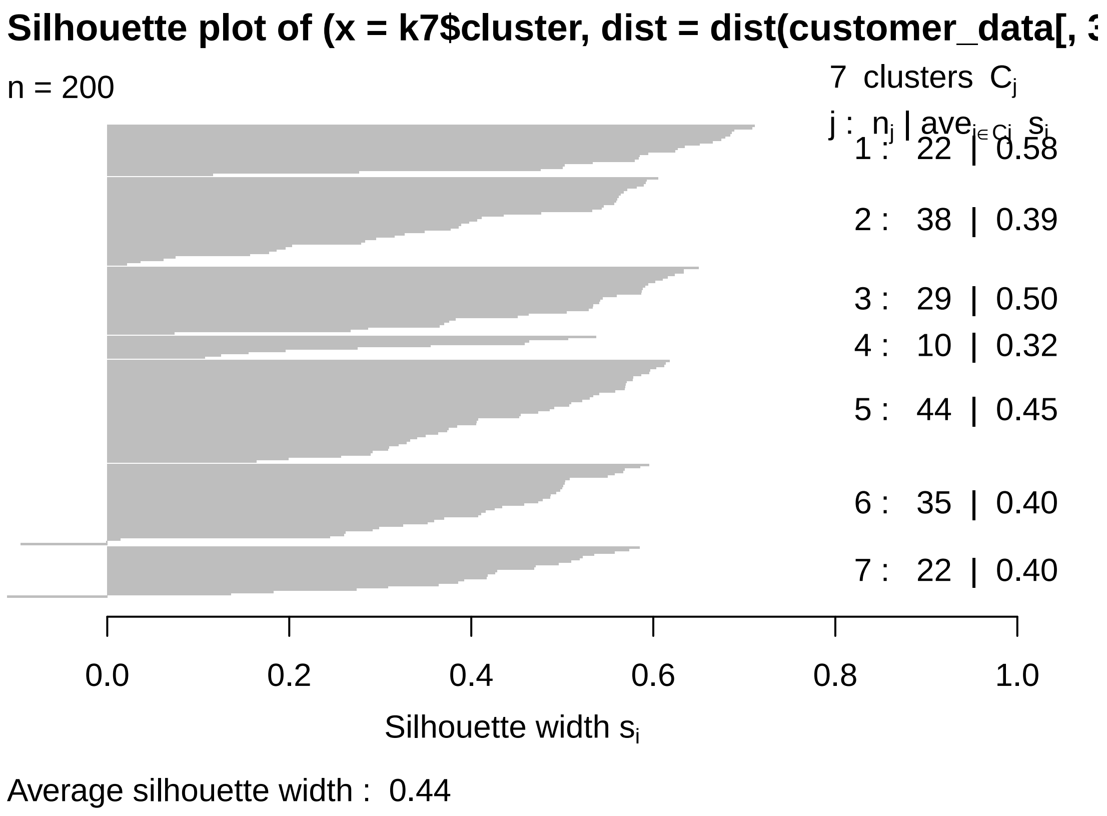
```
# K = 5 clusters
k5<-kmeans(customer_data[,3:5],5,iter.max=100,nstart=50,algorithm="Lloyd")
s5<-plot(silhouette(k5$cluster,dist(customer_data[,3:5],"euclidean")))
```



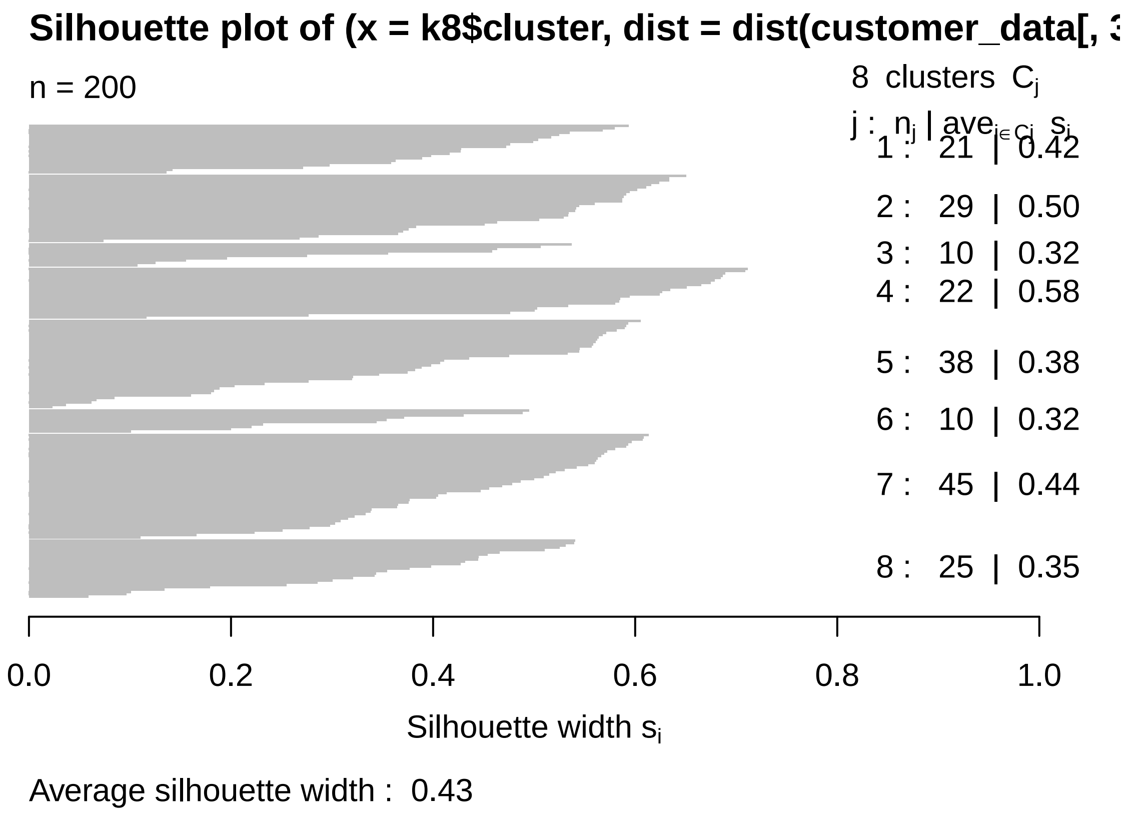
```r
# K = 6 clusters
k6<-kmeans(customer_data[,3:5],6,iter.max=100,nstart=50,algorithm="Lloyd")
s6<-plot(silhouette(k6$cluster,dist(customer_data[,3:5],"euclidean")))
```



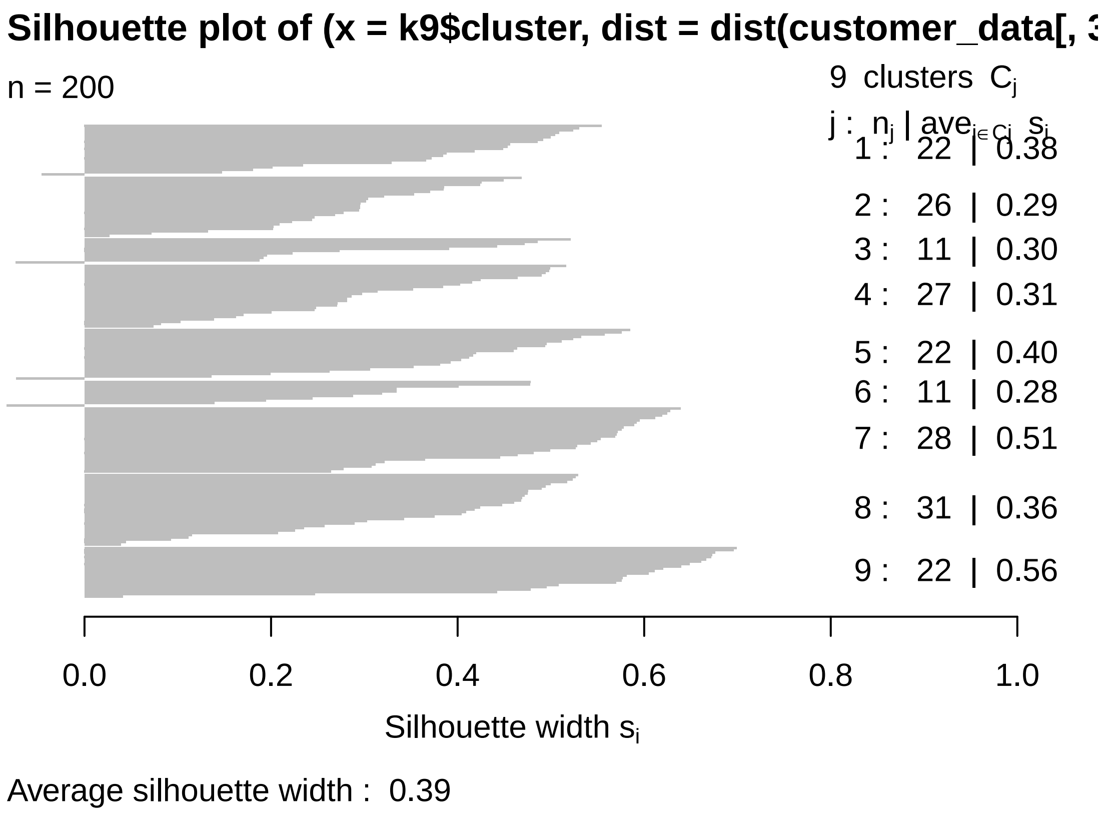
```r
# K = 7 clusters
k7<-kmeans(customer_data[,3:5],7,iter.max=100,nstart=50,algorithm="Lloyd")
s7<-plot(silhouette(k7$cluster,dist(customer_data[,3:5],"euclidean")))
```



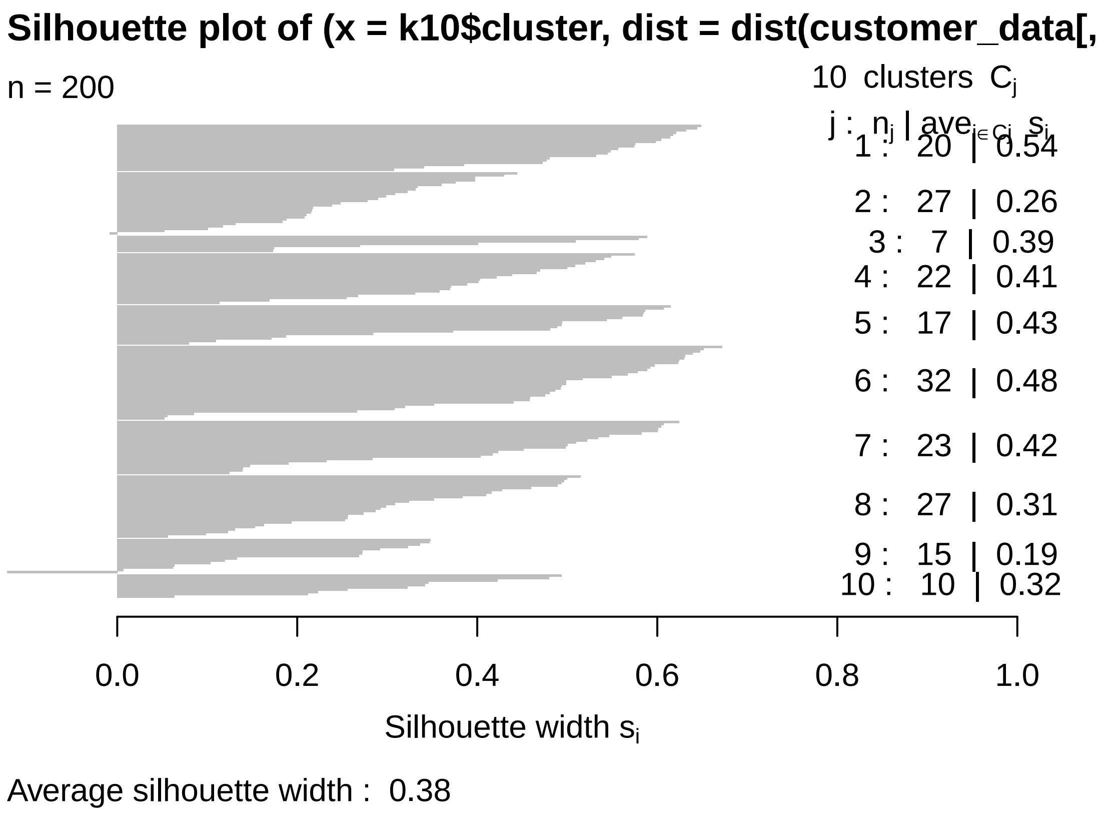
```
# K = 8 clusters
k8<-kmeans(customer_data[,3:5],8,iter.max=100,nstart=50,algorithm="Lloyd")
s8<-plot(silhouette(k8$cluster,dist(customer_data[,3:5],"euclidean")))
```



```
# K = 9 clusters
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")))
```

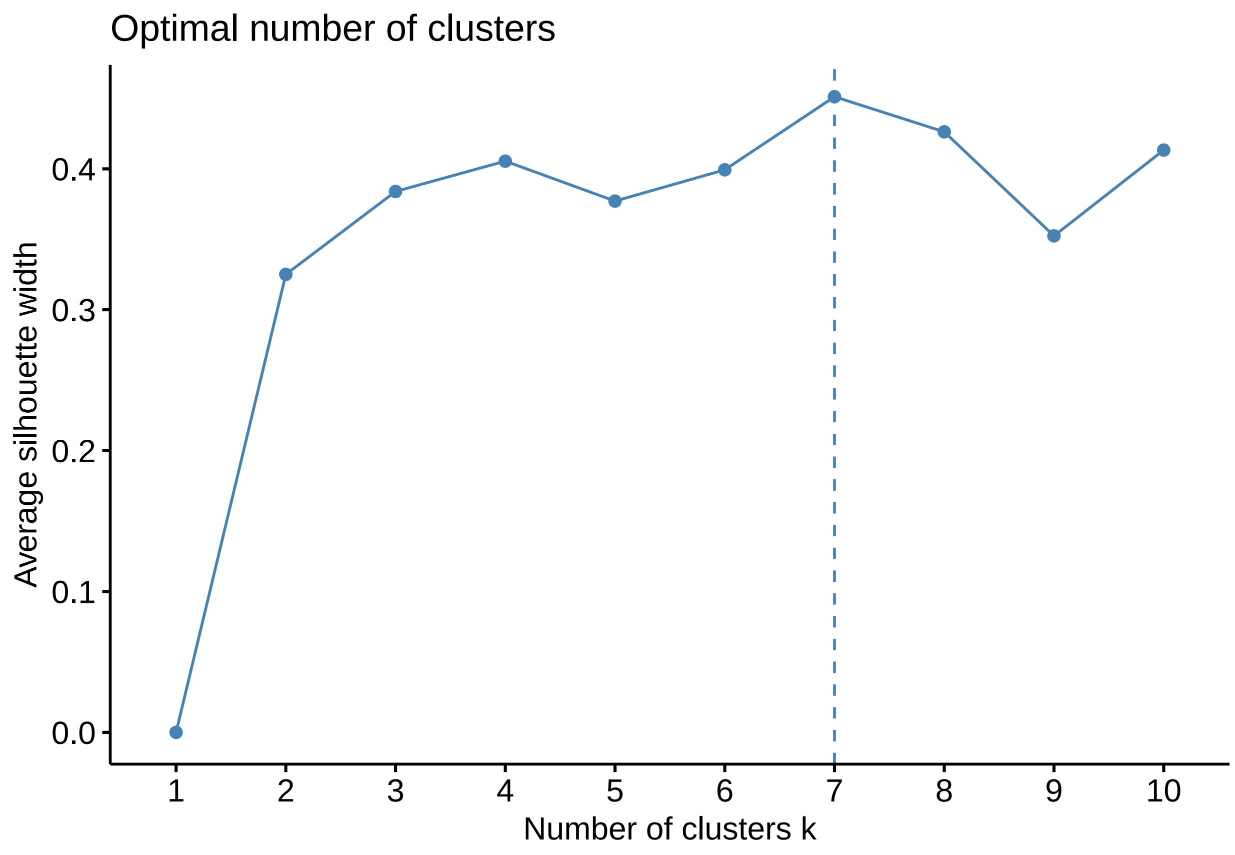


```
# K = 10 clusters
k10<-kmeans(customer_data[,3:5],10,iter.max=100,nstart=50,algorithm="Lloyd")
s10<-plot(silhouette(k10$cluster,dist(customer_data[,3:5],"euclidean")))
```



Maximum average silhouette can be seen for K = 6 which is 0.45. We can make use of the fviz_nbclust() function to determine and visualize the optimal number of clusters.

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



From above plot, we can conclude that K = 7 is the appropriate number of clusters for our customer data. However, maximum average silhouette value can be seen for K = 6 when we computed individually.

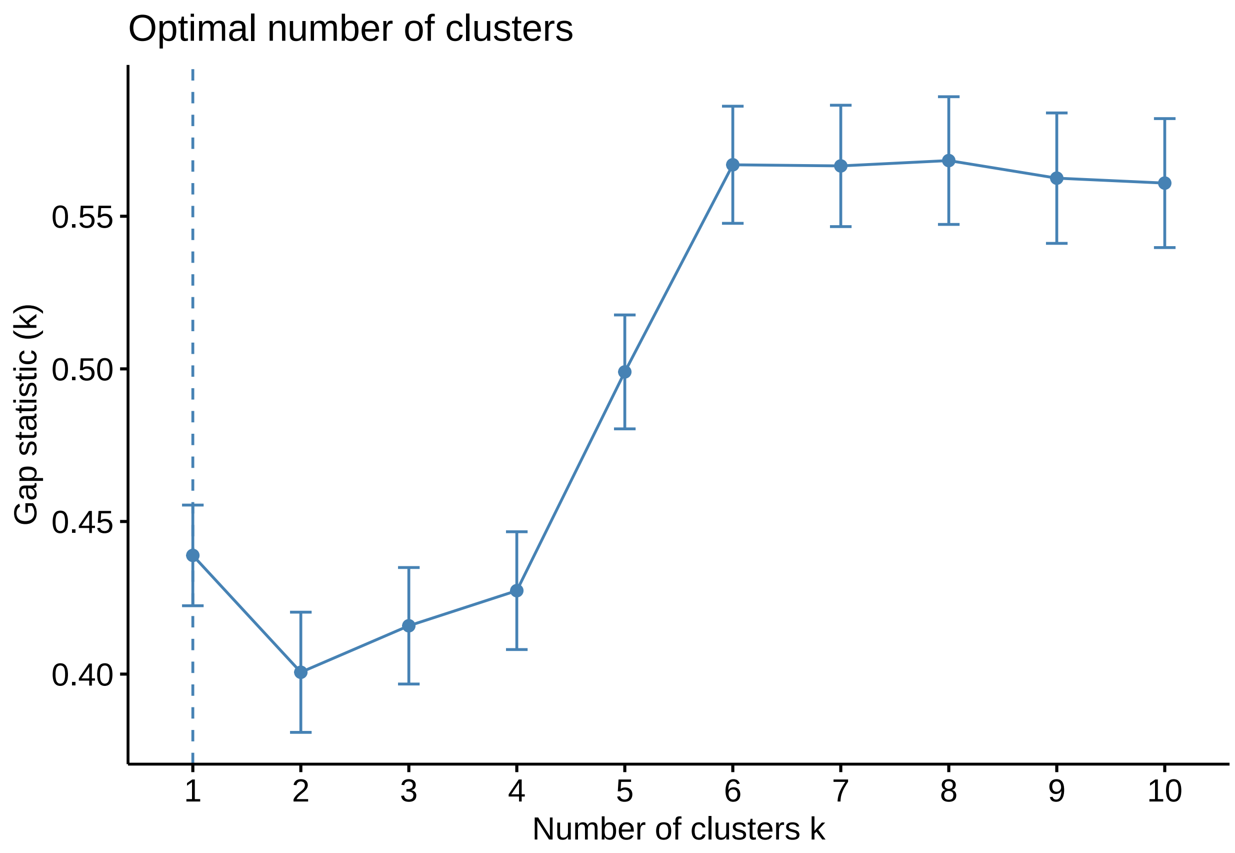
5.1.3 Gap statistic method

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. For computing the gap statistics method we can utilize the clusGap function for providing gap statistic as well as standard error for a given output.

```
set.seed(125, sample.kind = "Rounding")
```

```
## Warning in set.seed(125, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

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



From above plot, we can conclude that K = 6 is optimal number of cluster for our customer data. However, in our previous analysis we concluded that 4 is optimal based on elbow curve and 7 based on Silhouette method. Looking at all together, 6 and 7 have very minor difference and 4 is far away from optimal. Lets us take k = 6 as our optimal cluster.

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

```
## K-means clustering with 6 clusters of sizes 38, 45, 22, 21, 39, 35
##
## Cluster means:
##        Age Annual Income (k$) Spending Score (1-100)
## 1 27.00000           56.65789               49.13158
## 2 56.15556           53.37778               49.08889
## 3 25.27273           25.72727               79.36364
## 4 44.14286           25.14286               19.52381
## 5 32.69231           86.53846               82.12821
## 6 41.68571           88.22857               17.28571
##
## Clustering vector:
##   [1] 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
##  [38] 3 4 3 2 3 2 1 4 3 2 1 1 1 2 1 1 2 2 2 2 2 1 2 2 1 2 2 2 1 2 2 1 1 2 2 2 2
##  [75] 2 1 2 1 1 2 2 1 2 2 1 2 2 1 1 2 2 1 2 1 1 1 2 1 2 1 1 2 2 1 2 1 2 2 2 2 2
## [112] 1 1 1 1 1 2 2 2 2 1 1 1 5 1 5 6 5 6 5 6 5 1 5 6 5 6 5 6 5 6 5 1 5 6 5 6 5
## [149] 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 6
## [186] 5 6 5 6 5 6 5 6 5 6 5 6 5
##
```

```
## Within cluster sum of squares by cluster:
## [1]  7742.895  8062.133  4099.818  7732.381 13972.359 16690.857
##  (between_SS / total_SS =  81.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

From above output of our kmeans 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.
- **totss** – This represents the total sum of squares.
- **centers** – Matrix comprising of several cluster centers
- **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.

5.2 Principle Components Analysis

Lets inspect first Two Principle Components.

```
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
```

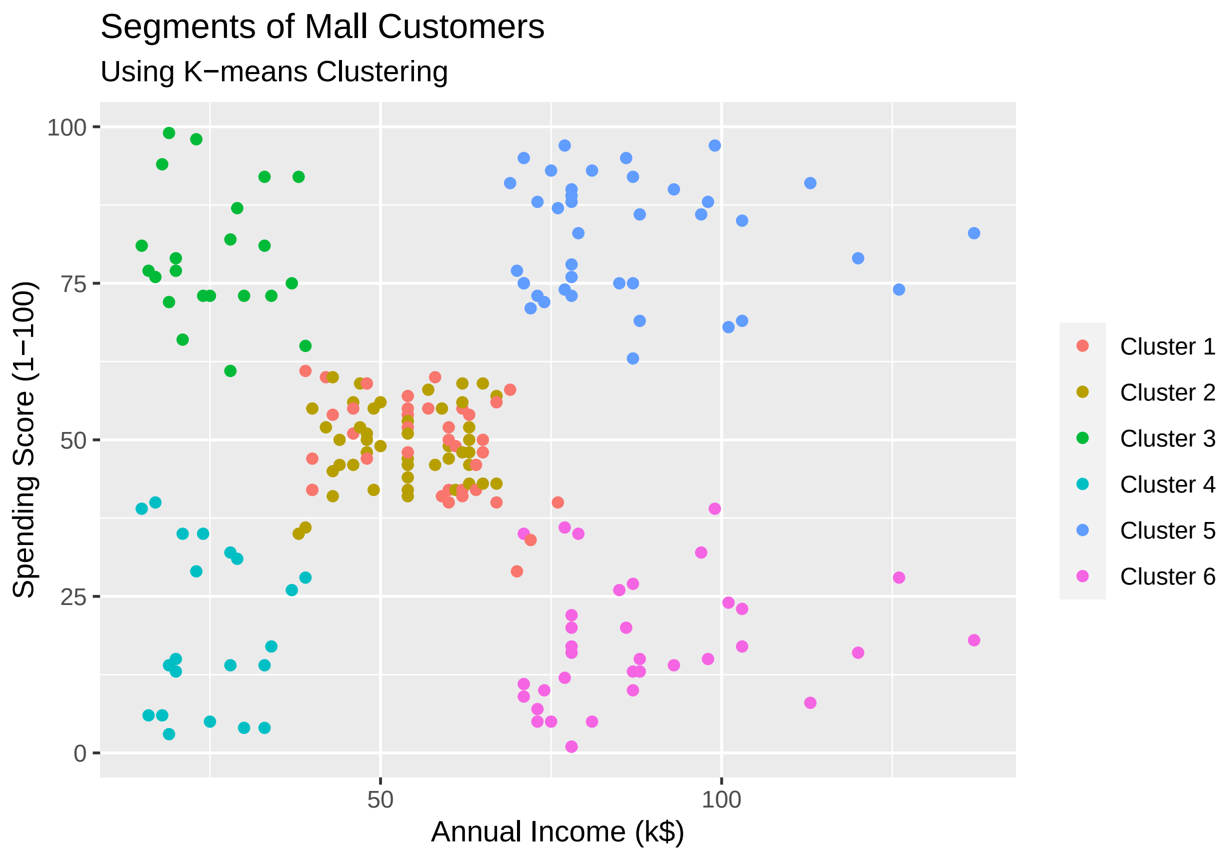
5.3 Visualizing the Clustering Results using the Annual Income, Spending Score, Age, and First Two Principle Components

5.3.1 Visualizing the clustering using Annual income and Spending Score

```
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
# Visualizing the clustering using Annual income and Spending Score
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("Segments of Mall Customers", subtitle = "Using K-means Clustering")
```

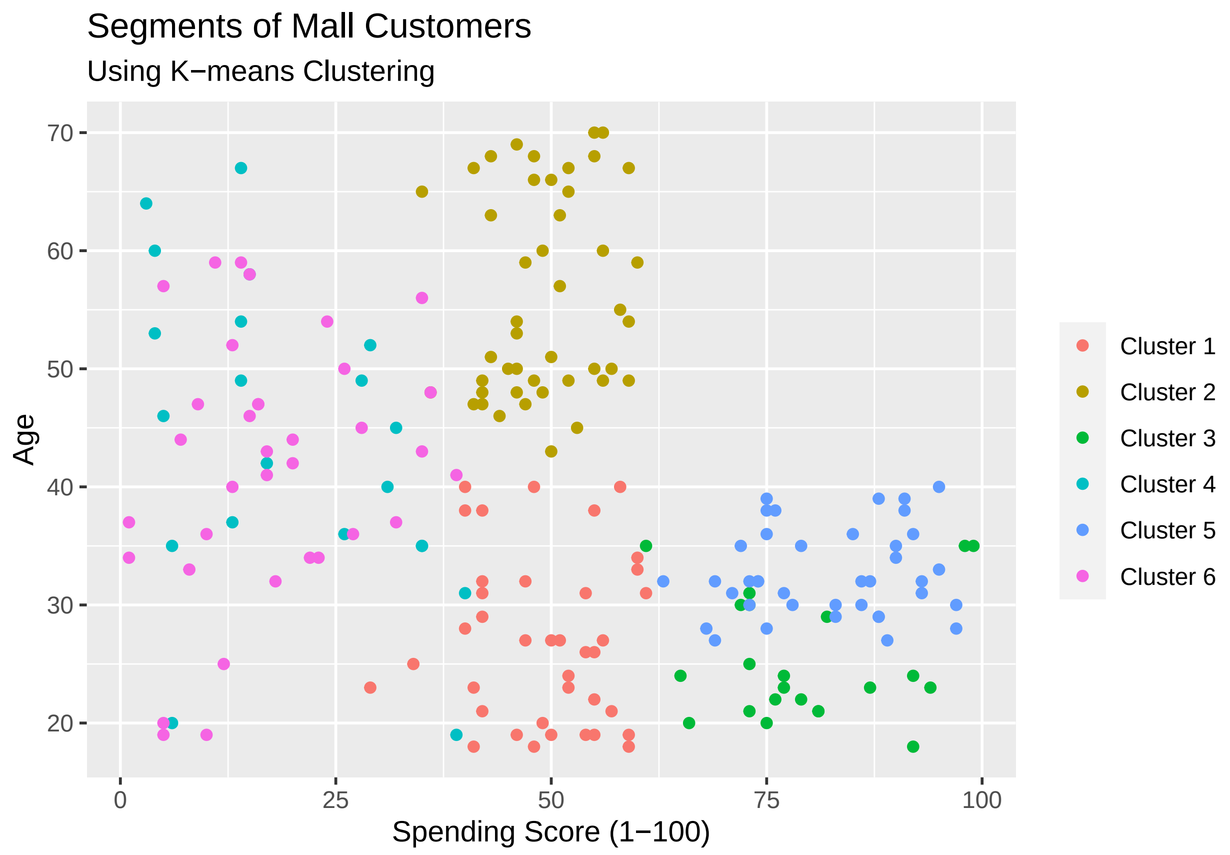


From above scatter plot, we can conclude that -

- **Cluster 1 and 2** - These two cluster consist of customers with medium annual income and medium annual spend of income.
- **Cluster 3** - These are the customers having low annual income but high annual spend of income.
- **Cluster 4** - These are the customers having low annual income and low annual spend of income.
- **Cluster 5** - These are the customers having high annual income and high annual spend of income.
- **Cluster 6** - These are the customers having high annual income but low annual spend of income.

5.3.2 Visualizing the cluster using Spending Score and Age

```
ggplot(customer_data, aes(x = `Spending Score (1-100)`, y = Age)) +
  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("Segments of Mall Customers", subtitle = "Using K-means Clustering")
```



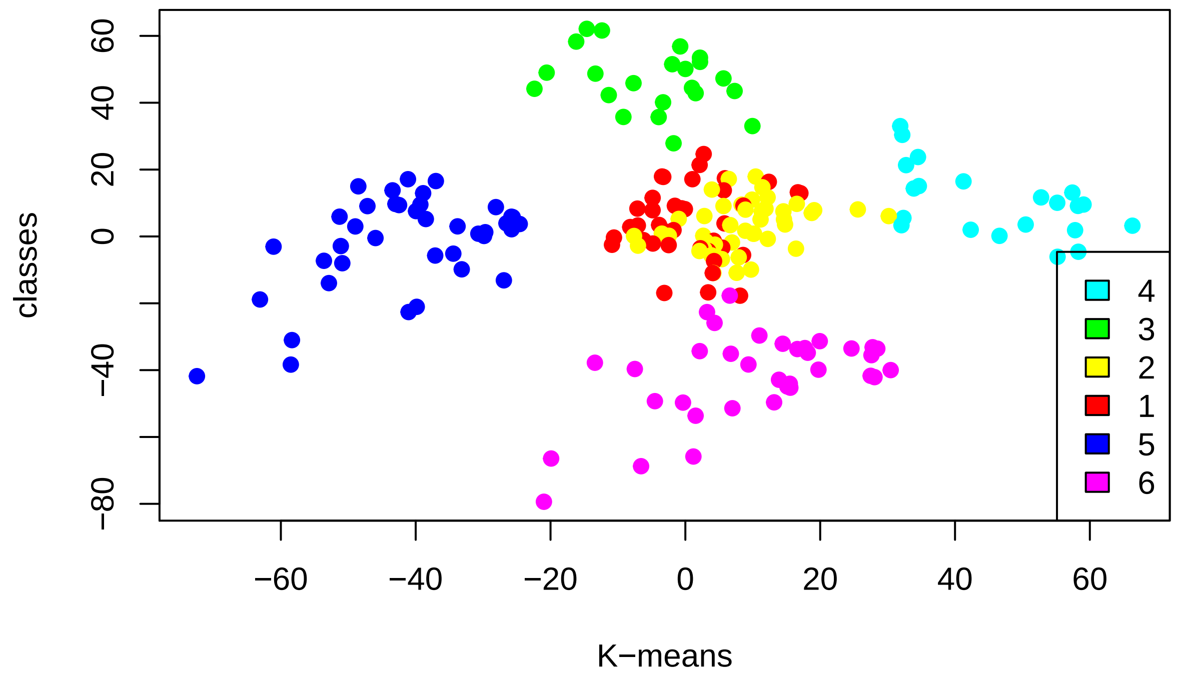
From above scatter plot, we can conclude that -

- **Cluster 1** - These are the customers with lower age group having medium annual spend of income.
- **Cluster 2** - These are the customers with higher age group having medium annual spend of income.
- **Cluster 3** - These are the customers with lower age group having high annual spend of income.
- **Cluster 4** - These are the customers with higher age group having low annual spend of income.
- **Cluster 5** - These are the customers with medium age group but high annual spend of income.
- **Cluster 6** - These are the customers with medium age group but low annual spend of income.

5.3.3 Visualizing customers into 6 cluster based on first two principal component

```
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], col = kCols(digCluster), pch = 19, xlab = "K-means",ylab = "classes")
legend("bottomright",unique(dignm),fill=unique(kCols(digCluster)))
```



Based on above cluster plot, we can conclude that -

- **Cluster 1 and 2** – These two clusters consist of customers with medium PCA1 and medium PCA2 score.
- **Cluster 3** – This comprises of customers with a high PCA2 and a medium PCA1.
- **Cluster 4** – This cluster comprises of customers with a high PCA1 score and a high PCA2.
- **Cluster 5** – This cluster represents customers having a high PCA2 and a low PCA1.
- **Cluster 6** – In this cluster, there are customers with a medium PCA1 and a low PCA2 score.

6 Result

In previous sections, we made use of k-means unsupervised machine learning algorithm with **k-values** between the range of 1 to 10 and applied three well know methods to determine optimized value of **k = 6** which can give us equally distributed cluster. Making a join on **K6$cluster** and **customer_data** gives us complete set of information along with cluster to which customer belong too. Below is a first six rows of mall customer data with cluster to which they are assigned too respectively.

```
customer_data_with_cluster <- as.data.frame(k6$cluster) %>%
  mutate(CustomerID = row_number()) %>%
  left_join(customer_data, by = 'CustomerID')

head(customer_data_with_cluster) %>%
  kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                  position = "center",
                  font_size = 10,
                  full_width = FALSE)
```

k6$cluster	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
4	1	Male	19	15	39
3	2	Male	21	15	81
4	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
3	6	Female	22	17	76

7 Conclusion

In this machine learning project, we went through the customer segmentation 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. Overall, we can conclude that using this method we can identify business customer base into different group so that specific marketing techniques can be applied to increase business revenue.

8 Appendix

8.1 1a - Enviroment

```
print("Operating System:")
```

```
## [1] "Operating System:"
```

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          0.2
## year           2020
## month          06
## day            22
## svn rev        78730
## language       R
## version.string R version 4.0.2 (2020-06-22)
## nickname       Taking Off Again
```

8.2 1b - Github Repository Link

https://github.com/niteshn99/Edx-HarvardX-CustomerSegment

- **data** - This directory contains input data.
- **images** - This directory contains non code generated image used in this report.
- **Edx-CapstoneProject-CustomerSegment.R** - This is R script file containing source code for analysis and model creation.
- **Edx-CapstoneProject-CustomerSegment.Rmd** - This is R markdown report file.
- **Edx-CapstoneProject-CustomerSegment.pdf** - This is report file in PDF format generated using R markdown.