A Project Report

on

HIGH VALUE CUSTOMERS IDENTIFICATION FOR AN E-COMMERCE COMPANY

submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

17WH1A0562	**Ms. BHADAVATH VANDANA**
17WH1A0567	**Ms. MOHANA NAGA S VYSHNAVI MEDISETTY**
17WH1A05B4	**Ms. SHANMITHA PARUCHURI**

by

under the esteemed guidance of

Dr. G. NAGA SATISH, Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

May 2021

DECLARATION

We hereby declare that the work presented in this project entitled "**HIGH VALUE CUSTOMERS IDENTIFICATION FOR AN E-COMMERCE COMPANY**" submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering for Women', Hyderabad is an authentic record of our original work carried out under the guidance of Dr. G. Naga Satish, Professor, Department of CSE.

[illegible]
28/5/21

Sign. with date:

Ms. BHADAVATH VANDANA

(17WH1A0562)

[illegible]
28/5/21

Sign. with date:

Ms. MOHANA NAGA S VYSHNAVI MEDISETTY

(17WH1A0567)

[illegible]
28/5/21

Sign. with date:

Ms. SHANMITHA PARUCHURI

(17WH1A05B4)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**HIGH VALUE CUSTOMERS IDENTIFICATION FOR AN E-COMMERCE COMPANY**" is a bonafide work carried out by Ms. BHADAVATH VANDANA (17WH1A0562); Ms. MOHANA NAGA S VYSHNAVI MEDISETTY (17WH1A0567) ; Ms. SHANMITHA PARUCHURI (17WH1A05B4) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering,BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the reward of any degree or diploma

Head of the Department	**Guide**
Dr. K Srinivas Reddy	**Dr. Ganti Naga Satish**
Professor & HoD,	**Professor**
Professor	**Department of CSE**
Department of CSE	

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. K. Srinivas Reddy, Hod**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project. Professor

We are extremely thankful and indebted to our internal guide, **Dr. G. Naga Satish, Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. BHADAVATH VANDANA
(17WH1A0562)

Ms. MOHANA NAGA S VYSHNAVI MEDISETTY
(17WH1A0567)

Ms. SHANMITHA PARUCHURI
(17WH1A05B4)

CONTENTS

ABSTRACT

The emergence of many competitors and entrepreneurs has caused a lot of tension among competing businesses to find new buyers and keep the old ones. As a result of the predecessor, the need for exceptional customer service becomes appropriate regardless of the size of the business. Furthermore, the ability of any business to understand the needs of each of its customers will provide greater customer support in providing targeted customer services and developing customized customer service plans. This understanding is possible through *structured customer service.* Each segment has customers who share the same market features. Big data ideas and machine learning have promoted greater acceptance of automated customer segmentation approaches. Clustering algorithms like k-means and Hierarchical clustering can be used for this purpose. The goal is to better understand customers' spending and ordering habits with the following features: Number of products ordered, average return rate and total spending. In this project, we aimed at identifying the right number of customer segments, providing the number of customers who are highly valued and identifying the clustering algorithm that gives maximum accuracy.

LIST OF FIGURES

1. INTRODUCTION

As a business grows in size, it will not be possible for the business to have an intuition about each and every customer. At such a stage, human judgments about which customers to pursue will not work and the business will have to use a data-driven approach to build a proper strategy.

For a medium to large size retail store, it is also imperative that they invest not only in acquiring new customers but also in customer retention. Many businesses get most of their revenue from their ‘best’ or high-valued customers. Since the resources that a company has are limited, it is crucial to find these customers and target them. It is equally important to find the customers who are dormant/are at high risk of churning to address their concerns. For this purpose, companies use the technique of customer segmentation.

1.1 Objectives

A UK-based online retail store has captured the sales data for different products for the period of one year (Nov 2016 to Dec 2017). The organization sells gifts primarily on the online platform. The customers who make a purchase directly for themselves. There are small businesses that buy in bulk and sell to other customers through the retail outlet channel. The organization wants to roll out a loyalty program to the high-value customers after identification of segments. The main goal is to identify the right number of customer segments, provide the number of customers who are highly valued and identify the clustering algorithm that gives maximum accuracy.

1.2 Methodology

To use the clustering methodologies such as ***K-means*** and ***Hierarchical algorithms*** to segment customers into Retail and wholesale(high and low valued) groups and find significant customers for the business who make high purchases of their favorite products.

1.3 Dataset

A transnational dataset that contains all the transactions occurring between Nov-2016 to Dec-2017 for a UK-based online retail store.

Domain : E-commerce

Attribute	**Description**
Invoice No	Invoice number
Stock Code	Product (item) code
Description	Product (item) name
Quantity	The quantities of each product (item) per transaction
Invoice Date	the day when each transaction was generated
Unit Price	Unit price (Product price per unit)
Customer ID	Customer number (Unique ID assigned to each customer)
Country	Country name (The name of the country where each customer resides)

1.4 System Requirements

Environment	Specifications
Hardware	128GB SSD 8GB RAM Intel I5 Core Processor
Software	Software Specifications- RStudio Utilities R (v4.0+) and packages Ubuntu 18.04

2. CUSTOMER SEGMENTATION ANALYSIS

2.1 Brief Introduction

Customer segmentation is the process of separating customers into groups on the basis of their shared behavior or other attributes. The groups should be homogeneous within themselves and should also be heterogeneous to each other. The overall aim of this process is to identify high-value customer base i.e., customers that have the highest growth potential or are the most profitable.

Insights from customer segmentation are used to develop tailor-made marketing campaigns and for designing overall marketing strategy and planning.

A key consideration for a company would be whether or not to segment its customers and how to do the process of segmentation. This would depend upon the company philosophy and the type of product or services it offers. The type of segmentation criterion followed would create a big difference in the way the business operates and formulates its strategy. This is elucidated below.

1. Zero segments: <*Undifferentiated approach*> This means that the company is treating all of its customers in a similar manner. In other words, there is no differentiated strategy, and all of the customer base is being reached out by a single mass marketing campaign.
2. One segment: <*Focused approach*> This means that the company is targeting a particular group or niche of customers in a tightly defined target market.
3. Two or more segments: <*Differentiated approach*> This means that the company is targeting 2 or more groups within its customer base and is making specific marketing strategies for each segment.
4. Thousands of segments: <*Hyper segmentation approach*> This means that the company is treating each customer as unique and is making a customized offer for each one of them.

Once the company has identified its customer base and the number of segments it aims to focus upon, it needs to decide the factors on whose basis it will decide to segment its customers.

2.2 The importance of Customer Segmentation

Customer segmentation has a lot of potential benefits. It helps a company to develop an effective strategy for targeting its customers. This has a direct impact on the entire product development cycle, the budget management practices, and the plan for delivering targeted promotional content to customers. For example, a company can make a high-end product, a budget product, or a cheap alternative product, depending upon whether the product is intended for its most high yield customers, frequent purchasers or for the low-value customer segment. It may also fine-tune the features of the product for fulfilling the specific needs of its customers.

Customer segmentation can also help a company to understand how its customers are alike, what is important to them, and what is not. Often such information can be used to develop personalized relevant content for different customer bases. Many studies have found that customers appreciate such individual attention and are more likely to respond and buy the product. They also come to respect the brand and feel connected with it. This is likely to give the company a big advantage over its competitors. In a world where everyone has hundreds of emails, push notifications, messages, and ads dropping into their content stream, no one has time for irrelevant content.

Finally, this technique can also be used by companies to test the pricing of their different products, improve customer service, and upsell and cross-sell other products or services.

2.3 Customer Segmentation Strategy

To start with customer segmentation, it is important to have a clear vision and a goal in mind. The following steps can be undertaken to find segments in the customer base on a broad level.

1. ***Analyze the existing customer pool:*** Understanding the geographical distribution, customer preferences/beliefs, reviewing website search page analytics, etc.

2. ***Develop an understanding of each customer**:* Mapping each customer to a set of preferences to understand and predict their behavior: the products, services, and content they would be interested in.
3. ***Define segment opportunities:*** Once the segments have been defined, there should be a proper business understanding of each segment and its challenges and opportunities. The entire company's marketing strategy can be branched out to cater to different niches of customers.
4. ***Research the segment:*** After cementing the definition and business relevance of different customer segments, a company needs to understand how to modify its products or services to better cater to them. For example, it may decide to provide higher discounts to some customers compared to others to expand its active customer base.
5. ***Tweak strategy:*** Experiment with new strategies and understand the impact of time and economy on the purchasing behavior of customers in different segments. And then the process should be repeated for refining the strategy as much as possible.

2.4 Cluster Analysis & Distance Measures

Cluster analysis is used in a variety of applications. For example, it can be used to identify consumer segments, or competitive sets of products, or groups of assets whose prices co-move, or for geo-demographic segmentation, etc. In general, it is often necessary to split our data into segments and perform any subsequent analysis within each segment in order to develop (potentially more refined) segment-specific insights. This may be the case even if there are not intuitively "natural" segments in our data.

To identify segments in the data one can, use statistical techniques broadly called Clustering techniques. Based on how we define "***similarities***" and "***differences***" between data observations, which can also be defined mathematically using distance metrics, one can find different segmentation solutions. A key ingredient of clustering and segmentation is exactly the definition of these distance metrics (between observations), which need to be defined creatively based on contextual knowledge and not only using "black box" mathematical equations and techniques.

The classification of observations into groups requires some methods for computing the distance or the (dis)similarity between each pair of observations. The result of this computation is known as a dissimilarity or distance matrix. There are many methods to calculate this distance information; the choice of distance measures is a critical step in clustering. It defines how the similarity of two elements (x, y) is calculated and it will influence the shape of the clusters.

The choice of distance measures is a critical step in clustering. It defines how the similarity of two elements (x, y) is calculated and it will influence the shape of the clusters. The classical methods for distance measures are ***Euclidean* and *Manhattan distances***, which are defined as follow:

Euclidean distance:

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

Manhattan distance:

$$d_{man}(x,y) = \sum_{i=1}^{n}|(x_i - y_i)|$$

Where, *x* and *y* are two vectors of length *n*.

Other dissimilarity measures exist such as correlation-based distances, which is widely used for gene expression data analyses.

3. DATA WRANGLING AND CLEANING

3.1 Importing the Ecommerce dataset

Before we import the dataset, it is a good practice that we set the working directory first.
The **working directory** is just a file path on your computer that sets the default location of any files you read into R, or save out of R.

```
> # Change my working directory to the following path
> setwd("/home/mohana/Desktop/HighValueCustomerIdentification")

> # Change my working directory to the following path
> getwd()
[1] "/home/mohana/Desktop/HighValueCustomerIdentification"
```

Importing data into R is a necessary step that, at times, can become time intensive. To ease this task, RStudio includes new features to import data from: csv, xls, xlsx, sav, dta, por, sas and stata files.

The data import features can be accessed from the environment pane or from the tool's menu. Theimporters are grouped into 3 categories: Text data, Excel data and statistical data. To access this feature, use the "Import Dataset" dropdown from the "Environment" pane or through the "File" menu, followed by the "Import Dataset" submenu.

Importing data from Text and CSV files

Importing "From Text (readr)" files allows you to import CSV files and in general, character delimited files using the readr package. This Text importer provides support to:

- Import from the file system or a url
- Change column data types
- Skip or include-only columns
- Rename the data set
- Skip the first N rows

- Use the header row for column names
- Trim spaces in names
- Change the column delimiter
- Encoding selection
- Select quote, escape, comment and NA identifiers

For example, one can import with ease a csv file Ecommerce.csv provided that the file is in the current working directory. If not, one must specify the path as well.

```
# Ecommerce dataset exists in the current working directory or else specify the path
data = read.csv("Ecommerce.csv")

# Top 6 rows of dataset
head(data)
```

Invoice No	StockCode	Description	Quantity	InvoiceDate	Unit Price	Customer ID	Country
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	29-Nov-16	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	29-Nov-16	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	29-Nov-16	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	29-Nov-16	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	29-Nov-16	3.39	17850	United Kingdom

3.2 Structure of dataset

str() function in R language is used for compactly displaying the internal structure of a R object. It can display even the internal structure of large lists which are nested. It provides one liner output for the basic R objects letting the user know about the object and its constituents. It can be used as an alternative to summary() but str() is more compact than summary(). It gives information about the rows(observations) and columns(variables) along with additional information like the names of the columns, class of each column followed by few of the initial observations of each of the columns.

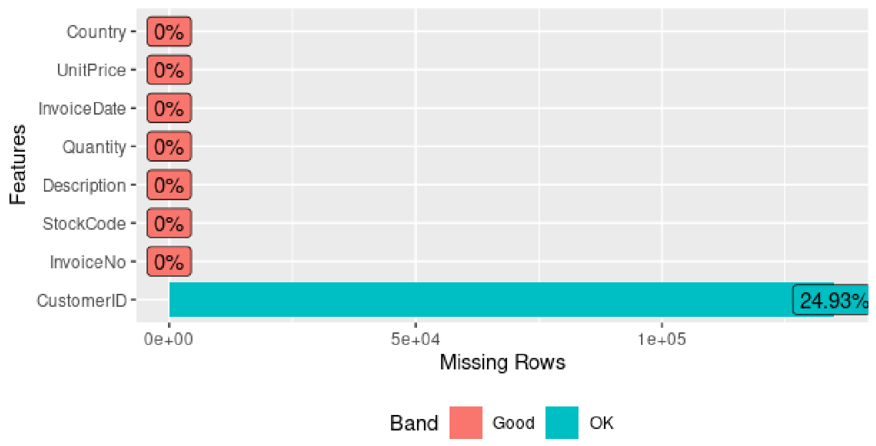
```
> # Structure of the data
> str(data)
'data.frame':	541909 obs. of  8 variables:
 $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
 $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
 $ Description: chr "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUPID HEARTS COAT HANGER" "KNITTED UNION FLAG HOT WATER BOTTLE" ...
 $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
 $ InvoiceDate: chr  "29-Nov-16" "29-Nov-16" "29-Nov-16" "29-Nov-16" ...
 $ UnitPrice : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
 $ CustomerID : int  17850 17850 17850 17850 17850 17850 17850 17850 17850 13047 ...
 $ Country    : chr "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" ...
```

3.3 Dealing with missing values

In R the missing values are coded by the symbol NA . To identify missing in your dataset the function is is.na() . When you import a dataset from other statistical applications the missing values might be coded with a number, for example 99 . In order to let R, know that is a missing value you need to recode it.

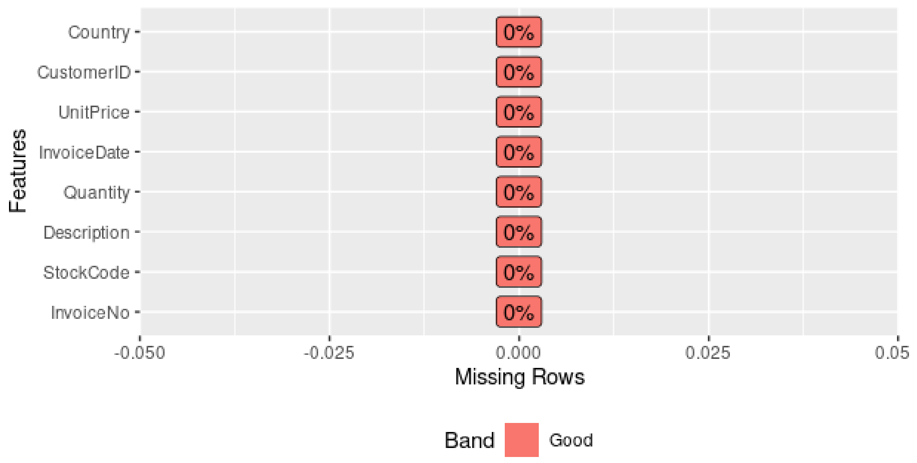
First, we perform some descriptive statistics (how many are missing? how many variables are missing?) and try to inspect and visualize the pattern of missing entries and get hints on the mechanism.

```
# look for missing values using the DataExplorer package
options(repr.plot.width=8, repr.plot.height=3)
plot_missing(data)
```



The na.omit() method from the dplyr library is a simple way to exclude missing observations. Dropping all the NA from the data is easy but it does not mean it is the most elegant solution.

```
# Omitting missing values in customerData
data <- na.omit(data)
plot_missing(data)
```



We could also impute(populate) missing values with the median or the mean. A good practice is to create two separate variables for the mean and the median. Once created, we can replace the missing values with the newly formed variables.

3.4 Data Transformation

Date conversion in R can be a real pain. However, it is a very important initial step when you first get your data into R to ensure that it has the correct type (e.g., Date). This gives you the correct functionality for working with data of that type.

R provides a number of handy features for working with date-time data. However, the sheer number of options/packages available can make things seem overwhelming at first. There are more than 10 packages providing support for working with date-time data in R, as well as being able to use the as. Date( ) function to convert character data to dates. In this post, I will provide an introduction to the functionality R offers for converting strings to dates. Converting dates enteredas strings into numeric dates in R is relatively simple for a single string and for a vector of strings if you have consistent date information. It is far trickier if the date information is represented inconsistently.

```
# Convert the date variable to the appropriate class
data$InvoiceDate <- as.Date(data$InvoiceDate, "%d-%B-%y")
```

We'll just add one more column at this stage, we'll calculate the line total by multiplying the Quantity by the UnitPrice for each line.

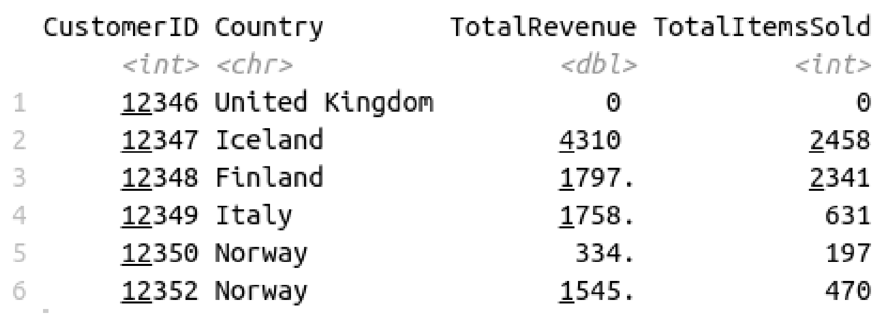
```
# Computing the line total
data <- data %>% mutate(LineTotal = Quantity * UnitPrice)
head(data)
```

```
  InvoiceNo StockCode                         Description Quantity InvoiceDate UnitPrice CustomerID
1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6   29-Nov-16      2.55      17850
2    536365     71053                 WHITE METAL LANTERN        6   29-Nov-16      3.39      17850
3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8   29-Nov-16      2.75      17850
4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6   29-Nov-16      3.39      17850
5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6   29-Nov-16      3.39      17850
6    536365     22752        SET 7 BABUSHKA NESTING BOXES        2   29-Nov-16      7.65      17850
         Country LineTotal
1 United Kingdom     15.30
2 United Kingdom     20.34
3 United Kingdom     22.00
4 United Kingdom     20.34
5 United Kingdom     20.34
6 United Kingdom     15.30
```

When it comes to marketing, being able to segment your customers so that you can market appropriate products and offers to them is a quick win. In this dataset, we have the customer ID, which we can use to start looking for differences between customers. While we have product IDs and descriptions, we are lacking in a simple product category ID. There is information we can extract from other fields, but that may be beyond the scope of this kernel, which is already getting a little long!

Let's look at our customer data from the point of view of loyalty. If we can identify a group of customers who shop regularly, we can target this group with dedicated marketing campaigns which may reinforce their loyalty, and perhaps lead to them becoming brand ambassadors on social media.

```
# Total revenue generated and Total Items purchased by each customer
customerData <- data %>%
  group_by(CustomerID, Country) %>%
  summarise(TotalRevenue = sum(LineTotal), TotalItemsSold = sum(Quantity))
head(customerData)
```



```
  CustomerID Country        TotalRevenue TotalItemsSold
       <int> <chr>                 <dbl>          <int>
1      12346 United Kingdom           0              0
2      12347 Iceland               4310           2458
3      12348 Finland               1797.          2341
4      12349 Italy                 1758.           631
5      12350 Norway                 334.           197
6      12352 Norway               1545.           470
```

Few customers are residents of two countries in the same year. Such data contributes to very less percentage of the dataset and can be removed to ensure correct functionality for working with data.

```
length(unique(customerData$CustomerID))
n_occur <- data.frame(table(customerData$CustomerID))
dualResidentsIds = (n_occur[n_occur$Freq > 1,])$Var1
customerData <- subset(customerData, !(customerData$CustomerID %in% dualResidentsIds))
remove(n_occur)
remove(dualResidentsIds)
```

Looking at outliers in R

Outliers can be dangerous for your data science activities because most statistical parameters such as mean, standard deviation and correlation are highly sensitive to outliers. Consequently, any statistical calculation based on these parameters is affected by the presence of outliers.

Whether it is good or bad to remove outliers from your dataset depends on whether they affect your model positively or negatively. Remember that outliers aren't always the result of badly recorded observations or poorly conducted experiments. They may also occur due to natural fluctuations in the experiment and might even represent an important finding of the experiment.

Whether you're going to drop or keep the outliers requires some amount of investigation. However, it is not recommended to drop an observation simply because it appears to be an outlier.

Statisticians have devised several ways to locate the outliers in a dataset. The most common methods include the Z-score method and the Interquartile Range (IQR) method. However, I prefer the IQR method because it does not depend on the mean and standard deviation of a dataset and I'll be going over this method throughout the tutorial.

The interquartile range is the central 50% or the area between the 75th and the 25th percentile of a distribution. A point is an outlier if it is above the 75th or below the 25th percentile by a factor of 1.5 times the IQR.

For example, if

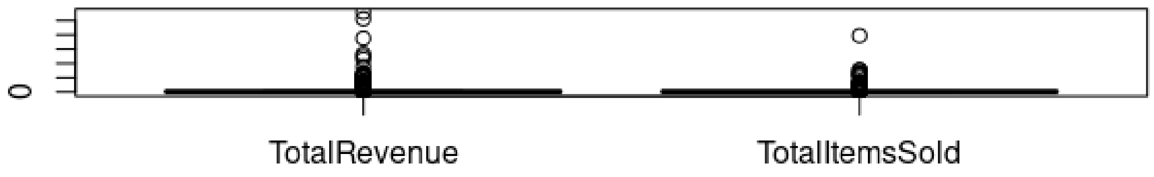
Q1= 25th percentile

Q3= 75th percentile

Then, IQR= Q3 – Q1

An outlier would be a point below [Q1- (1.5)IQR] or above [Q3+(1.5)IQR].

```
# Removing CustomerId and Country Columns as most of the clustering algorithms accepts
#only continuous variables
customerData <- customerData[-c(1:2)]

# Visualizing outliers
boxplot(customerData)$out
```



Eliminating outliers

We used the quantile() function to find the 25th and the 75th percentile of the dataset, and the IQR() function which elegantly gives me the difference of the 75th and 25th percentiles.

It may be noted here that the quantile() function only takes in numerical vectors as inputs whereas warpbreaks is a data frame. The IQR function also requires numerical vectors and therefore arguments are passed in the same way.4

Using the subset() function, you can simply extract the part of your dataset between the upper and lower ranges leaving out the outliers.

```
# ELIMINATING OUTLIERS IN 'customerData'

# IQR and the Quantiles
iqr <- IQR(customerData$TotalRevenue)
Q <- quantile(customerData$TotalRevenue, probs=c(.25, .75), na.rm = FALSE)

# Extract the part of dataset between the upper and lower ranges leaving out the outliers
eliminated <- subset(customerData, customerData$TotalRevenue > (Q[1]  -  1.5*iqr)  &
customerData$TotalRevenue < (Q[2]+1.5*iqr))

customerData <- eliminated
```

The standard k-means algorithm isn't directly applicable to categorical data, for various reasons. The sample space for categorical data is discrete and doesn't have a natural origin. A Euclidean distance function on such a space isn't really meaningful.

Categorical data is a problem for most algorithms in machine learning. Suppose in ‘customer Data’, we have some categorical variable called "Country" that could take on the values United Kingdom, Australia, France and so on. If we simply encode these numerically as 1,2, and 3 respectively, our algorithm will think that the United Kingdom (1) is actually closer to Australia (2) than it is to France (3) which doesn't make any actual sense. We need to use a representation that lets the computer understand that these things are all actually equally different.

One simple way is to use what's called a one-hot representation. Rather than having one variable like "Country" that can take on three or more values, we separate it into different variables. These would be "Country-United Kingdom", "Country-Australia", "Country-France" and so on which all can only take on the value 1 or 0. This increases the dimensionality of the space, but now you could use any clustering algorithm you like. It does sometimes make sense to z-score or whiten the data after doing this process.

```
> length(unique(data$Country))
[1] 37
```

In ‘customerData’, there are 37 unique countries in the ‘Country’ column. If we apply one-hot encoding, 37 columns will be additionally created. While you can use **PCA** on binary data (e.g. **one-hot encoded** data) that does not mean it is a good thing, or it will work very well. **PCA** is designed for continuous variables. For this reason, ‘Country’ column is omitted.

Also, one might also disregard the ‘CustomerID’ column in ‘customerData’ as ids are independent and identically distributed.

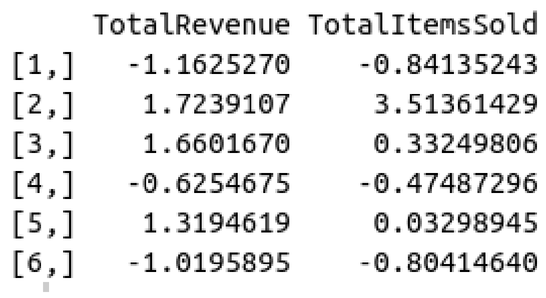
```
customerData <- customerData[-c(1:2)]    ##Removing CustomerId and Country Columns
```

Scaling the data

In statistics, Standardization (sometimes called data normalization or feature scaling) refers to theprocess of rescaling the values of the variables in a data set, so they share a common scale. Often performed as a pre-processing step, particularly for cluster analysis, standardization may be important in case of working with data where each variable has a different unit or where thescales of each of the variables are very different from one another (e.g., 0-1 vs 0-1000). The reason this importance is particularly high in cluster analysis is because groups are defined based on the distance between points in mathematical space.

When working with data where each variable means something different, (e.g., age and weight) the fields are not directly comparable. One year is not equivalent to one pound and may or may not have the same level of importance in sorting a group of records. In a situation where one field has a much greater range of value than another (because the field with the wider range of values likely has greater distances between values), it may end up being the primary driver of what defines clusters. Standardization helps to make the relative weight of each variable equal by converting each variable to a unitless measure or relative distance.

```
# Scaling the data before applying the clustering algorithms
customerData <- scale(customerData)
head(customerData)
```

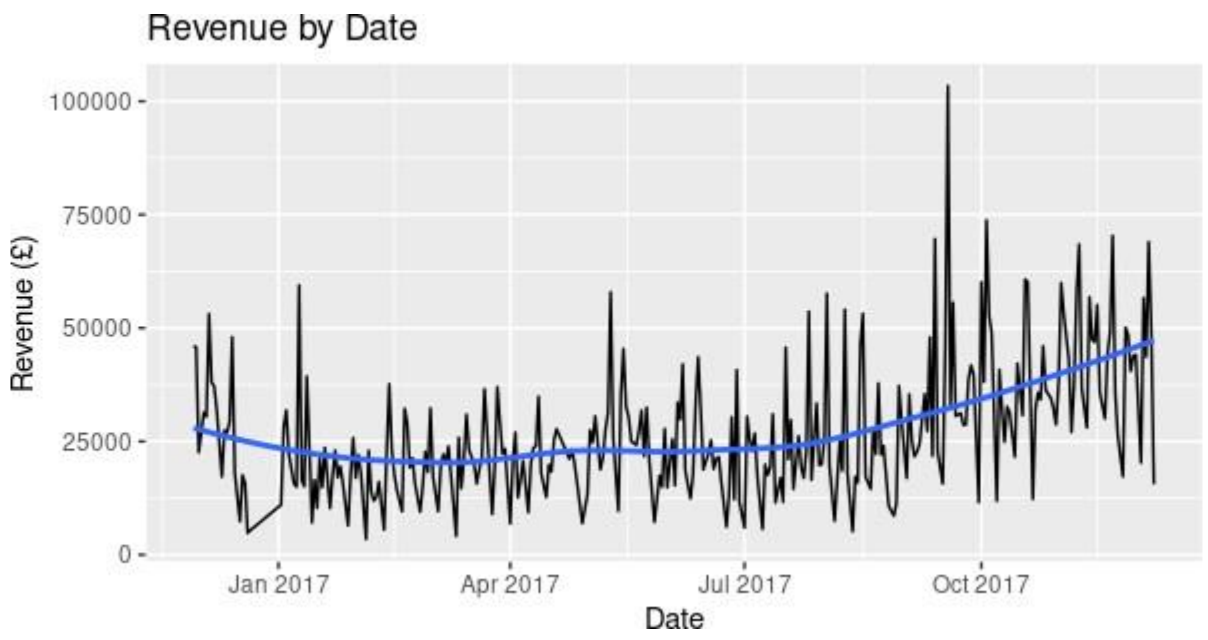


```
     TotalRevenue TotalItemsSold
[1,]   -1.1625270    -0.84135243
[2,]    1.7239107     3.51361429
[3,]    1.6601670     0.33249806
[4,]   -0.6254675    -0.47487296
[5,]    1.3194619     0.03298945
[6,]   -1.0195895    -0.80414640
```

4. EXPLORATORY DATA ANALYSIS

Now that the Data Cleaning and Wrangling is done, we can take a look at some of the 'big picture' aspects of the dataset.

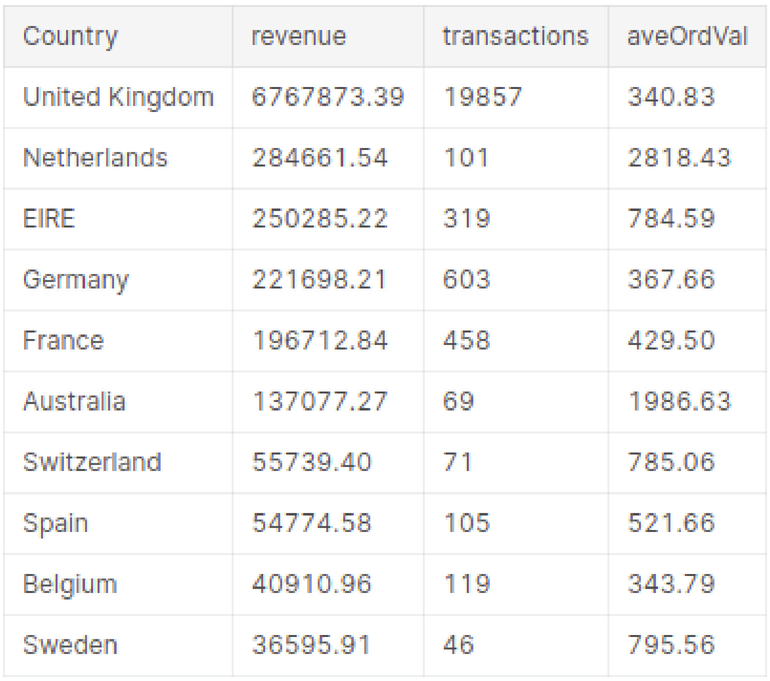
```
# Day Analysis
options(repr.plot.width=8, repr.plot.height=3)
data %>%
  group_by(InvoiceDate) %>%
  summarise(revenue = sum(LineTotal)) %>%
   ggplot(aes(x = InvoiceDate, y = revenue)) + geom_line() + geom_smooth(method = 'auto', se
= FALSE) + labs(x = 'Date', y = 'Revenue (£)', title = 'Revenue by Date')
```



It appears as though sales are trending up, so that's a good sign.

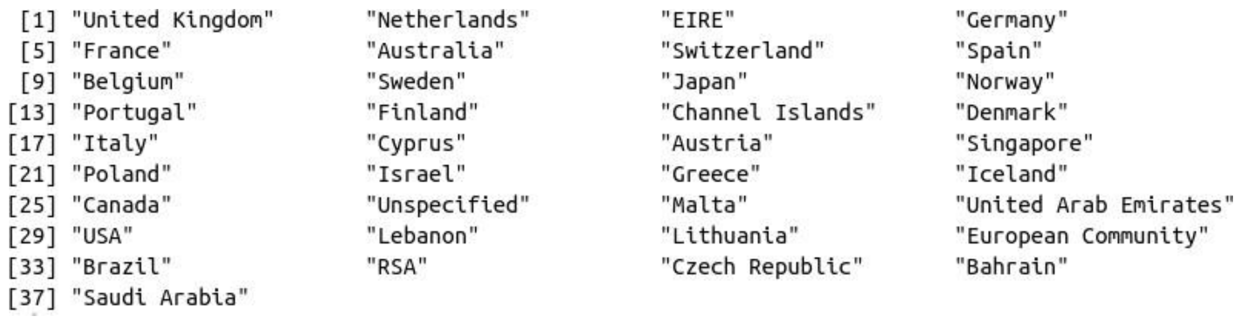
```
# Country Summary
countrySummary <- data %>%
group_by(Country) %>%
  summarise(revenue = sum(LineTotal), transactions = n_distinct(InvoiceNo)) %>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup() %>%
  arrange(desc(revenue))
```

```
head(countrySummary, n = 10)
```



Country	revenue	transactions	aveOrdVal
United Kingdom	6767873.39	19857	340.83
Netherlands	284661.54	101	2818.43
EIRE	250285.22	319	784.59
Germany	221698.21	603	367.66
France	196712.84	458	429.50
Australia	137077.27	69	1986.63
Switzerland	55739.40	71	785.06
Spain	54774.58	105	521.66
Belgium	40910.96	119	343.79
Sweden	36595.91	46	795.56

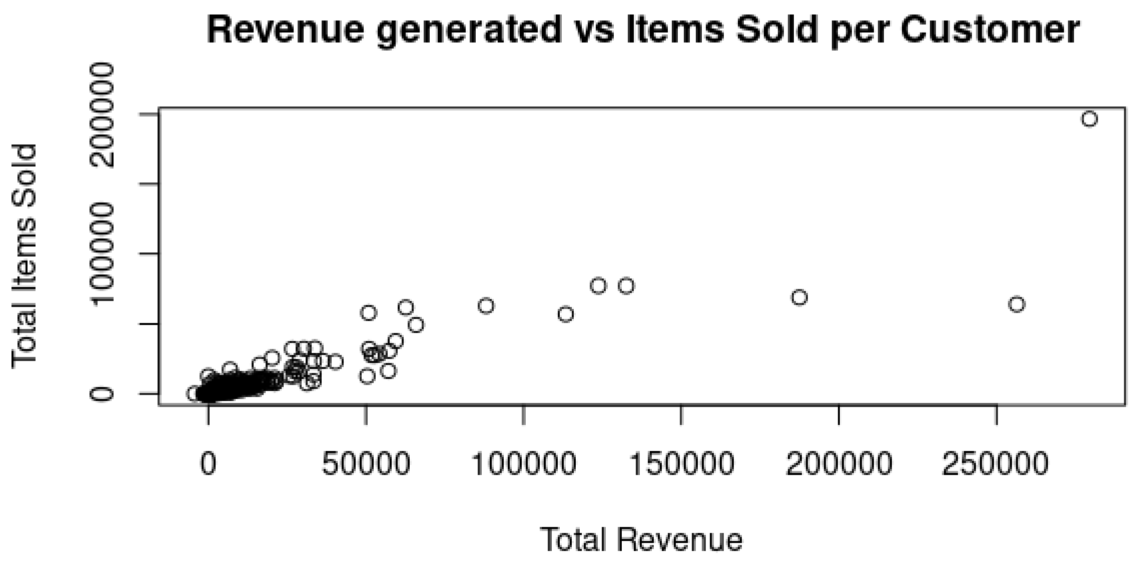
```
unique(countrySummary$Country)
```



```
 [1] "United Kingdom"       "Netherlands"          "EIRE"                 "Germany"
 [5] "France"               "Australia"            "Switzerland"          "Spain"
 [9] "Belgium"              "Sweden"               "Japan"                "Norway"
[13] "Portugal"             "Finland"              "Channel Islands"      "Denmark"
[17] "Italy"                "Cyprus"               "Austria"              "Singapore"
[21] "Poland"               "Israel"               "Greece"               "Iceland"
[25] "Canada"               "Unspecified"          "Malta"                "United Arab Emirates"
[29] "USA"                  "Lebanon"              "Lithuania"            "European Community"
[33] "Brazil"               "RSA"                  "Czech Republic"       "Bahrain"
[37] "Saudi Arabia"
```

A lot of different countries contribute a good amount of revenue. As it seems that refunds and/or cancellations are present in the dataset as revenue with a negative value, we can assume that the revenue figures here are net of refunds; something that it is important to consider when shipping goods overseas. However, additional information on the costs incurred dealing with these refunds would allow us to make more appropriate recommendations.

```
# Plotting Revenue generated vs Items Sold per Customer

plot(x = customerData$TotalRevenue,
    y = customerData$TotalItemsSold,
    xlab = "Total Revenue",
    ylab = "Total Items Sold",
    main = "Revenue generated vs Items Sold per Customer"
)
```



The graph shows the majority of the sales occupying the space near the 0 point of the graph, with a few exceptions, that seem to buy huge quantities with little revenue generated or a handful of items with massive revenue.

5. K-MEANS CLUSTERING

5.1 Introduction to unsupervised learning and k-means clustering

Clustering is a broad set of techniques for finding subgroups of observations within a data set. When we cluster observations, we want observations in the same group to be similar and observations in different groups to be dissimilar. Because there isn't a response variable, this is an unsupervised method, which implies that it seeks to find relationships between the n observations without being trained by a response variable. Clustering allows us to identify which observations are alike, and potentially categorize them therein. K-means clustering is the simplest and the most commonly used clustering method for splitting a dataset into a set of k groups.

In contrast to supervised learning where the label (output) of a predictive model is explicitly specified in advance, unsupervised learning allows the algorithm to identify the clusters within the data itself and subsequently label them accordingly. K-means clustering is an example of an unsupervised learning algorithm and it works as follows:

1. Choose a number of clusters, K (this is what the k stands for in k-means clustering), into which the data are to be divided
2. Assign arbitrary K number of cluster centres
3. Assign data points to cluster centre which is nearest to them, most commonly using the Euclidean distance formula
4. Once all the data points have been clustered according to their cluster centres, calculate the centroid of each cluster, using the mean values of the data points assigned to that cluster
5. Reset cluster centres using those centroids
6. Repeat steps 3 to 5 until no re-assignment of cluster centres take place i.e., until the algorithm has fully converged

In machine learning, the value K is what we would call a hyper-parameter. Hyper-parameters are external to the mode and their values cannot be estimated from the data. Instead, they are typically specified by the user using rough guidelines, past experiences and experimentation.

The basic idea behind k-means clustering consists of defining clusters so that the total intra-cluster variation (known as total within-cluster variation) is minimized. There are several k-means algorithms availableIn machine learning, the value K is what we would call a hyper-parameter. Hyper-parameters are external to the mode and their values cannot be estimated from the data. Instead, they are typically specified by the user using rough guidelines, past experiences and experimentation.

. 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:

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

where:

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

We define the total within-cluster variation as follows:

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

The *total within-cluster sum of squares* measures the compactness (i.e goodness) of the clustering and we want it to be as small as possible.

5.2 Computing k-means clustering in R

We can compute k-means in R with the k means function. Here will group the data into three clusters (centers = 3). The k means function also has a start option that attempts multiple initial configurations and reports on the best one. For example, adding start = 25 will generate 25 initial configurations. This approach is often recommended.

```
> # Grouping the data in to 3 clusters using k-means
> k3 <- kmeans(customerData, centers = 3, nstart = 25)
> str(k3)
List of 9
 $ cluster     : int [1:3708] 2 3 2 3 2 3 2 1 3 1 ...
 $ centers     : num [1:3, 1:2] 1.93 -0.632 0.375 1.91 -0.658 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "1" "2" "3"
  .. ..$ : chr [1:2] "TotalRevenue" "TotalItemsSold"
 $ totss       : num 7414
 $ withinss    : num [1:3] 543 378 448
 $ tot.withinss: num 1368
 $ betweenss   : num 6046
 $ size        : int [1:3] 527 2194 987
 $ iter        : int 2
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

The output of k means 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.
- tot.withinss: Total within-cluster sum of squares, i.e., sum(withinss).
- betweenss: The between-cluster sum of squares, i.e., $totss-tot.withinss$.
- size: The number of points in each cluster.

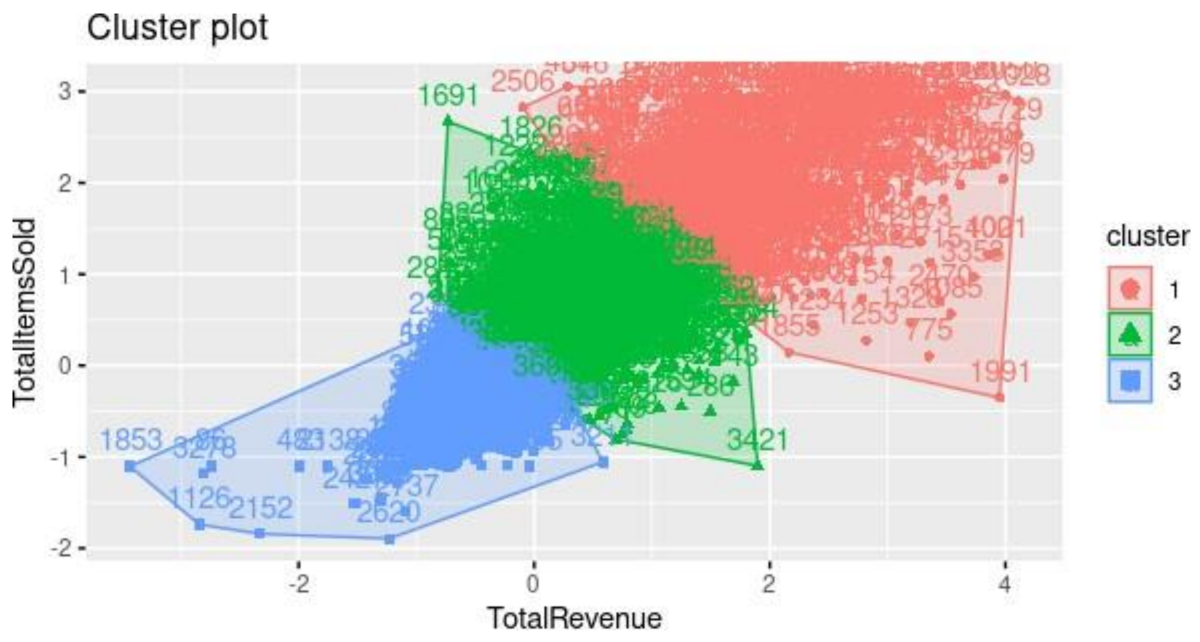
If we print the results, we'll see that our groupings resulted in 3 cluster sizes of *527, 987 and 2194*. We see the cluster centers (means) for the three groups across the two variables (*TotalRevenue, TotalItemsSold*). We also get the cluster assignment for each observation.

```
> k3
K-means clustering with 3 clusters of sizes 527, 987, 2194

Cluster means:
  TotalRevenue TotalItemsSold
1    1.9302862      1.9103617
2    0.3749642      0.4423527
3  -0.6323384     -0.6578682

Clustering vector:
   [1] 3 2 3 2 3 2 3 1 2 1 3 3 1 3 3 2 2 3 2 3 1 2 1 1 1 3 3 1 3 3 2 1 1 1 3 3 3 1 1 1 3 2 3 3 2 3 2 1
  [49] 1 2 3 3 3 2 3 1 3 3 3 2 3 3 3 3 2 3 2 3 2 2 2 2 2 2 3 3 1 2 3 3 3 3 2 1 3 3 3 3 3 2 3 3 2 3 2 2
  [97] 3 2 1 1 3 1 3 3 1 2 3 1 1 3 3 2 3 3 2 3 2 2 3 3 2 3 3 3 3 3 1 3 3 3 3 2 2 3 3 2 2 1 3 3 3 3 2 1
 [145] 3 3 3 3 3 3 3 2 2 1 3 3 3 3 3 3 2 1 2 3 3 1 3 3 3 1 3 3 3 1 2 2 3 1 1 2 3 2 3 2 3 1 2 3 2 3 3 2
 [193] 1 3 2 3 3 3 3 3 3 1 1 3 3 1 2 2 2 1 3 2 3 2 3 2 3 2 2 3 3 1 3 1 2 2 3 2 2 3 3 2 2 2 3 2 3 3 3 3
 [241] 2 3 3 2 3 3 3 1 3 1 1 3 2 2 2 2 2 2 2 1 2 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 2 3 3 1 2 3 2 2 3 1
 [289] 3 2 3 3 3 3 3 1 3 3 1 2 2 3 3 2 2 3 3 1 2 3 1 2 3 3 1 3 1 3 3 3 3 1 2 3 3 2 3 3 2 2 3 3 2 3 3 3
 [337] 3 3 3 1 2 1 3 1 3 2 3 3 3 2 2 1 3 3 2 2 1 3 2 3 3 3 3 1 1 1 2 2 3 3 3 1 3 3 2 3 3 3 2 3 3 3 3 1
 [385] 3 3 3 3 3 2 3 2 3 3 3 3 3 3 3 3 2 1 2 3 1 3 1 2 2 2 2 3 3 2 2 2 3 3 2 3 3 1 1 3 3 2 3 3 3 3 3 2
 [433] 3 3 3 3 3 3 2 2 3 3 3 3 3 3 1 3 3 3 3 2 3 1 3 2 2 1 3 2 2 3 3 2 3 3 2 3 3 1 1 2 3 1 3 3 3 3 3 1
 [481] 2 3 3 2 1 2 1 3 3 1 3 2 2 3 3 3 3 1 3 3 3 3 2 3 3 2 2 3 3 1 2 2 3 2 3 2 3 1 3 3 3 3 3 3 1 3 3 3
 [529] 3 1 2 2 3 2 3 3 3 3 3 3 3 1 3 3 2 1 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 3 3 3 3 2 3 3 1 3 3 3 3 3 3
 [577] 1 3 2 1 2 3 2 2 1 3 1 3 3 3 3 2 2 2 3 2 3 2 3 3 1 3 3 3 3 2 3 3 2 3 2 3 3 3 1 2 3 1 3 3 3 3 3 2
 [625] 2 2 3 3 3 2 3 3 3 2 2 3 2 3 3 3 2 3 1 3 3 3 3 1 1 1 2 3 1 2 2 3 3 1 3 3 2 3 3 3 2 1 3 1 3 2 3 2
 [673] 3 1 3 3 1 2 1 3 1 3 3 3 3 2 3 2 2 2 3 3 1 3 3 1 2 2 3 3 3 3 3 2 1 3 3 3 2 1 1 2 3 2 3 3 2 3 3 3
 [721] 3 2 3 3 3 2 2 3 1 2 3 3 3 1 3 2 3 2 3 3 3 2 3 2 2 3 3 2 3 3 3 2 3 1 3 2 3 3 2 3 3 3 3 2 2 2 3 3
 [769] 3 3 3 3 1 1 1 1 2 2 2 2 3 2 3 3 3 1 3 1 3 2 2 3 3 1 1 2 3 3 3 3 3 3 3 3 2 1 2 3 3 3 2 3 1 3 3 3
 [817] 3 1 2 3 2 3 3 3 3 2 3 3 2 2 3 1 3 2 2 3 2 2 3 2 1 2 3 3 3 3 1 1 3 3 2 2 3 2 2 3 1 3 3 2 3 2 1 1
 [865] 1 2 3 3 3 3 2 3 3 2 2 3 2 3 3 3 1 3 3 1 3 2 3 3 2 1 3 3 1 3 2 2 1 3 3 3 3 1 3 3 2 3 3 3 3 3 2 1
 [913] 1 2 3 3 3 2 3 3 3 2 1 3 3 1 3 3 2 2 1 3 3 3 3 3 3 1 2 3 3 1 3 3 3 2 3 1 3 3 1 3 3 3 1 3 3 3 3 3
 [961] 3 2 3 3 2 3 3 3 2 2 2 3 3 3 2 1 3 3 2 3 3 3 3 2 3 3 3 3 3 3 2 3 1 1 3 3 1 3 2 3
 [ reached getOption("max.print") -- omitted 2708 entries ]

Within cluster sum of squares by cluster:
[1] 543.0121 447.8109 377.5755
 (between_SS / total_SS =  81.5 %)

Available components:
```

```
[1] "cluster"     "centers"     "totss"       "withinss"    "tot.withinss" "betweenss"
[7] "size"        "iter"        "ifault"
```

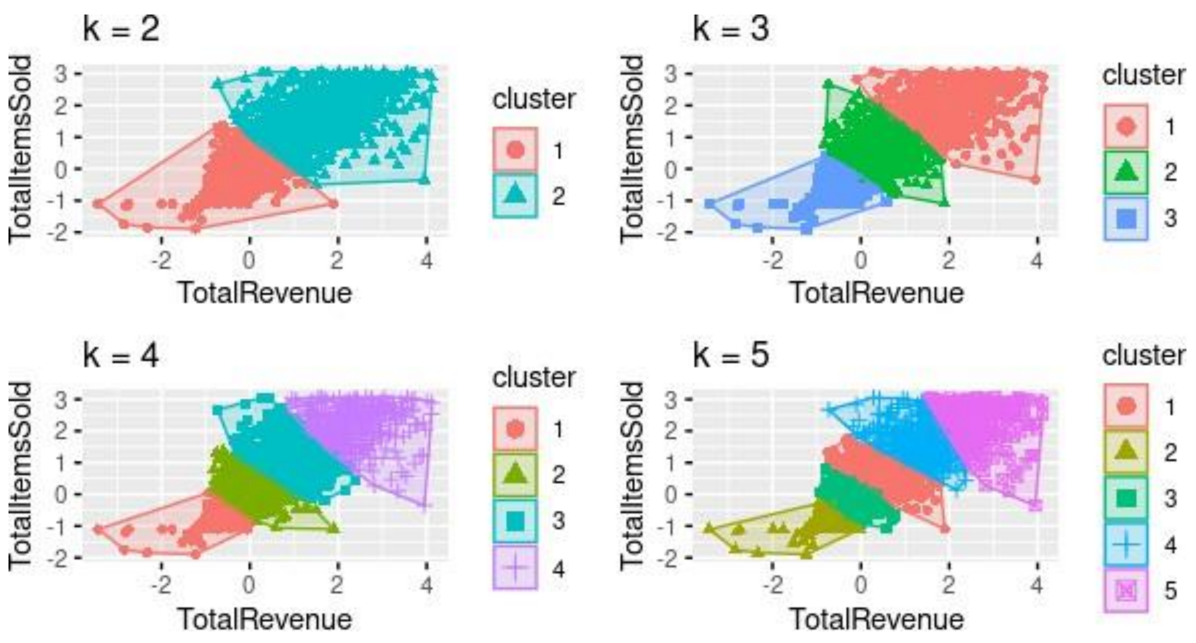
We can also view our results by using ***fviz_cluster***. This provides a nice illustration of the clusters. If there are more than two dimensions (variables) fviz_cluster will perform principal component analysis (PCA) and plot the data points according to the first two principal components that explain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

```
> fviz_cluster(k3, data = customerData)
```



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 2, 4, and 5 clusters, and the results are shown in the figure:

```
# Executing kmeans for 2,4,5 number of clusters
k2 <- kmeans(customerData, centers = 2, nstart = 25)
k4 <- kmeans(customerData, centers = 4, nstart = 25)
k5 <- kmeans(customerData, centers = 5, nstart = 25)

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

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



Although this visual assessment tells us where true delineations occur between clusters, it does not tell us what the optimal number of clusters is.

5.3 Determining optimum clusters using Elbow method

One of the most common ways to select the number of clusters is to identify a number after which there is no longer a significant improvement in the total within-groups sum of squares.

The total within-groups sum of squares is a measure of the within-cluster homogeneity i.e., how similar the data points in each cluster are. As you would expect, increasing the number of clusters will cause the data points in each cluster to be more similar to each other as the clusters become more specific to each data point, thus decreasing the total within-groups sum of squares.

What we are looking for is when the plot of the total within-groups sum of squares starts to plateau, indicating a diminishing improvement if we were to further increase the number of clusters. The rationale behind this is so that we do not risk overfitting or in other words, prevent our algorithm from learning the noise in the dataset rather than just the signal.

The Elbow method

The elbow method is used to determine the optimal number of clusters in k-means clustering. The elbow method plots the value of the cost function produced by different values of *k*. If *k* increases, average distortion will decrease, each cluster will have fewer constituent instances, and the instances will be closer to their respective centroids. However, the improvements in average distortion will decline as *k* increases. The value of *k* at which improvement in distortion declines the most is called the elbow, at which we should stop dividing the data into further clusters.

The basic idea is to define clusters such that the total intra-cluster variation (known as total within-cluster variation or total within-cluster sum of square) is minimized:

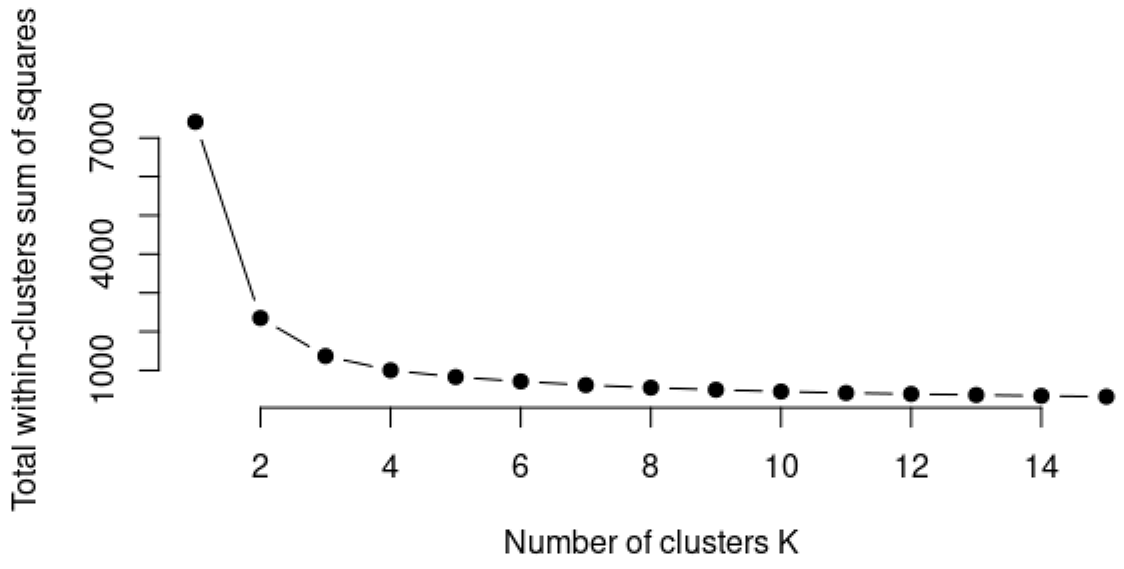
$$minimize\left(\sum_{k=1}^{k} W(C_k)\right)$$

where C_k is the k^{th} cluster and $W(C_k)$ is the within-cluster variation.

The total within-cluster sum of squares (wss) measures the compactness of the clustering and we want it to be as small as possible. Thus, we can use the following algorithm to define the optimal clusters:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of *k*. For instance, by varying *k* from 1 to 10 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.

```
# Determining optimal number of clusters using Elbow Method
set.seed(123)

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



To determine the optimal number of clusters, we have to select the value of k at the "elbow" i.e., the point after which the distortion/inertia start decreasing in a linear fashion. Thus, for the given data, we conclude that the optimal number of clusters for the ecommerce data is **3**.

Therefore, there are ***527*** customers who are highly valued when identified using K-means clustering algorithm.

6. HIERARCHICAL CLUSTERING

6.1 Introduction to Hierarchical clustering

A **Hierarchical clustering** method works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data point as a separate cluster. Then, it repeatedly executes the subsequent steps:

1. Identify the 2 clusters which can be closest together, and
2. Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.

In Hierarchical Clustering, the aim is to produce a hierarchical series of nested clusters. A diagram called **Dendrogram** (A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits) graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or cluster are break up (top-down view).

6.2 Hierarchical Clustering Algorithms: Agglomerative and Divisive

The basic method to generate hierarchical clustering are:

1. Agglomerative:

Initially consider every data point as an **individual** Cluster and at every step, **merge** the nearest pairs of the cluster. (It is a bottom-up method). At first everydata data set is considered as an individual entity or cluster. At every iteration, the clusters merge with different clusters until one cluster is formed.

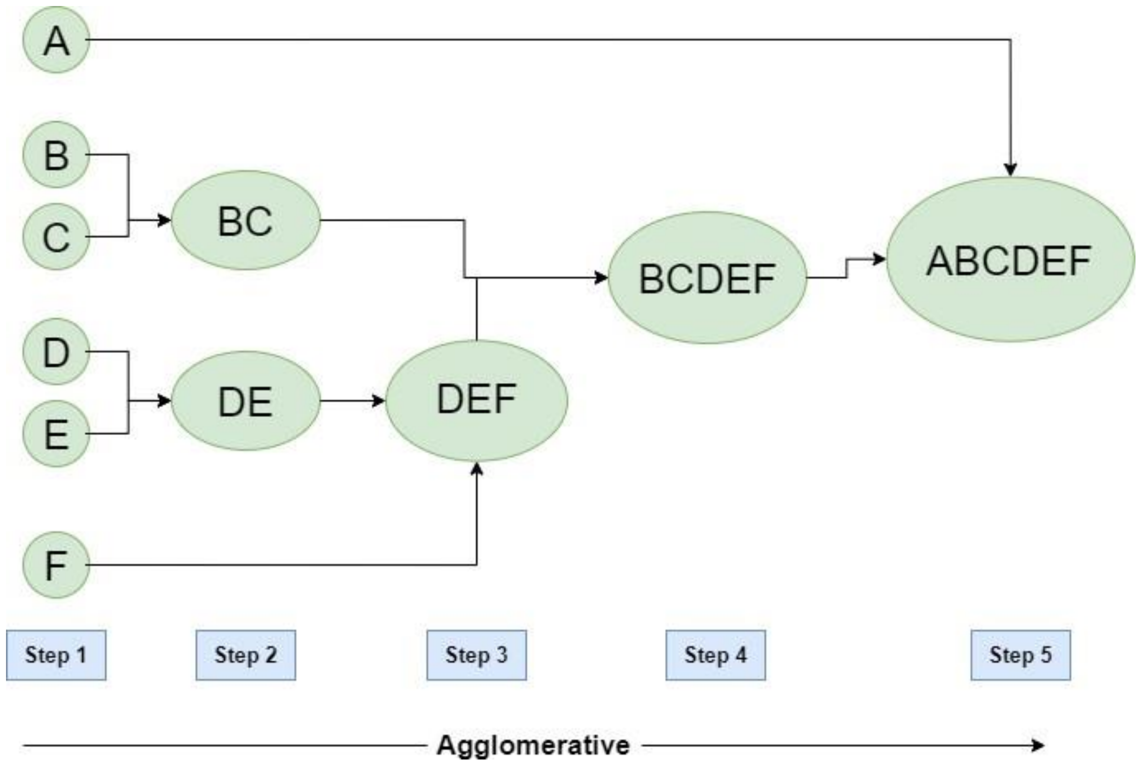
Algorithm for Agglomerative Hierarchical Clustering is:

- Calculate the similarity of one cluster with all the other clusters (calculate proximity matrix)
- Consider every data point as a individual cluster
- Merge the clusters which are highly similar or close to each other.

- Recalculate the proximity matrix for each cluster
- Repeat Step 3 and 4 until only a single cluster remains.

A demonstration of how the actual algorithm works no calculation has been performed below all the proximity among the clusters are assumed.

Let's say we have six data points **A, B, C, D, E, F**.

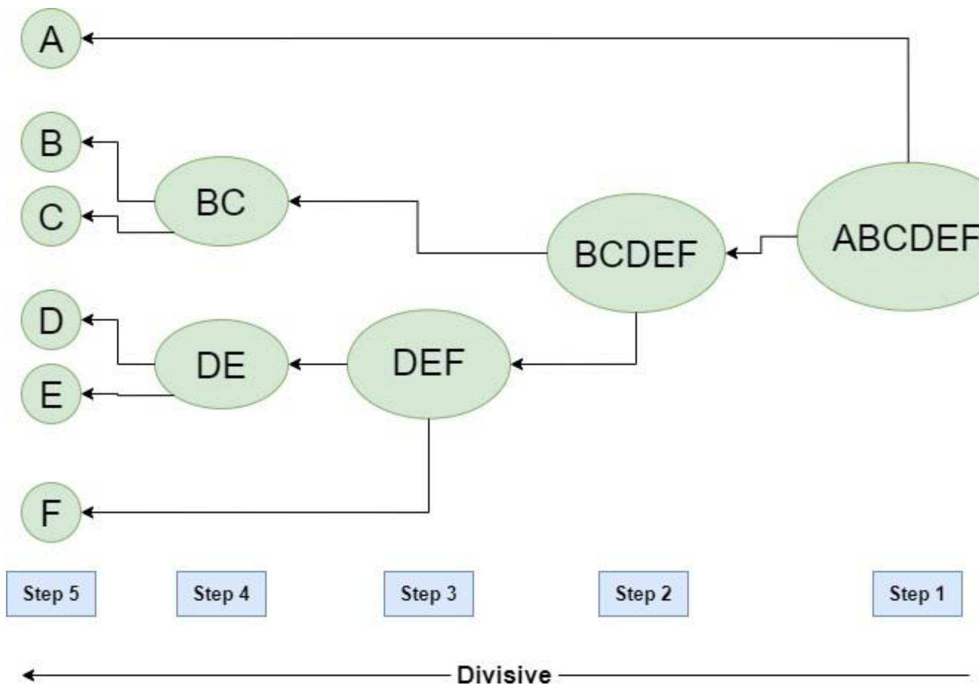


- **Step-1:**
  Consider each alphabet as a single cluster and calculate the distance of one cluster from all the other clusters.
- **Step-2:**
  In the second step comparable clusters are merged together to form a single cluster. Let's say cluster (B) and cluster (C) are very similar to each other therefore we merge them in the second step similarly with cluster (D) and (E) and at last, we get the clusters
  [(A), (BC), (DE), (F)]

- **Step-3:**
  We recalculate the proximity according to the algorithm and merge the two nearest clusters([(DE), (F)]) together to form new clusters as [(A), (BC), (DEF)]
- **Step-4:**
  Repeating the same process; The clusters DEF and BC are comparable and merged together to form a new cluster. We're now left with clusters [(A), (BCDEF)].
- **Step-5:**
  At last the two remaining clusters are merged together to form a single cluster [(ABCDEF)].

2. Divisive:

We can say that the Divisive Hierarchical clustering is precisely the **opposite** of the Agglomerative Hierarchical clustering. In Divisive Hierarchical clustering, we take into account all of the data points as a single cluster and in every iteration, we separate the data points from the clusters which aren't comparable. In the end, we are left with N clusters.



6.3 Measure of dissimilarity between two clusters of observations

In k-means, we measure the (dis)similarity of observations using distance measures (i.e. Euclidean distance, Manhattan distance, etc.) In R, the Euclidean distance is used by default to measure the dissimilarity between each pair of observations. As we already know, it's easy to compute the dissimilarity measure between two pairs of observations with the `get_dist` function.

However, a bigger question is: *How do we measure the dissimilarity between two clusters of observations?* A number of different cluster agglomeration methods (i.e, linkage methods) have been developed to answer this question. The most common types of methods are:

- **Maximum or complete linkage clustering:** It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the largest value (i.e., maximum value) of these dissimilarities as the distance between the two clusters. It tends to produce more compact clusters.
- **Minimum or single linkage clustering:** It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the smallest of these dissimilarities as a linkage criterion. It tends to produce long, "loose" clusters.
- **Mean or average linkage clustering:** It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2 and considers the average of these dissimilarities as the distance between the two clusters.
- **Centroid linkage clustering:** It computes the dissimilarity between the centroid for cluster 1 (a mean vector of length p variables) and the centroid for cluster 2.
- **Ward's minimum variance method:** It minimizes the total within-cluster variance. At each step the pair of clusters with minimum between-cluster distance are merged.

6.4 Computing Hierarchical Clustering in R

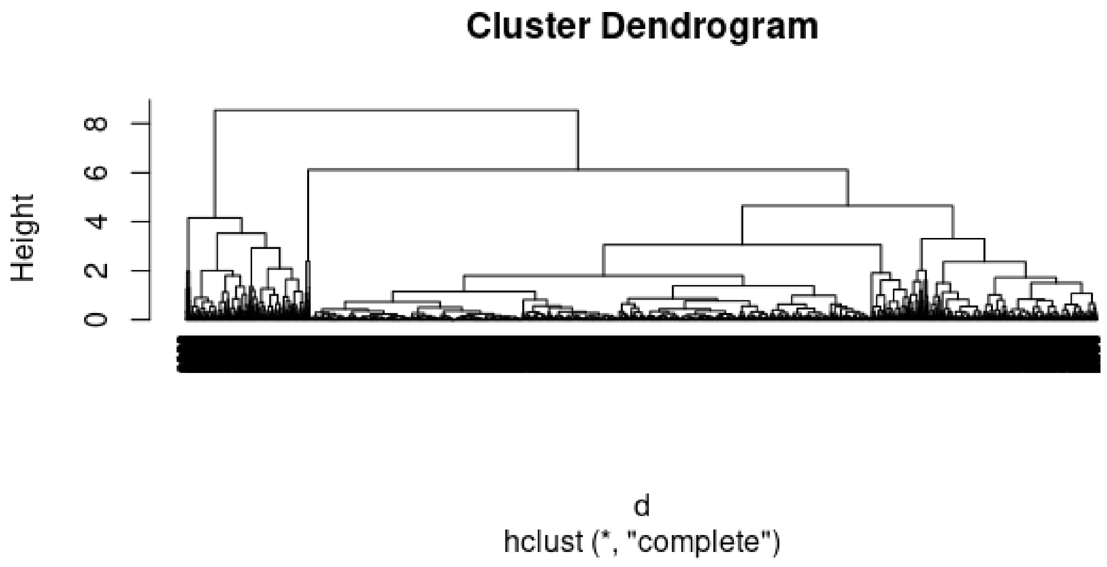
There are different functions available in R for computing hierarchical clustering. The commonly used functions are:

- hclust [in stats package] and agnes [in cluster package] for agglomerative hierarchical clustering (HC)
- diana [in cluster package] for divisive HC

Agglomerative Hierarchical Clustering

We can perform agglomerative HC with hclust. First, we compute the dissimilarity values with dist and then feed these values into hclust and specify the agglomeration method to be used (i.e. “complete”, “average”, “single”, “ward.D”).

```
# Dissimilarity matrix
d <- dist(customerData, method = "euclidean")

# Hierarchical clustering using Complete Linkage
hc1 <- hclust(d, method = "complete" )

# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1)
```

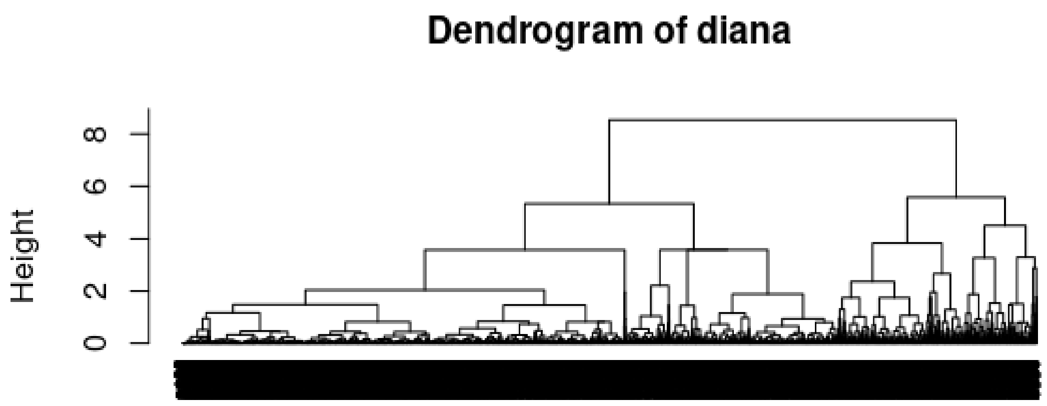


Alternatively, we can use the agnes function. These functions behave very similarly.

Divisive Hierarchical Clustering

The R function diana provided by the cluster package allows us to perform divisive hierarchical clustering. Diana works similar to agnes; however, there is no method to provide.

```
# compute divisive hierarchical clustering
hc2 <- diana(customerData)

# Divise coefficient; amount of clustering structure found
hc2$dc
## [1] 0.8514345

# plot dendrogram
pltree(hc2, cex = 0.6, hang = -1, main = "Dendrogram of diana")
```



6.5 Assessing various methods

```
# methods to assess
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")

# function to compute coefficient
ac <- function(x) {
  agnes(customerData, method = x)$ac
}
map_dbl(m, ac)
```

```
  average    single  complete      ward
0.9912533 0.9671068 0.9965356 0.9997057
```

Agglomerative coefficient, which measures the amount of clustering structure found (values closer to 1 suggest strong clustering structure) can be used to identify stronger clustering structures. The Ward's method is identified as the strongest clustering structure
of the four methods assessed with agglomerative coefficient **0.9997**

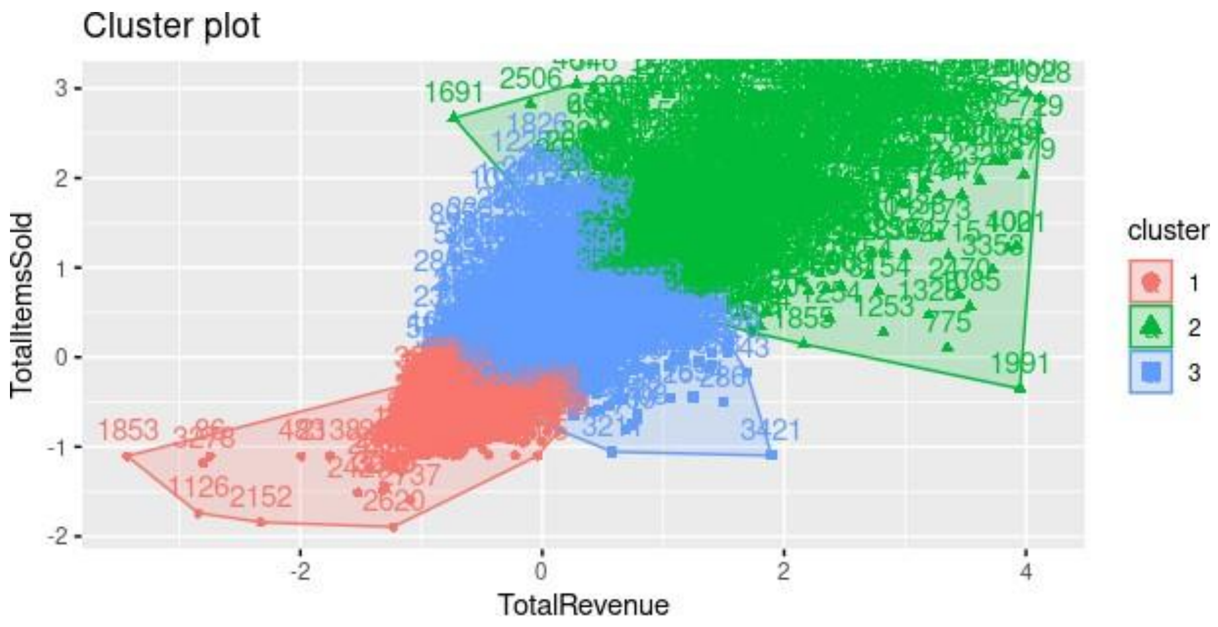
6.6 Working with Dendrograms

In the dendrogram displayed above, each leaf corresponds to one observation. As we move up the tree, observations that are similar to each other are combined into branches, which are themselves fused at a higher height.

The height of the fusion, provided on the vertical axis, indicates the (dis)similarity between two observations. The higher the height of the fusion, the less similar the observations are. Note that, conclusions about the proximity of two observations can be drawn only based on the height where branches containing those two observations first are fused. We cannot use the proximity of two observations along the horizontal axis as a criterion of their similarity.

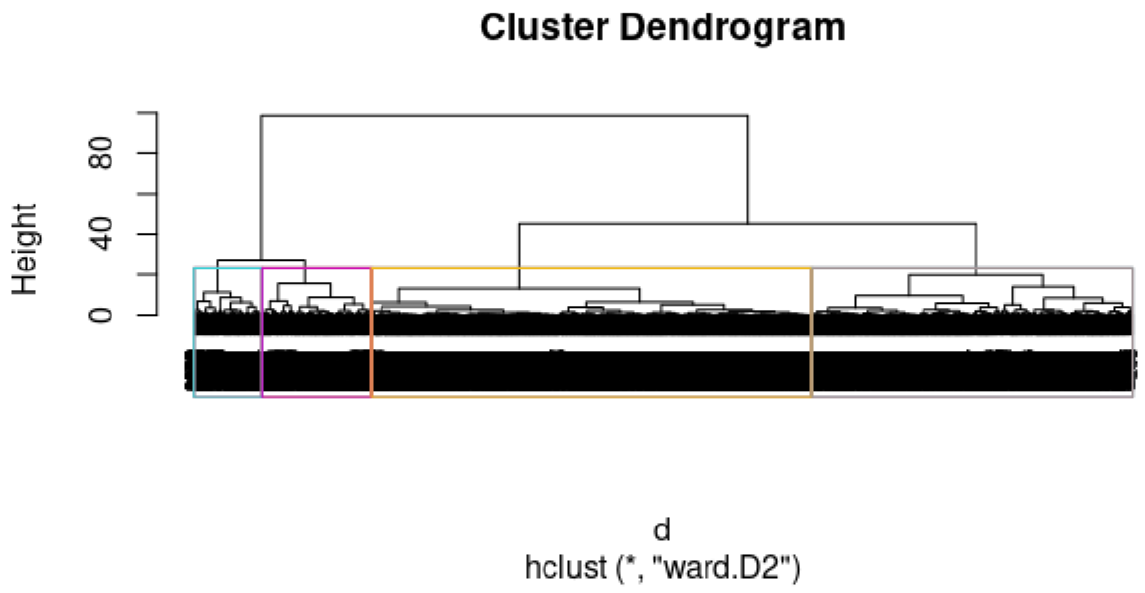
The height of the cut to the dendrogram controls the number of clusters obtained. It plays the same role as the k in k-means clustering. In order to identify sub-groups (i.e., clusters), we cancut the dendrogram with *cutree*.

```
# Ward's method minimizes the total within-cluster variance
# At each step the pair of clusters with minimum between-cluster distance are merged
hc3 <- hclust(d, method = "ward.D2" )

# Cut tree into 3 groups
sub_grp <- cutree(hc3, k = 3)

# Number of members in each cluster
table(sub_grp)

# Visualizing the result
fviz_cluster(list(data = customerData, cluster = sub_grp))
```



It’s also possible to draw the dendrogram with a border around the 4 clusters. The argument border is used to specify the border colors for the rectangles.

```
plot(hc3, cex = 0.6)
rect.hclust(hc3, k = 4, border = 5:10)
```

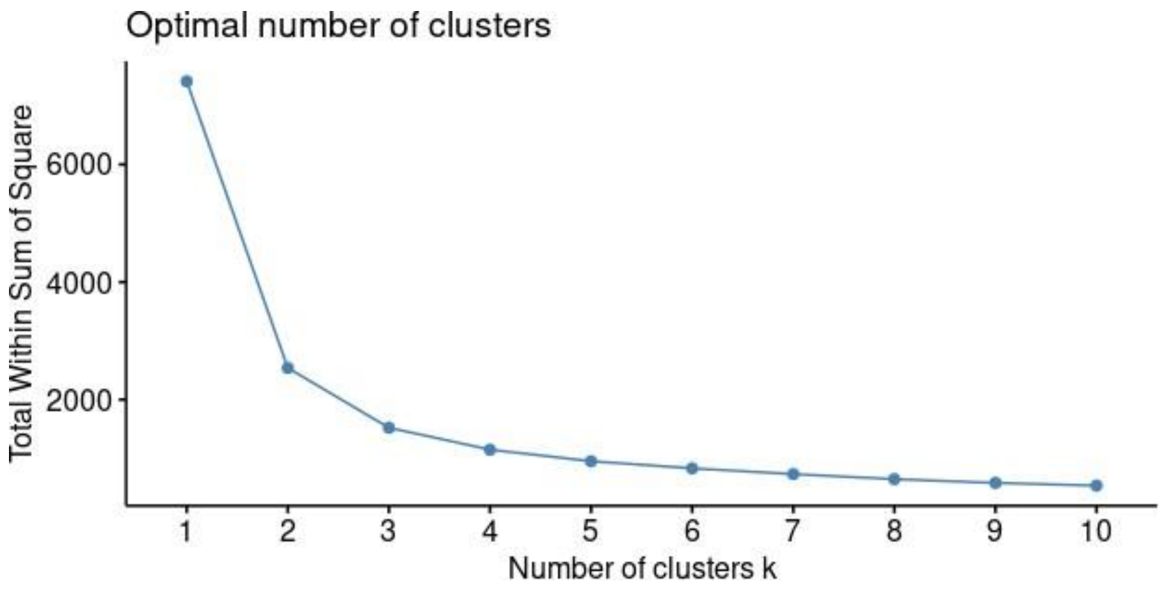


6.7 Determining Optimal clusters

Similar to how we determined optimal clusters with k-means clustering, we can execute similar approaches for hierarchical clustering.

Elbow Method

The elbow method looks at the percentage of variance explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point, the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion". This "elbow" cannot always be unambiguously identified.
To perform the elbow method we just need to change the second argument in *fviz_nbclust* to *FUN = hcut*.



Here also, for the given data, we conclude that the optimal number of clusters for the ecommerce data is **3**. Therefore, there are ***700*** customers who are highly valued when identified using Hierarchical clustering algorithm.

6.8 WSS FOR SOME VALUE 'K'

```
wssForHierarchical <- function(k){
  sub_grp <- cutree(hc3, k)
  df=as.data.frame(customerData)

  df2=df %>%
    mutate(cluster = sub_grp)%>%
    group_by(cluster)%>%
    summarize(TR=mean(TotalRevenue), TS=mean(TotalItemsSold))

  df1=df  %>%
    mutate(cluster = sub_grp)

  df3=left_join(df1,df2,by="cluster")

  D=(df3$TotalRevenue-df3$TR)^2+(df3$TotalItemsSold-df3$TS)^2

  df4=cbind(df3,D)
  WSS=sum(df4$D)
  WSS
}

# Calculating WSS at hcut=3
wssForHierarchical(3)

## [1] 1522.435
```

As the optimal number of clusters determined by using the elbow method is **3,** we have calculated WSS value for the same. The obtained result is **1522.435** can be used to draw further conclusions.

7. Conclusion and Future Scopes

Clustering is rather a subjective statistical analysis and there can be more than one appropriate algorithm, depending on the dataset at hand or the type of problem to be solved. So, choosing between k-means and hierarchical clustering is not always easy. If you have a good reason to think that there is a specific number of clusters in your dataset (for example if you would like to distinguish diseased and healthy patients depending on some characteristics but you do not know in which group patients belong to), you should probably opt for the k-means clustering as this technique is used when the number of groups is specified in advance. If you do not have any reason to believe there is a certain number of groups in your dataset (for instance in marketing when trying to distinguish clients without any prior belief on the number of different types of customers), then you should probably opt for the hierarchical clustering to determine in how many clusters your data should be divided.

In addition to this, if you are still undecided, note that, on the one hand, with a large number of variables, k-means may be computationally faster than hierarchical clustering if the number of clusters is small. On the other hand, the result of a hierarchical clustering is a structure that is more informative and interpretable than the unstructured set of flat clusters returned by k-means. Therefore, it is easier to determine the optimal number of clusters by looking at the dendrogram of a hierarchical clustering than trying to predict this optimal number in advance in case of k-means.

As we have seen in the above section, the results of the clustering are almost similar to the same dataset. It may be possible that when we have a very large dataset, the shape of clustersmay differ a little.

Many of the popular methods used for determining the optimal number of clusters for algorithms such as k-means are based on the within-cluster sum of squares (WSS). The WSS metric is based on the distance between the observations and the cluster centroids. In general, the lower the WSS, the closer the observations are to the centroids, which indicates the better fit. However, we need to find a balance between the WSS and the number of clusters, as increasing the number of clusters indefinitely (up until the number of observations) should always result in a better fit.

If we were to consider this metric, we obtained a total within SS of 1368 through k-means and 1522 through Agglomerative Hierarchical Clustering. As stated above, lower WSS indicates a better fit. So, we can conclude that the K-means clustering algorithm resulted better and 527 customers are identified as high valued customers. This is not it, there are other metrics to explore and be evaluated.

Clustering can be a very useful tool for data analysis in the unsupervised setting. However, there are a number of issues that arise in performing clustering. Each of the decisions we make can have a strong impact on the results obtained. In practice, we try several different choices, and look for the one with the most useful or interpretable solution. With these methods, there is no single right answer - any solution that exposes some interesting aspects of the data should be considered.

8. REFERENCES

[1] *Blanchard, Tommy. Bhatnagar, Pranshu. Behera,Trash*. (2019). Marketing Analytics Scientific Data: Achieve your marketing objectives with Python's data analytics capabilities. S.l: Packt printing is limited

[2] *Griva, A., Bardaki, C., Pramatari, K.,Papakiriakopoulos, D.* (2018). Sales business analysis:Customer categories use market basket data. Systems Expert Systems, 100, 1-16.

[3] *Hong, T., Kim, E.* (2011). It separates consumers from online stores based on factors that affect the customer's intention to purchase. Expert System Applications, 39 (2), 2127-2131.

[4] *Hwang, Y. H.* (2019). Hands-on Advertising Science Data: Develop your machine learning marketing strategies... using python and r. S.l:Packt printing is limited

[5] *Puwanenthiren Premkanth,* -Market Classification and Its Impact on Customer Satisfaction and Special Reference to the Commercial Bank of Ceylon PLC.‖ Global Journal of Management and Business Publisher Research: Global Magazenals Inc. (USA). 2012. Print ISSN: 0975-5853. Volume 12 Issue 1.

[6] *Puwanenthiren Premkanth,*-Market Classification and Its Impact on Customer Satisfaction and Special Reference to the Commercial Bank of Ceylon PLC.‖ Global Journal of Management and Business Publisher Research: Global Magazenals Inc. (USA). 2012. Print ISSN: 0975-5853. Volume 12 Issue 1.

[7] By *Jerry W Thomas*. 2007. Accessed at:www.decisionanalyst.com on July 12, 2015.

[8] *Jianfu, L., Jianshuang L., Huaiqing H.* (2011). A Simple and Accurate Approach to Hierarchical Clustering. Journal of Computational Information Systems, 7(7), 2577--2584.