# GROUP 320: CUSTOMER SEGMENTATION AND MARKET BASKET ANALYSIS FOR AN ONLINE RETAIL


| First Name | Last Name | Email address |
|---|---|---|
| Bhavya Sree | Bindela | bbindela@hawk.iit.edu |


## Table of Contents

# Table of Figures

# 1. Introduction

Customer Segmentation and Market analysis for an online retail non-store is to explore different types of customers and their granular purchase behavior.

**Customer segmentation** is the process of dividing the customers into different groups based on their behavior with the company. There will be different types of customers for any firm like some customers will be frequent buyers, some customers will be less frequent but buy high-value products, and so on. These are called different segments of customers. Using customer segments, the company can market their strategies to each group effectively.

**Market Basket Analysis** is a technique to identify the strength of association between pairs of products that are purchased together. This technique is based on a concept that if a person buys a product A, how likely the person will buy a product B.

Therefore, Market Basket Analysis is to uncover the product purchase behavior of customers by extracting the association rules of the products. These will be helpful to understand purchase decisions made by the customers and the company can use the association rules in their future marketing strategies.

# 2. Data

Dataset belongs to a UK-based registered non-store online retail.

This dataset contains all the transactions that occurred between 01/12/2010 and 09/12/2011.

This dataset has been collected from UCI Machine Learning Repository.

The dataset can be found in the below link

https://archive.ics.uci.edu/ml/datasets/online+retail

**Format of the dataset**

Number of Instances: 541909

Number of Attributes: 8

| Attribute | Description | Data Type |
|---|---|---|
| InvoiceNo | Invoice number | Character |
| StockCode | Product Code | Character |
| Description | Product Name | Character |
| Quantity | Quantity of each product per transaction | Number |
| InvoiceDate | Invoice Date and Time | Number |
| UnitPrice | Price of the Product per unit | Number |
| CustomerID | Customer Number | Number |
| Country | Country Name | Character |

*Table 1: Metadata of the datafile*

In this dataset, there is no target variable, as we will be doing unsupervised learning on the online retail dataset.

The whole project will be implemented in R.

# 3. Problems and Solutions

Below are the research problems identified to solve in this project.

1. Identify the customers into different groups based on their behavior with the online retail. Identify the Loyal Customers.

2. Identify best 10 Association rules for the products purchased in the online retail. I.e., to identify the Products that are frequently bought together in the online retail.

# 4. KDD

## 4.1. Data Processing

### 4.1.1. Identify the Customers into different groups based on their behavior with the online retail.

- As a first step, load the data into R session.

```
> data=read.xlsx("Online Retail.xlsx", 1)
>
> # 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"
OT WATER BOTTLE" ...
 $ Quantity   : num  6 6 8 6 6 2 6 6 6 32 ...
 $ InvoiceDate: num  40513 40513 40513 40513 40513 ...
 $ UnitPrice  : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
 $ CustomerID : num  17850 17850 17850 17850 17850 ...
 $ Country    : chr  "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" ...
```

*Figure 1: Load the data and Structure of data*

As the type of the InvoiceDate is Numerical, changed its type to date.

```
> # Change the type of InvoiceDate
> data$InvoiceDate <- as.POSIXct(data$InvoiceDate * (60*60*24),
+                                origin="1899-12-30", tz="GMT")
> head(data)
  InvoiceNo StockCode                          Description Quantity         InvoiceDate UnitPrice CustomerID        Country
1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6 2010-12-01 08:26:00      2.55      17850 United Kingdom
2    536365     71053                 WHITE METAL LANTERN        6 2010-12-01 08:26:00      3.39      17850 United Kingdom
3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8 2010-12-01 08:26:00      2.75      17850 United Kingdom
4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6 2010-12-01 08:26:00      3.39      17850 United Kingdom
5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6 2010-12-01 08:26:00      3.39      17850 United Kingdom
6    536365     22752        SET 7 BABUSHKA NESTING BOXES        2 2010-12-01 08:26:00      7.65      17850 United Kingdom
```

*Figure 2: Change the type of InvoiceDate*

### (I) Cleaning the data

- Check for the Missing values

```
> # Check for the sum of Null values in all columns
> colSums(is.na(data))
  InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice  CustomerID     Country
          0           0        1454           0           0           0      135080           0
```

*Figure 3: check for Missing values*

Data has missing values in Description and CustomerID columns.

As CustomerID is the unique ID given to every customer, we cannot replace the missing values with any valid value. Therefore, removing the records with Null values in CustomerID.

```
> # Omit the rows with Null values in CustomerID.
> library(tidyr)
> retail_data <- data %>%
+   drop_na(CustomerID)
>
> # Check for Null values
> colSums(is.na(retail_data))
  InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice  CustomerID     Country
          0           0           0           0           0           0           0           0
```

*Figure 4: Remove the records with Null values in CustomerID*

After removing the records with missing values in CustomerID, there are no more missing values in the data.

- Check the Data Structure (Summary of the data)

Summary of the data is as follows

```
> # Summary of retail data
> summary(retail_data)
  InvoiceNo          StockCode         Description          Quantity          InvoiceDate                    UnitPrice
 Length:406829      Length:406829      Length:406829      Min.   :-80995.00   Min.   :2010-12-01 08:26:00   Min.   :    0.00
 Class :character   Class :character   Class :character   1st Qu.:     2.00   1st Qu.:2011-04-06 15:02:00   1st Qu.:    1.25
 Mode  :character   Mode  :character   Mode  :character   Median :     5.00   Median :2011-07-31 11:48:00   Median :    1.95
                                                          Mean   :    12.06   Mean   :2011-07-10 16:30:57   Mean   :    3.46
                                                          3rd Qu.:    12.00   3rd Qu.:2011-10-20 13:06:00   3rd Qu.:    3.75
                                                          Max.   : 80995.00   Max.   :2011-12-09 12:50:00   Max.   :38970.00

   CustomerID      Country
 Min.   :12346   Length:406829
 1st Qu.:13953   Class :character
 Median :15152   Mode  :character
 Mean   :15288
 3rd Qu.:16791
 Max.   :18287
```

*Figure 5: summary of input data*

The minimum value of Quantity is identified as -80995.00. The value of Quantity can be negative only in the return transactions. As we won't be considering return transactions for our analysis, we will be filtering the return transactions out and consider only sales transactions.

The minimum unitPrice is identified as 0.00, which won't be the case in the sales transactions. Therefore, we will be considering the records, whose quantity is greater than 0.

```
> ## Remove unrelated and return transactions
> retail_data <- retail_data[retail_data$Quantity >= 0,]
> retail_data <- retail_data[retail_data$UnitPrice > 0,]
>
> summary(retail_data)
  InvoiceNo          StockCode         Description          Quantity         InvoiceDate                    UnitPrice
 Length:397884      Length:397884      Length:397884      Min.   :    1.00   Min.   :2010-12-01 08:26:00   Min.   :   0.001
 Class :character   Class :character   Class :character   1st Qu.:    2.00   1st Qu.:2011-04-07 11:12:00   1st Qu.:   1.250
 Mode  :character   Mode  :character   Mode  :character   Median :    6.00   Median :2011-07-31 14:39:00   Median :   1.950
                                                          Mean   :   12.99   Mean   :2011-07-10 23:41:23   Mean   :   3.116
                                                          3rd Qu.:   12.00   3rd Qu.:2011-10-20 14:33:00   3rd Qu.:   3.750
                                                          Max.   :80995.00   Max.   :2011-12-09 12:50:00   Max.   :8142.750

   CustomerID      Country
 Min.   :12346   Length:397884
 1st Qu.:13969   Class :character
 Median :15159   Mode  :character
 Mean   :15294
 3rd Qu.:16795
 Max.   :18287
```

*Figure 6: Filtering out return and Invalid transactions*

**(II) Transforming the data**

Recency, Frequency, Monetary (RFM) is a data-driven technique that is mostly used for customer segmentation. In this method, we use time from the last purchase (Recency), a total number of purchases in a period of time (Frequency) and total amount spent in a period of time (Monetary) attributes.

As a part of data Transformation, calculate Recency, Frequency, Monetary for every customer and form a new dataset.

Recency is the Number of days from the last purchase. For this, first we will calculate the time difference of every Invoice i.e., difference between max InvoiceDate and InvoiceDate. The minimum time difference of every customer is taken as the Recency.

```
> ## Recency
> # max Invoice Date
> max_date <- max(retail_data$InvoiceDate)
> max_date
[1] "2011-12-09 12:49:59 GMT"
>
> # Difference between the InvoiceDate and max_date
> retail_data$time_delta <- as.numeric(difftime(as.Date(max_date), as.Date(retail_data$InvoiceDate), units="days"))
> head(retail_data$time_delta, 2)
[1] 373 373
>
> # install.packages('dplyr')
> library(dplyr)
>
> # minimum time_delta (maxdate - recent InvoiceDate) of every customer
> Recency <- retail_data %>%
+               select(CustomerID, time_delta) %>%
+               group_by(CustomerID) %>%
+               slice(which.min(time_delta))
```

*Figure 7: Calculate Recency*

Frequency is the number of visits by the customer. To calculate this, we will group the data by CustomerId and get the count of Invoice number.

```
> # Frequency
> # Group the data by customerId and get the count of invoiceNo
> Frequency <- retail_data %>%
+                 select(CustomerID, InvoiceNo) %>%
+                 group_by(CustomerID) %>%
+                 summarize(count = n())
`summarise()` ungrouping output (override with `.groups` argument)
```

*Figure 8: Calculate Frequency*

Monetary is the total amount spent by the customer. To calculate this, we will group the data by CustomerId and get the sum of amount spent (UnitPrice * Quantity).

```
> ## Monetry
> # Group the data by CustomerID and get the sum of amount spent
> # Calculate total price i.e., quantity*unitprice
> retail_data$Total_Price <- (retail_data$Quantity)*(retail_data$UnitPrice)
>
> Monetry <- retail_data %>%
+                 select(CustomerID, Total_Price) %>%
+                 group_by(CustomerID) %>%
+                 summarize(Total = sum(Total_Price))
`summarise()` ungrouping output (override with `.groups` argument)
>
```

*Figure 9: Calculate Monetary*

Now merge Recency, Frequency and Monetary values by CustomerID and form new dataset.

```
> # merge Recency, Frequency and Monetary values by CustomerID
> RF <- merge(Recency, Frequency, by='CustomerID')
> RFM <- merge(RF, Monetry, by='CustomerID')
> colnames(RFM) <- c('CustomerID', 'Recency', 'Frequency', 'Monetary')
```

*Figure 10: Merge Recency, Frequency and Monetary values*

Structure and Summary of the RFM data is as below

```
> str(RFM)
'data.frame':   4338 obs. of  4 variables:
 $ CustomerID: num  12346 12347 12348 12349 12350 ...
 $ Recency   : num  325 2 75 18 310 36 204 232 214 22 ...
 $ Frequency : int  1 182 31 73 17 85 4 58 13 59 ...
 $ Monetary  : num  77184 4310 1797 1758 334 ...
>
> # Summary of the RFM data
> summary(RFM)
   CustomerID       Recency         Frequency          Monetary
 Min.   :12346   Min.   :  0.00   Min.   :   1.00   Min.   :     3.75
 1st Qu.:13813   1st Qu.: 17.00   1st Qu.:  17.00   1st Qu.:   307.42
 Median :15300   Median : 50.00   Median :  41.00   Median :   674.48
 Mean   :15300   Mean   : 92.06   Mean   :  91.72   Mean   :  2054.27
 3rd Qu.:16779   3rd Qu.:141.75   3rd Qu.: 100.00   3rd Qu.:  1661.74
 Max.   :18287   Max.   :373.00   Max.   :7847.00   Max.   :280206.02
```

*Figure 11: Structure and Summary of RFM data*

- Pre-processing the RFM data – Check for the outliers

```
> # Check for outliers
> boxplot(RFM[-1], main="Boxplot for RFM values")
```

*Figure 12: Check for outliers*

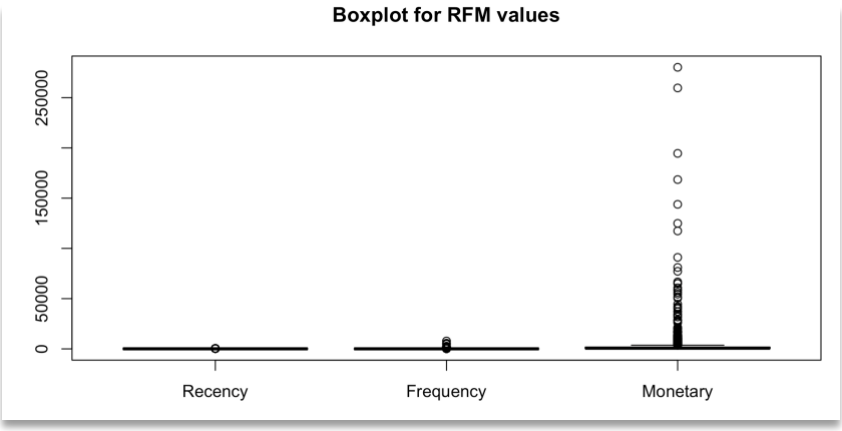


*Figure 13: Boxplot for RFM values*

From the boxplot, it is clear that Recency, Frequency and Monetary values have outliers.

For customer segmentation, we will be using clustering techniques.

Outliers will have more effect on clustering as they can change the centroids of the clusters. Therefore, we will be removing outliers as the pre-processing for clustering.

Remove Outliers

```
> # Remove outliers
> Recency_outliers <- boxplot(RFM$Recency, plot=FALSE)$out
> RFM_data <- RFM[-which(RFM$Recency %in% Recency_outliers),]
>
> Frequency_outliers <- boxplot(RFM$Frequency, plot=FALSE)$out
> RFM_data <- RFM[-which(RFM$Frequency %in% Frequency_outliers),]
>
> Monetary_outliers <- boxplot(RFM$Monetary, plot=FALSE)$out
> RFM_data <- RFM[-which(RFM$Monetary %in% Monetary_outliers),]
```

*Figure 14: Remove Outliers*

Structure of the RFM data after removing the outliers is as below

```
> str(RFM_data)
'data.frame':   3911 obs. of  4 variables:
 $ CustomerID: num  12348 12349 12350 12352 12353 ...
 $ Recency   : num  75 18 310 36 204 232 214 22 1 52 ...
 $ Frequency : int  31 73 17 85 4 58 13 59 19 129 ...
 $ Monetary  : num  1797 1758 334 2506 89 ...
```

*Figure 15: Structure of RFM data after removing outliers*

**(III) Standardizing the data**

In order to compare the data and draw conclusions, we need to scale the numerical data.

```
> # Scale the Numerical Variables
> vars <- c('Recency','Frequency','Monetary')
> Clust_Data <- data.frame(RFM_data[-1])
> Clust_Data[vars] <- lapply(RFM_data[vars], scale)
```

*Figure 16: Scaling the numerical variables*

The top rows of the new data (Clust_data) is as below

```
> head(Clust_Data)
     Recency   Frequency   Monetary
3 -0.2403706 -0.39472592  1.0990271
4 -0.8015356  0.14852823  1.0511234
5  2.0732046 -0.57581063 -0.6665412
6 -0.6243256  0.30374370  1.9545102
7  1.0296345 -0.74396072 -0.9627256
8  1.3052945 -0.04549111  0.2326333
```

*Figure 17: Head rows of clustering data*

### 4.1.2. Identify best 10 Association rules for the products in the online retail

This is to identify the Products that are frequently bought together in the online retail.

**(I) Cleaning the data**

- Check for the missing values

```
> # Check for the missing values
> colSums(is.na(data))
  InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice  CustomerID     Country
          0           0        1454           0           0           0      135080           0
```

*Figure 18: Check for missing values*

Data has missing values in Description and CustomerID columns.

As description (ProductName) is the main and only one attribute, we use for association rules and the number of missing values is small compared to the total size of the dataset i.e., 541909. We will be removing the records with Null values in Description.

As we won't be using customerID for this analysis i.e., while building association rules for the products that are brought together.

```
> # Remove rows with Description as Null
> rules_data <- data %>%
+              drop_na(Description)
```

*Figure 19: Remove the records having NULL in product name*

Summary of the data is as below

```
> summary(rules_data)
  InvoiceNo          StockCode         Description          Quantity         InvoiceDate
 Length:540455      Length:540455      Length:540455      Min.   :-80995.0   Min.   :2010-12-01 08:26:00
 Class :character   Class :character   Class :character   1st Qu.:     1.0   1st Qu.:2011-03-28 11:49:00
 Mode  :character   Mode  :character   Mode  :character   Median :     3.0   Median :2011-07-20 11:38:00
                                                          Mean   :     9.6   Mean   :2011-07-04 16:20:42
                                                          3rd Qu.:    10.0   3rd Qu.:2011-10-19 11:49:00
                                                          Max.   : 80995.0   Max.   :2011-12-09 12:50:00

   UnitPrice           CustomerID       Country
 Min.   :-11062.06   Min.   :12346   Length:540455
 1st Qu.:     1.25   1st Qu.:13953   Class :character
 Median :     2.08   Median :15152   Mode  :character
 Mean   :     4.62   Mean   :15288
 3rd Qu.:     4.13   3rd Qu.:16791
 Max.   : 38970.00   Max.   :18287
                     NA's   :133626
```

*Figure 20: Summary of the Input data*

The minimum value of Quantity is identified as -80995.00. The value of Quantity can be negative only in the return transactions. As we won't be considering return transactions for our analysis, we will be filtering the return transactions out and consider only sales transactions.

The minimum UnitPrice is identified as 0.00, which won't be the case in the sales transactions. Therefore, we will be considering the records, whose quantity is greater than 0.

```
> #remove the unrelated and return transactions
> rules_data <- rules_data[rules_data$Quantity >= 0,]
> rules_data <- rules_data[rules_data$UnitPrice > 0,]
```

*Figure 21: Filter out return and invalid transactions*

- Check for the presence of irrelevant data in Description (ProductName) attribute.

For this, as a first step, check whether the unique number of StockCodes is same as the unique number of products (Descriptions).

```
> # Check for the unique StockCode and Description
> length(unique(rules_data$StockCode))
[1] 3922
> length(unique(rules_data$Description))
[1] 4026
```

*Figure 22: Check for the length of unique StockCode and Description*

Generally, each stockCode represents a unique Product and a Product can have more than one StockCode as the StockCode number can be reviewed whenever the product is restoked (after the completion of all products of that StockCode in the shop).

In this case, there are a greater number of products than the stockCode. Therefore, we need to check whether all the Descriptions (ProductName) are valid or not.

Some of the descriptions in the data is as below

```
> head(rules_data$Description, 3)
[1] "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN"                "CREAM CUPID HEARTS COAT HANGER"
```

*Figure 23: Data Format of Description*

- Trim the Description (Product Name)

As Description is the type Character, we trim the data to remove any leading and trailing spaces.

```
> # trim the Description
> library(stringr)
> rules_data$Description <- str_trim(rules_data$Description)
```

*Figure 24: Trim the Description*

- Check for the special characters in the Description (Product Name)

```
> # Check for Special characters
> rules_data[grepl('[!#$%&*+;<=>?@[]^`|~]', rules_data$Description),]
[1] InvoiceNo   StockCode   Description Quantity    InvoiceDate UnitPrice   CustomerID  Country
<0 rows> (or 0-length row.names)
```

*Figure 25: Check for the special characters in Description*

- Check for the lowercase letters in Description

It seems Descriptions are in uppercase letters. Therefore, checking if there are any lowercase letters too in the Description.

```
> # Check for Description in lower case
> head(rules_data[grepl('[:lower:]', rules_data$Description),])
     InvoiceNo StockCode                       Description Quantity         InvoiceDate UnitPrice CustomerID
2060    536557     22686     FRENCH BLUE METAL DOOR SIGN No        1 2010-12-01 14:41:00      1.25      17841
2231    536569         M                            Manual        1 2010-12-01 15:35:00      1.25      16274
2242    536569         M                            Manual        1 2010-12-01 15:35:00     18.95      16274
2558    536592     21594      Dr. Jam's Arouzer Stress Ball        1 2010-12-01 17:06:00      4.21         NA
3296    536620     22965 3 TRADITIONAl BISCUIT CUTTERS  SET        6 2010-12-02 10:27:00      2.10      14135
3400    536626     22965 3 TRADITIONAl BISCUIT CUTTERS  SET        6 2010-12-02 10:46:00      2.10      13418
            Country
2060 United Kingdom
2231 United Kingdom
2242 United Kingdom
2558 United Kingdom
3296 United Kingdom
3400 United Kingdom
```

*Figure 26: Check for the lowercase letters in the Description*

There are many rows with manual as Descriptions (just shown few for reference). As product Name manual makes no sense, we will be removing the records with Manual in Description.

```
> # Remove the data with description as Manual
> rules_data <- rules_data[rules_data$Description != 'Manual',]
```

*Figure 27: Remove the data with Description as NULL*

- Check for the StockCodes with multiple products i.e, Descriptions

To achieve this, first we need to group the data by StockCode and Description and get the unique StockCode and Description. Then check for the duplicate StockCodes in the list of Unique StockCode and Description.

```
> # Identify the StockCodes having two descriptions and use the descriptions to process the multi descriptions
> df <- data.frame(rules_data %>%
+   select(StockCode, Description) %>%
+   group_by(StockCode, Description) %>%
+   count(StockCode, Description))
>
```

*Figure 28: Check for the StockCodes with multiple products*

List of the Descriptions of the duplicated StockCodes is as below.

```
> x <- df$StockCode[duplicated(df$StockCode)]
> sort(df$Description[df$StockCode %in% x])
  [1] "16 PC CUTLERY SET PANTRY DESIGN"     "16 PIECE CUTLERY SET PANTRY DESIGN"  "3 DRAWER ANTIQUE WHITE WOOD CABINET"
  [4] "3 TRADITIONAl BISCUIT CUTTERS  SET"  "3 TRADITIONAL COOKIE CUTTERS  SET"   "36 DOILIES DOLLY GIRL"
  [7] "36 DOILIES VINTAGE CHRISTMAS"        "50'S CHRISTMAS GIFT BAG LARGE"       "60 CAKE CASES VINTAGE CHRISTMAS"
 [10] "70'S ALPHABET WALL ART"              "72 CAKE CASES VINTAGE CHRISTMAS"     "ACRYLIC JEWEL SNOWFLAKE,PINK"
 [13] "ADULT APRON APPLE DELIGHT"           "ALUMINIUM HEART"                     "ALUMINIUM STAMPED HEART"
 [16] "ANIMALS AND NATURE WALL ART"         "ANTIQUE SILVER BOOK MARK WITH BEADS" "ANTIQUE SILVER T-LIGHT GLASS"
 [19] "ANTIQUE SILVER TEA GLASS ETCHED"     "APRON APPLE DELIGHT"                 "ASS COL CIRCLE MOBILE"
 [22] "ASSORTED COLOURED CIRCLE MOBILE"     "BAKING MOULD CHOCOLATE CUP CAKES"    "BAKING MOULD CHOCOLATE CUPCAKES"
 [25] "BAKING MOULD CUPCAKE CHOCOLATE"      "BAKING MOULD TOFFEE CUP  CHOCOLATE"  "BAKING MOULD TOFFEE CUP CHOCOLATE"
 [28] "BEADED CHANDELIER T-LIGHT HOLDER"    "BELL HEART ANTIQUE GOLD"             "BELL HEART DECORATION"
 [31] "BICYCLE SAFTEY WALL ART"             "BIRTHDAY BANNER TAPE"                "BIRTHDAY PARTY CORDON BARRIER TAPE"
 [34] "BISCUIT TIN VINTAGE LEAF"            "BLUE 3 PIECE POLKADOT CUTLERY SET"   "BLUE 3 PIECE POLKADOT CUTLERY SET"
 [37] "BLUE FELT HANGING HEART W FLOWER"    "BLUE FELT HANGING HEART WITH FLOWER" "BREAD BIN DINER STYLE IVORY"
 [40] "BREAD BIN DINER STYLE MINT"          "BREAD BIN, DINER STYLE, IVORY"       "BREAD BIN, DINER STYLE, MINT"
 [43] "BUFFALO BILL WALL ART"               "BUNDLE OF 3 RETRO EXERCISE BOOKS"    "BUNDLE OF 3 RETRO NOTE BOOKS"
 [46] "BUNTING , SPOTTY"                    "BUTTERFLY CUSHION COVER"             "CAKESTAND, 3 TIER, LOVEHEART"
 [49] "CANDLEHOLDER PINK HANGING HEART"     "CANNISTER VINTAGE LEAF DESIGN"       "CAT AND BIRD WALL ART"
 [52] "CHARLOTTE BAG ALPHABET  DESIGN"      "CHARLOTTE BAG VINTAGE ALPHABET"      "CHILDREN'S SPACEBOY MUG"
 [55] "CHILDRENS CUTLERY DOLLY GIRL"        "CHILDRENS CUTLERY POLKADOT BLUE"     "CHILDRENS CUTLERY POLKADOT BLUE"
 [58] "CHILDRENS CUTLERY POLKADOT GREEN"    "CHILDRENS CUTLERY POLKADOT GREEN"    "CHILDRENS CUTLERY POLKADOT PINK"
 [61] "CHILDRENS CUTLERY POLKADOT PINK"     "CHILDRENS CUTLERY RETROSPOT RED"     "CHILDRENS CUTLERY RETROSPOT RED"
 [64] "CHILDRENS CUTLERY SPACEBOY"          "CHILDRENS SPACEBOY MUG"              "CHRISTMAS GINGHAM HEART"
 [67] "CHRISTMAS HANGING HEART WITH BELL"   "CHRISTMAS MUSICAL ZINC HEART"        "CHRISTMAS RETROSPOT HEART WOOD"
 [70] "CHRISTMAS TABLE CANDLE SILVER SPIKE" "CHRISTMAS TABLE SILVER CANDLE SPIKE" "CLASSIC CAFE SUGAR DISPENSER"
 [73] "CLASSIC CHROME BICYCLE BELL"         "CLASSIC CROME BICYCLE BELL"          "CLASSIC GLASS COOKIE JAR"
 [76] "CLASSIC GLASS SWEET JAR"             "CLASSIC SUGAR DISPENSER"             "COLOUR GLASS. STAR T-LIGHT HOLDER"
 [79] "COLOURED GLASS STAR T-LIGHT HOLDER"  "CORDIAL GLASS JUG"                    "CORDIAL JUG"
 [82] "CREAM HANGING HEART T-LIGHT HOLDER"  "CRYSTAL CHANDELIER T-LIGHT HOLDER"   "DECORATION , WOBBLY CHICKEN, METAL"
 [85] "DECORATION , WOBBLY RABBIT , METAL"  "DECORATION HEN ON NEST, HANGING"     "DECORATION SITTING BUNNY"
 [88] "DECORATION WOBBLY CHICKEN"           "DECORATION WOBBLY RABBIT METAL"      "DECORATIVE VINTAGE COFFEE  BOX"
 [91] "DECROTIVEVINTAGE COFFEE GRINDER BOX" "DOG AND BALL WALL ART"               "DOG LICENCE WALL ART"
 [94] "DOILEY BISCUIT TIN"                  "DOILEY STORAGE TIN"                  "DOLLCRAFT BOY JEAN-PAUL"
 [97] "DOLLCRAFT GIRL AMELIE"               "DOLLCRAFT GIRL AMELIE KIT"           "DOLLCRAFT GIRL NICOLE"
[100] "DOLLY GIRL MINI BACKPACK"            "DOLLY GIRL MINI RUCKSACK"            "DOLLY GIRL WALL ART"
[103] "DONKEY TAIL GAME"                    "DOORKNOB CERAMIC IVORY"              "DOORKNOB CRACKED GLAZE BLUE"
[106] "DOORKNOB CRACKED GLAZE GREEN"        "DOORKNOB CRACKED GLAZE IVORY"        "DOORKNOB CRACKED GLAZE PINK"
[109] "DOORMAT VINTAGE LEAF"                "DOORMAT VINTAGE LEAVES DESIGN"       "DOUBLE CERAMIC PARLOUR HOOK"
[112] "DRAWER KNOB CERAMIC IVORY"           "DRAWER KNOB CRACKLE GLAZE BLUE"      "DRAWER KNOB CRACKLE GLAZE GREEN"
[115] "DRAWER KNOB CRACKLE GLAZE IVORY"     "DRAWER KNOB CRACKLE GLAZE PINK"      "EASTER DECORATION SITTING BUNNY"
[118] "ELEPHANT BIRTHDAY CARD"              "ELEPHANT, BIRTHDAY CARD,"            "ENAMEL BOWL PANTRY"
```

*Figure 29: List of Products of duplicated StockCodes*

Here, we got a list of 453 Descriptions (given few for reference). When we observer this list, there are many typo errors in the product names.

For example, “16 PC CUTLERY SET PANTRY DESIGN” and “16 PIECE CUTLERY SET PANTRY DESIGN” are same with same StockCode, but these will be treated as different by the data mining techniques and these will affect the support, confidence and lift values of our association rules.

Therefore, we need to treat these typo errors if the description has the same StockCode.

```
> rules_data$Description <- ifelse(rules_data$Description == "WALL ART BICYCLE SAFTEY", "WALL ART BICYCLE SAFETY",
+ ifelse(rules_data$Description == "WALL ART,ONLY ONE PERSON", "WALL ART ONLY ONE PERSON",
+ ifelse(rules_data$Description == "WHITE WIRE PLANT POT HOLDER", "WHITE HEARTS WIRE PLANT POT HOLDER",
+ ifelse(rules_data$Description == "WHITE METAL LANTERN", "WHITE MOROCCAN METAL LANTERN",
+ ifelse(rules_data$Description == "WOODLAND MINI RUCKSACK", "WOODLAND MINI BACKPACK",
+ ifelse(rules_data$Description == "WRAP RED DOILEY", "WRAP RED VINTAGE DOILY"  ,
+ ifelse(rules_data$Description == "WRAP VINTAGE PETALS  DESIGN", "WRAP VINTAGE LEAF DESIGN",
+ ifelse(rules_data$Description == "ZINC PLANT POT HOLDER" ,"ZINC HEARTS PLANT POT HOLDER",
+ ifelse(rules_data$Description == "ZINC  STAR T-LIGHT HOLDER", "ZINC STAR T-LIGHT HOLDER",
+ ifelse(rules_data$Description == "ZINC T-LIGHT HOLDER STAR LARGE", "ZINC T-LIGHT HOLDER STARS LARGE", rules_data$Description
 ))))))))))
```

*Figure 30: Correction of Typo errors in Description (ProductName)*

Treated almost 200 products which are having type errors (just a few are shown for reference, whole code will be available in the code file).

Now, the structure of the rules_data is as below

```
> str(rules_data)
'data.frame':   529782 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 MOROCCAN METAL LANTERN" "CREAM CUPID HEARTS COAT HANGER" "KNI
TTED UNION FLAG HOT WATER BOTTLE" ...
 $ Quantity   : num  6 6 8 6 6 2 6 6 6 32 ...
 $ InvoiceDate: POSIXct, format: "2010-12-01 08:26:00" "2010-12-01 08:26:00" "2010-12-01 08:26:00" "2010-12-01 08:26:00" ...
 $ UnitPrice  : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
 $ CustomerID : num  17850 17850 17850 17850 17850 ...
 $ Country    : chr  "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" ...
```

*Figure 31: Structure of data after pre-processing*

Check for the unique number of StockCodes and unique number of Product Names.

```
> length(unique(rules_data$StockCode))
[1] 3920
> length(unique(rules_data$Description))
[1] 3904
```

*Figure 32: Recheck the length of unique StockCode and Description*

Now, the counts look valid as a product can have multiple StockCodes and as we are using product name to get the frequent item sets. This difference won't be affecting our association rules.

### (II) Transforming the data

The granularity of our dataset is one line per item.

In order to know the products that are frequently bought together, we need to have all the items bought together in a single row i.e., the granularity of the dataset should be one row per Invoice.

Therefore, to get the items that are bought in a single invoice, we need to group the data by Invoice No and concatenate the products using ',.

```
> # Use ddply function to get all the items bought together in a row separated by ,.
> # To get the items bought together, get Description by grouping the data on InvoiceNo.
> Association_data <- ddply(rules_data,c("InvoiceNo"),
+                           function(x)paste(x$Description,
+                                            collapse = ","))
```

*Figure 33: Get the list of products bought together*

Structure of the association data is as below

```
> str(Association_data)
'data.frame':   19868 obs. of  2 variables:
 $ InvoiceNo: chr  "536365" "536366" "536367" "536368" ...
 $ V1       : chr  "WHITE HANGING HEART T-LIGHT HOLDER,WHITE MOROCCAN METAL LANTERN,CREAM CUPID HEARTS COAT HANGER,KNITTED UNI
ON FL"| __truncated__ "HAND WARMER UNION JACK,HAND WARMER RED POLKA DOT" "ASSORTED COLOUR BIRD ORNAMENT,POPPY'S PLAYHOUSE BEDR
OOM,POPPY'S PLAYHOUSE KITCHEN,FELTCRAFT PRINCESS CHARLOTTE "| __truncated__ "JAM MAKING SET WITH JARS,RED COAT RACK PARIS FASH
ION,YELLOW COAT RACK PARIS FASHION,BLUE COAT RACK PARIS FASHION" ...
```

*Figure 34: Structure of the data after transformation*

Remove the column InvoiceNo, as we won't be needing it in our process and change the column name from V1 to items.

```
> colnames(Association_data) <- c("items")
> Association_data$InvoiceNo <- NULL
```

*Figure 35: Remove column InvoiceNo*

In order to build the association rules, the attributes need to be the type of factor.

```
> Association_data$items <- as.factor(Association_data$items)
> str(Association_data)
'data.frame':   19868 obs. of  2 variables:
 $ items: Factor w/ 19868 levels "536365","536366",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ NA   : chr  "WHITE HANGING HEART T-LIGHT HOLDER,WHITE MOROCCAN METAL LANTERN,CREAM CUPID HEARTS COAT HANGER,KNITTED UNION F
L"| __truncated__ "HAND WARMER UNION JACK,HAND WARMER RED POLKA DOT" "ASSORTED COLOUR BIRD ORNAMENT,POPPY'S PLAYHOUSE BEDROOM,
POPPY'S PLAYHOUSE KITCHEN,FELTCRAFT PRINCESS CHARLOTTE "| __truncated__ "JAM MAKING SET WITH JARS,RED COAT RACK PARIS FASHION,
YELLOW COAT RACK PARIS FASHION,BLUE COAT RACK PARIS FASHION" ...
```

*Figure 36: Change the type of items to factor*

Association data contains the products that are bought together (i.e., in a single Invoice).

To view this data in better format, writing the association data to a csv file.

```
> # To view the transactions data in the better format
> write.csv(Association_data, "transactional_data.csv", quote=FALSE, row.names = FALSE)
```

*Figure 37: Write the transformed data to csv file*

The csv file looks like below

| | items |
|---|---|
| 2 | WHITE HANG WHITE MOR CREAM CUPI KNITTED UN RED WOOLL SET 7 BABUS GLASS STAR FROSTED T-LIGHT HOLDER |
| 3 | HAND WARN HAND WARMER RED POLKA DOT |
| 4 | ASSORTED C POPPY'S PLA POPPY'S PLA FELTCRAFT F IVORY KNITT BOX OF 6 AS BOX OF VINT BOX OF VINT HOME BUILT LOVE BUILDI RECIPE BOX DOORMAT NEW ENGLAND |
| 5 | JAM MAKINC RED COAT R/ YELLOW COA BLUE COAT RACK PARIS FASHION |
| 6 | BATH BUILDING BLOCK WORD |
| 7 | ALARM CLOC ALARM CLOC ALARM CLOC PANDA AND STARS GIFT INFLATABLE VINTAGE HE SET/2 RED R ROUND SNA SPACEBOY LI LUNCH BOX CIRCUS PAR/ CHARLOTTE RED TOADST SET 2 TEA TC VINTAGE SE/ MINI JIGSAW MINI JIGSAW MINI PAINT S |

*Figure 38: Glimpse of a csv file*

Each row contains all the products that are bought in a single Invoice.

## 4.2. Data Mining Methods and Processes

### 4.2.1. Identify the Customers into different groups based on their behavior with the online retail.

Clustering is a technique to group the set of objects such that the objects of the same group will have similar characteristics compared to the objects of other groups.

In our case, we need to divide the customers into different groups based on their behavior with the online retail that is, we need to have the customers with similar characteristics as a group. Therefore, Clustering will be the best solution to identify the customers into different groups (customer segmentation).

In this project, we use both K-means Clustering and Hierarchal clustering algorithms on RFM data for customer segmentation.

To build Clustering models, as a first step, we need to find an optimal number of clusters our data can be divided into.

In this project we use both Silhouette Method and Elbow to find optimal number of clusters that is K value.

**Silhouette Method**

Silhouette Method is the interpretation and validation of consistency within the clusters. This method calculates the silhouette coefficient for each point to measure how much a point is similar to its own cluster compared to other clusters.

Silhouette Score is average of silhouette coefficients of all data points.

The range of silhouette score is between [1, -1], where +1 indicates that the objects are well matched to its own cluster rather than with neighboring clusters.

Silhouette coefficient of $i^{th}$ point s(i) is calculated as below

$$s(i) = \frac{b(i) - a(i)}{\max\left(b(i), a(i)\right)}$$

Where a(i) is the average distance of $i^{th}$ point with all other points in the same cluster,

b(i) is the average distance of $i^{th}$ point with all other points in the closest cluster.

**Elbow Method**

Elbow method is used to determine the optimal number of clusters. In this method, a plot is drawn with the value of within-cluster-Sum of Squared Errors (WSS) function produced by different number of clusters and pick the elbow of the curve as the number of clusters to use.

The elbow is the optimal point after which, adding cluster does not worth the diminishing WSS.

WSS is the sum of squared errors in the cluster. Squared Error for a point is nothing but the square of the distance of the point to the centroid of the cluster. The distance between the point and centroid can be calculated using either Euclidean Distance or the Manhattan Distance.

Steps:

1. For each K-value, calculate the total WSS (within sum of squares of all clusters).
2. Plot the graph for K-value Vs total WSS
3. The value of K with a bend in the plot is considered as the appropriate K-value (clusters).

#### 4.2.1.1. K-Means Clustering

K-Means Clustering aims to divide the data into K partitions in which each data point belongs to the cluster with the nearest cluster centroid.

- Calculate optimal number of clusters using Silhouette method.

We calculate optimal number of clusters with K-means algorithm for the data using Silhouette method.

```
> # Silhouette Score
> set.seed(123)
> fviz_nbclust(Clust_Data, kmeans, method = "silhouette")
```

*Figure 39: Calculate Silhouette score with kmeans method*

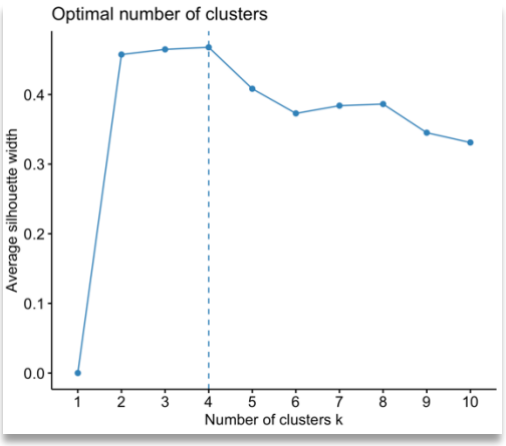


*Figure 40: Graph for Optimal number of clusters using silhouette method with kmeans*

The x-axis of this plot represents Number of clusters and y-axis of the plot represents Average Silhouette width.

From the above plot, it is clear that optimal number of clusters are 4 with highest average silhouette width.

- K-Means Clustering model with K as 4

As optimal number of clusters is 4 using Silhouette method, we will build K-Means clustering model with K as 4.

```
> # Compute K-Means with k as 4
> set.seed(101)
> KM_Model1 <- kmeans(Clust_Data, 4, nstart = 25)
> fviz_cluster(KM_Model1, data=Clust_Data,
+              geom="point",
+              ellipse.type = "convex",
+              ggtheme = theme_bw())
```

*Figure 41: KM_Model1 - KMeans model with K as 4*

Below is the graph of 4 clusters by K-means algorithm

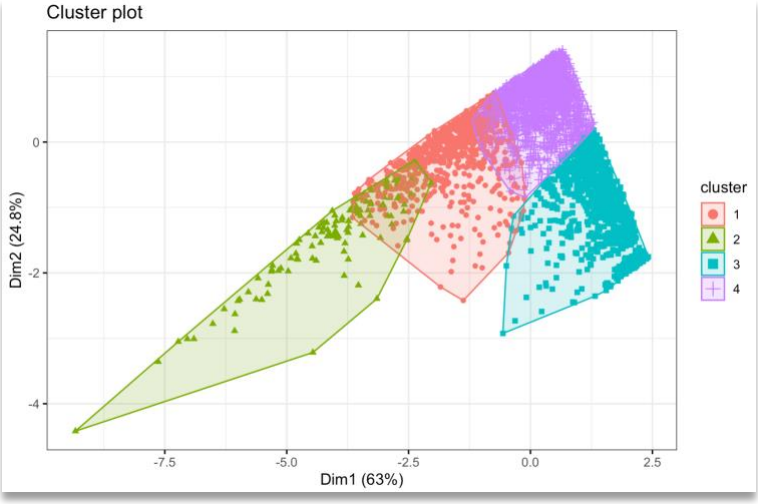


*Figure 42: KM_Model1 - Graph of clusters*

The total data was partitioned into 4 clusters based on their characteristics.

From the graph, the data points with different colors indicates the points in different clusters.

The size and centers of these clusters are as below

```
> # Size of Clusters
> KM_Model1$size
[1]  786  119  985 2021
> # Center of clusters
> KM_Model1$centers
     Recency  Frequency   Monetary
1 -0.5719567  0.7426322  1.4600967
2 -0.7496632  4.0453278  1.8326463
3  1.5205852 -0.4883279 -0.6109894
4 -0.4745218 -0.2890153 -0.3779794
```

*Figure 43: KM_Model1 - Size and Centers of Clsuters*

- Calculate optimal number of clusters using Elbow method.

We calculate optimal number of clusters with K-means algorithm for the data using Elbow method.

```
> # Elbow method
> set.seed(123)
> fviz_nbclust(Clust_Data, kmeans, method='wss')
```

*Figure 44: Calculate optimal number of clusters using Elbow Method with kmeans*

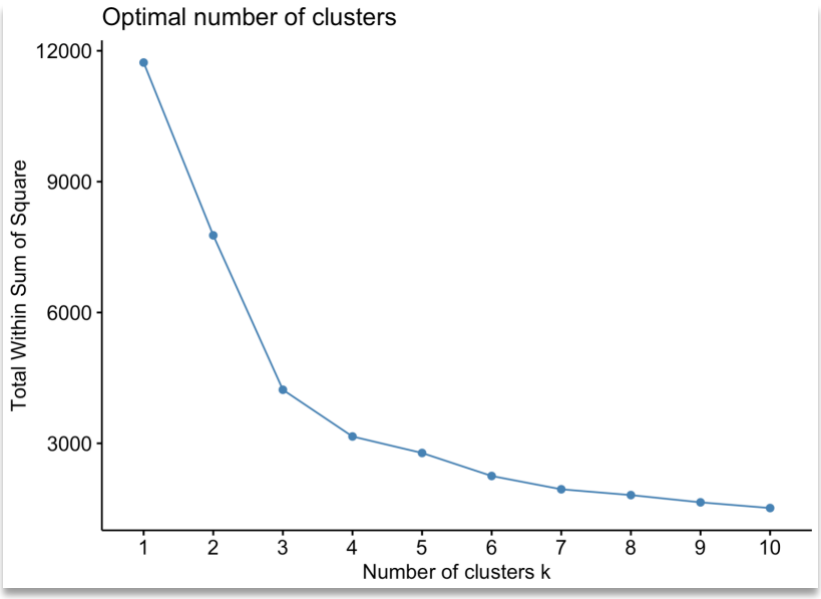


*Figure 45: Graph of optimal number of clusters using Elbow method with kmeans*

The x-axis of the above plot represents Number of clusters and the y-axis represents Total Within Sum of square.

In the above plot, the elbow of the curve is clearly at point 3. Therefore, the optimal number of clusters are 3 for data with k-means algorithm using elbow method.

- K-Means Clustering model with K as 3

As optimal number of clusters is 3 using Elbow method, we will build K-Means clustering model with K as 3.

```
> # Compute K-Means with k as 3
> set.seed(101)
> KM_Model2 <- kmeans(Clust_Data, 3, nstart = 25)
> # nstart=25 will generate 25 random centroids and choose the best one for algorithm.
>
> fviz_cluster(KM_Model2, data=Clust_Data,
+              geom="point",
+              ellipse.type = "convex",
+              ggtheme = theme_bw())
```

*Figure 46: KM_Model2 - KMeans model with K as 3*

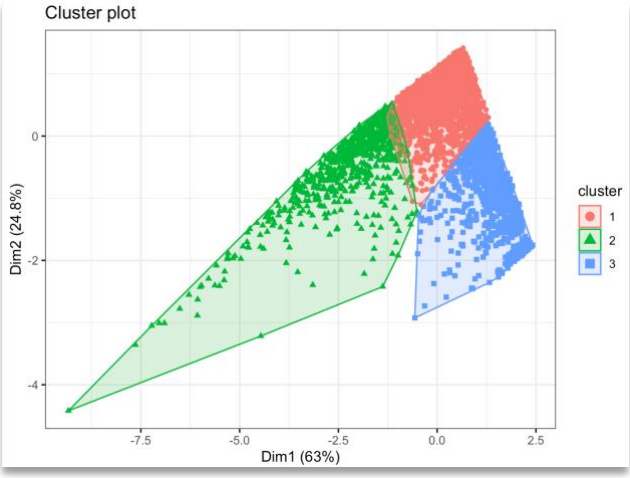


*Figure 47: KM_Model2 - Graph of Clusters*

The total data was partitioned into 3 clusters based on their characteristics using K-means algorithm.

From the graph, the data points with different colors indicates the points in different clusters.

The size and centers of these clusters are as below

```
> # Size of Clusters
> KM_Model2$size
[1] 2154  758  999
> # Center of clusters
> KM_Model2$centers
     Recency  Frequency   Monetary
1 -0.4781634 -0.2581498 -0.3034692
2 -0.6310794  1.3699627  1.6535819
3  1.5098321 -0.4828600 -0.6003427
```

*Figure 48: KM_Model2: Size and Centers of Cluster*

### 4.2.1.2. Hierarchical Clustering

Hierarchical Clustering is only of the clustering techniques that builds the hierarchy of clusters.

- Calculate optimal number of clusters using Silhouette method.

We calculate optimal number of clusters with hierarchical algorithm for the data using Silhouette method.

```
> # Silhouette Score
> set.seed(123)
> fviz_nbclust(Clust_Data, hcut, method = "silhouette")
```

*Figure 49: Calculate Silhouette score with hcut method*

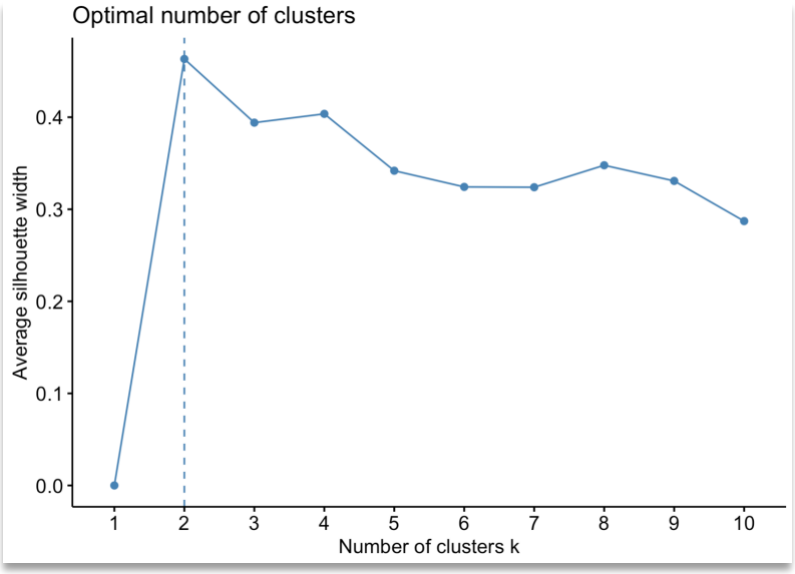


*Figure 50: Graph for Optimal number of clusters using silhouette method with hcut*

The x-axis of this plot represents Number of clusters and y-axis of the plot represents Average Silhouette width.

From the above plot, it is clear that optimal number of clusters are 2 with highest average silhouette width.

- Hierarchical Clustering model with K as 2

As optimal number of clusters is 2 using Silhouette method, we will build Hierarchical clustering model with K as 2.

```
> # calculate distance between vectors of Clust_Data
> d <- dist(Clust_Data, method='euclidean')
> HC_model1 <- hclust(d, method='ward.D2')
>
> # Dendrogram, customizing the plot to remove labels
> HC_model1_d <- as.dendrogram(HC_model1)
> nodePar <- list(lab.cex = 0.6, pch = c(NA, 19),
+                 cex = 0.2, col = "skyblue")
> plot(HC_model1_d, ylab = "Height", nodePar = nodePar, leaflab = "none",
+      main = "Dendrogram with K=2")
> rect.hclust(HC_model1, k=2, border="green")
```

*Figure 51: HC_Model1 - Hierarchical model with K as 2*

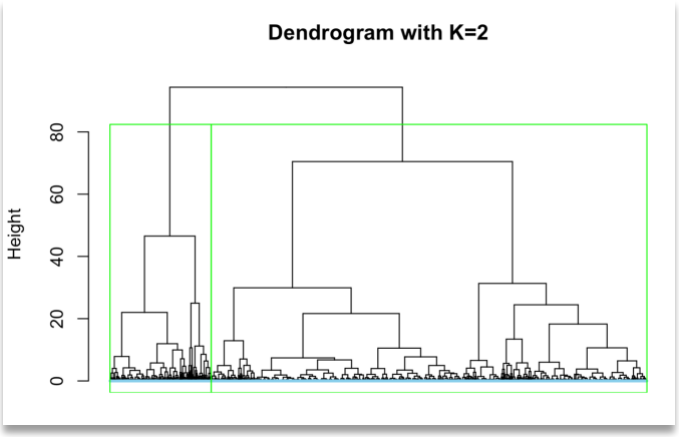


*Figure 52: HC_Model1 - Dendrogram*

The total data was partitioned into 2 clusters based on their characteristics using hierarchical clustering algorithm.

In the above dendrogram, total data is partitioned in two hierarchical clusters (green line indicates the partition).

The size and centers of the clusters is as below

```
> # size of groups
> HC_data1 %>%
+   group_by(groups2) %>%
+   summarise(count=n())
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 2 x 2
  groups2 count
    <int> <int>
1       1  3174
2       2   737
> # Centers of groups
> apply (Clust_Data, 2, function (x) tapply (x, groups2, mean))
     Recency  Frequency   Monetary
1  0.1508643 -0.2854663 -0.4000176
2 -0.6497195  1.2294028  1.7227353
```

*Figure 53: HC_Model1 - Size and Centers of clusters*

- Calculate optimal number of clusters using Elbow method.

We calculate optimal number of clusters with hierarchical algorithm for the data using Elbow method.

```
> # Elbow method
> set.seed(123)
> fviz_nbclust(Clust_Data, hcut, method='wss')
```

*Figure 54: Calculate optimal number of clusters using Elbow Method with hcut*

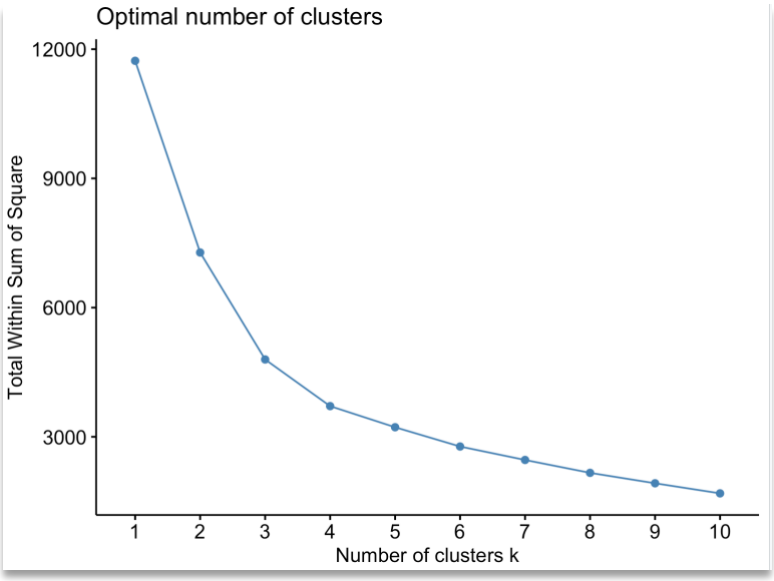


*Figure 55: Graph of optimal number of clusters using Elbow method with hcut*

The x-axis of the above plot represents Number of clusters and the y-axis represents Total Within Sum of square.

In the above plot, the elbow of the curve is clearly at point 3. Therefore, the optimal number of clusters are 3 for data with hierarchical algorithm using elbow method.

- Hierarchical Clustering model with K as 3

As optimal number of clusters is 3 using Elbow method, we will build Hierarchical clustering model with K as 3.

```
> HC_model2 <- hclust(d, method='ward.D2')
>
> # Dendrogram, customizing the plot to remove labels
> HC_model2_d <- as.dendrogram(HC_model1)
> nodePar <- list(lab.cex = 0.6, pch = c(NA, 19),
+                 cex = 0.2, col = "skyblue")
> plot(HC_model2_d, ylab = "Height", nodePar = nodePar, leaflab = "none",
+      main = "Dendrogram with K=3")
> rect.hclust(HC_model2, k=3, border="green")
>
```

*Figure 56: HC_Model2 - Hierarchical model with k as 3*

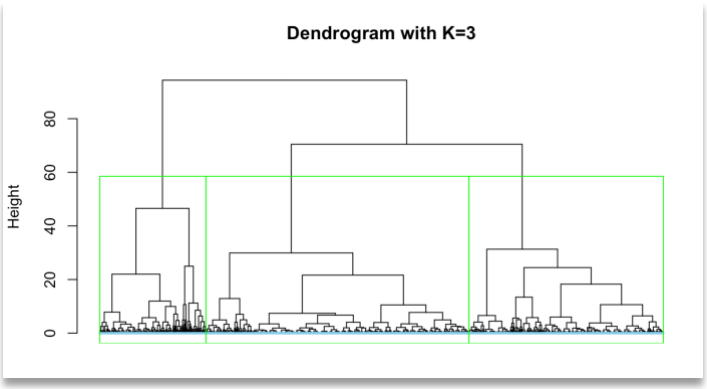


*Figure 57: HC_Model2 - Dendrogram*

The total data was partitioned into 3 clusters based on their characteristics using hierarchical clustering algorithm.

In the above dendrogram, total data is partitioned in 3 hierarchical clusters (green line indicates the partition).

The size and centers of the clusters is as below

```
> groups3 <- cutree(HC_model2, k=3)
>
> HC_data2 <- data.frame(Clust_Data, groups3)
> head(HC_data2, 1)
     Recency  Frequency Monetary groups3
3 -0.2403706 -0.3947259 1.099027       1
>
> # size of groups
> HC_data2 %>%
+   group_by(groups3) %>%
+   summarise(count=n())
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 3 x 2
  groups3 count
    <int> <int>
1       1  1349
2       2   737
3       3  1825
>
> # Centers of groups
> apply (Clust_Data, 2, function (x) tapply (x, groups3, mean))
     Recency  Frequency   Monetary
1  1.1628303 -0.4485738 -0.4882853
2 -0.6497195  1.2294028  1.7227353
3 -0.5971588 -0.1649007 -0.3347721
```

*Figure 58: HC_Model2 - Size and Centers of cluster*

### 4.2.2. Identify best 10 Association rules for the products in the online retail

We will be using Association rules mining technique in order to know the products that are frequently bought together.

To generate association rules, the data should be in transactions format. Therefore, read the data from csv file as transactions.

```
> transactional_data <- read.transactions('transactional_data.csv', format = 'basket', sep=',', quote="")
Warning message:
In asMethod(object) : removing duplicated items in transactions
```

*Figure 59: Read the data as transactions*

- Generate association rules using Apriori Algorithm

Apriori Algorithm is using to find frequent item sets and their relevant association rules.

```
> # Association rules
> rules.1 <- apriori(transactional_data, parameter=list(support=0.1, confidence = 0.1))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.1    0.1    1 none FALSE            TRUE       5     0.1      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 1986

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[3938 item(s), 19869 transaction(s)] done [0.16s].
sorting and recoding items ... [3 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [3 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 60: Association rules with min.support 0.1 and min.confidence 0.1*

With minimum support of 0.1 and minimum confidence of 0.1, 3 rules have been generated.

```
> rules.05 <- apriori(transactional_data, parameter=list(support=0.05, confidence = 0.1))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.1    0.1    1 none FALSE            TRUE       5    0.05      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 993

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[3938 item(s), 19869 transaction(s)] done [0.17s].
sorting and recoding items ... [34 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [3 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 61: Association rules with min.support 0.05 and min.confidence 0.1*

With minimum support of 0.05 and minimum confidence of 0.1, 3 association rules have been generated for the products bought together.

```
> rules.029 <- apriori(transactional_data, parameter=list(support=0.0293, confidence = 0.1, minlen=2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.1    0.1    1 none FALSE            TRUE       5  0.0293      2     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 582

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[3938 item(s), 19869 transaction(s)] done [0.20s].
sorting and recoding items ... [139 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [20 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 62: Association rules with min.support 0.0293 and min.confidence 0.1*

With minimum support of 0.0293 and minimum confidence of 0.1, 20 association rules have been generated for the products bought together.

Inspect the 20 association rules ordered by confidence

```
> inspect(sort(rules.029, by='confidence'))
     lhs                                    rhs                                    support    confidence coverage   lift      count
[1]  {PINK REGENCY TEACUP AND SAUCER}    => {GREEN REGENCY TEACUP AND SAUCER}    0.03180834 0.8261438  0.03850219 16.203999 632
[2]  {PINK REGENCY TEACUP AND SAUCER}    => {ROSES REGENCY TEACUP AND SAUCER}    0.03009714 0.7816993  0.03850219 14.583647 598
[3]  {GREEN REGENCY TEACUP AND SAUCER}   => {ROSES REGENCY TEACUP AND SAUCER}    0.03860285 0.7571570  0.05098394 14.125776 767
[4]  {ROSES REGENCY TEACUP AND SAUCER}   => {GREEN REGENCY TEACUP AND SAUCER}    0.03860285 0.7201878  0.05360109 14.125776 767
[5]  {JUMBO BAG PINK POLKADOT}           => {JUMBO BAG RED RETROSPOT}           0.04152197 0.6773399  0.06130152  6.442349 825
[6]  {ALARM CLOCK BAKELIKE GREEN}        => {ALARM CLOCK BAKELIKE RED}          0.03221098 0.6530612  0.04932307 12.346026 640
[7]  {JUMBO  BAG BAROQUE BLACK WHITE}    => {JUMBO BAG RED RETROSPOT}           0.02944285 0.6270096  0.04695757  5.963645 585
[8]  {GREEN REGENCY TEACUP AND SAUCER}   => {PINK REGENCY TEACUP AND SAUCER}    0.03180834 0.6238894  0.05098394 16.203999 632
[9]  {JUMBO STORAGE BAG SUKI}            => {JUMBO BAG RED RETROSPOT}           0.03643867 0.6114865  0.05959032  5.816000 724
[10] {ALARM CLOCK BAKELIKE RED}          => {ALARM CLOCK BAKELIKE GREEN}        0.03221098 0.6089439  0.05289647 12.346026 640
[11] {JUMBO SHOPPER VINTAGE RED PAISLEY} => {JUMBO BAG RED RETROSPOT}           0.03422417 0.5787234  0.05913735  5.504383 680
[12] {ROSES REGENCY TEACUP AND SAUCER}   => {PINK REGENCY TEACUP AND SAUCER}    0.03009714 0.5615023  0.05360109 14.583647 598
[13] {LUNCH BAG PINK POLKADOT}           => {LUNCH BAG RED RETROSPOT}           0.03049977 0.5559633  0.05485933  7.058425 606
[14] {LUNCH BAG  BLACK SKULL.}           => {LUNCH BAG RED RETROSPOT}           0.03226131 0.5035350  0.06406966  6.392803 641
[15] {LUNCH BAG RED RETROSPOT}           => {LUNCH BAG  BLACK SKULL.}           0.03226131 0.4095847  0.07876592  6.392803 641
[16] {JUMBO BAG RED RETROSPOT}           => {JUMBO BAG PINK POLKADOT}           0.04152197 0.3949258  0.10513866  6.442349 825
[17] {LUNCH BAG RED RETROSPOT}           => {LUNCH BAG PINK POLKADOT}           0.03049977 0.3872204  0.07876592  7.058425 606
[18] {JUMBO BAG RED RETROSPOT}           => {JUMBO STORAGE BAG SUKI}            0.03643867 0.3465773  0.10513866  5.816000 724
[19] {JUMBO BAG RED RETROSPOT}           => {JUMBO SHOPPER VINTAGE RED PAISLEY} 0.03422417 0.3255146  0.10513866  5.504383 680
[20] {JUMBO BAG RED RETROSPOT}           => {JUMBO  BAG BAROQUE BLACK WHITE}    0.02944285 0.2800383  0.10513866  5.963645 585
```

*Figure 63: Inspect Association rules*

If we observe the above association rules, some rules are redundant.

For example, we have association rules

{GREEN REGENCY TEACUP AND SAUCER} => {ROSES REGENCY TEACUP AND SAUCER}

{ROSES REGENCY TEACUP AND SAUCER} => {GREEN REGENCY TEACUP AND SAUCER}

These both rules indicate that the products GREEN REGENCY TEACUP AND SAUCER and ROSES REGENCY TEACUP AND SAUCER are frequently bought together. So, there is no point in having two same rules. In order to avoid these types of redundant rules and get valid rules, we will prune redundant rules

- Prune the redundant rules

We prune the redundant rules by checking is any rule, a subset of other rules.

```
> rules <- sort(rules.029, by='confidence')
> # Check for the rules which are subset of other rules
> subset_matrix <- is.subset(rules,rules)
>
> # Assigning FALSE to diagonal line (as every rule is a subset of it's own, neglect diagonal positions)
> subset_matrix[lower.tri(subset_matrix, diag=T)] <- F
>
> # Check the item sets which are redudant and which are not
> redundant <- apply(subset_matrix, 2, any)
>
> rules.pruned <- rules[!redundant]
> inspect(rules.pruned)
     lhs                                    rhs                                    support    confidence coverage   lift      count
[1]  {PINK REGENCY TEACUP AND SAUCER}    => {GREEN REGENCY TEACUP AND SAUCER}    0.03180834 0.8261438  0.03850219 16.203999 632
[2]  {PINK REGENCY TEACUP AND SAUCER}    => {ROSES REGENCY TEACUP AND SAUCER}    0.03009714 0.7816993  0.03850219 14.583647 598
[3]  {GREEN REGENCY TEACUP AND SAUCER}   => {ROSES REGENCY TEACUP AND SAUCER}    0.03860285 0.7571570  0.05098394 14.125776 767
[4]  {JUMBO BAG PINK POLKADOT}           => {JUMBO BAG RED RETROSPOT}           0.04152197 0.6773399  0.06130152  6.442349 825
[5]  {ALARM CLOCK BAKELIKE GREEN}        => {ALARM CLOCK BAKELIKE RED}          0.03221098 0.6530612  0.04932307 12.346026 640
[6]  {JUMBO  BAG BAROQUE BLACK WHITE}    => {JUMBO BAG RED RETROSPOT}           0.02944285 0.6270096  0.04695757  5.963645 585
[7]  {JUMBO STORAGE BAG SUKI}            => {JUMBO BAG RED RETROSPOT}           0.03643867 0.6114865  0.05959032  5.816000 724
[8]  {JUMBO SHOPPER VINTAGE RED PAISLEY} => {JUMBO BAG RED RETROSPOT}           0.03422417 0.5787234  0.05913735  5.504383 680
[9]  {LUNCH BAG PINK POLKADOT}           => {LUNCH BAG RED RETROSPOT}           0.03049977 0.5559633  0.05485933  7.058425 606
[10] {LUNCH BAG  BLACK SKULL.}           => {LUNCH BAG RED RETROSPOT}           0.03226131 0.5035350  0.06406966  6.392803 641
```

*Figure 64: Prune the Redundant rules*

Finally, we have got 10 valid association rules with minimum support 0.0293 and minimum confidence 0.1.

# 5. Evaluations and Results

## 5.1. Evaluation Methods

There are no clear evaluations for unsupervised learning techniques. Clustering is good as long as it can serve the usage. Association rules are good as long as they have much support and we can interpret the results.

### 5.1.1. Identify the Customers into different groups based on their behavior with the online retail.

For clustering models, we have two main evaluation measures to validate the results.

1. Internal measures and
2. External measures

Internal measures evaluate the quality of clusters based on the data.

External measures evaluate the quality of clusters using external knowledge i.e., class labels.

#### 5.1.1.1. Internal measures

There are three Internal measures they are Connectivity, silhouette width and Dunn Index

**Connectivity:** Degree of connectedness of the clusters determined by KNN. This should be minimum.

**Silhouette width:** Measures the compactness of the clusters.

$$\boldsymbol{S_i = \frac{(b_i - a_i)}{max(a_i, b_i)}}$$

$a_i$ is the average distance of object i from all other objects of same cluster.

$b_i$ is the average distance of object i from all other objects of different cluster.

The range of $S_i$ is [-1,1].

**Dunn Index:** Measures the separation of clusters.

$$\boldsymbol{D = \frac{min.separation}{max.diameter}}$$

This should be maximum.

We calculate internal measures for the clustering models using both kmeans and hierarchical methods for values of k 2,3 and 4.

```
> # Internal
> internal <- clValid(as.matrix(Clust_Data), nClust = 2:4,
+                     clMethods = c("hierarchical","kmeans"),
+                    validation = "internal")

The number of items to be clustered is larger than 'maxitems'
The memory and time required may be excessive, do you wish to continue?
(y to continue, any other character to exit)
y
```

*Figure 65: Calculate Internal measures*

```
> summary(internal)

Clustering Methods:
 hierarchical kmeans

Cluster sizes:
 2 3 4

Validation Measures:
                                   2        3        4

hierarchical Connectivity    3.0290  15.8480  26.2401
             Dunn            0.2803   0.0540   0.0540
             Silhouette      0.8232   0.6360   0.4251
kmeans       Connectivity  153.5032 255.6778 276.3798
             Dunn            0.0024   0.0031   0.0041
             Silhouette      0.4574   0.4648   0.4688

Optimal Scores:

             Score  Method       Clusters
Connectivity 3.0290 hierarchical 2
Dunn         0.2803 hierarchical 2
Silhouette   0.8232 hierarchical 2
```

*Figure 66: Summary of Internal measures*

Connectivity, Dunn Index and Silhouette width are calculated for all the clustering models build using Kmeans and hierarchical algorithms for values of K 2,3 and 4.

Comparing all the internal measure values of all the models, we have got optimal scores for hierarchical clustering model with K as 2.

Using Internal validation measures, we can conclude that for the given data the hierarchical clustering model with two clusters is the best fit.

### 5.1.1.2. External measures

As the data is not pre-labeled, we cannot use External validation measures like Accuracy by implementing Classification techniques.

### 5.1.1.3. Manual Evaluation

- Compare the centroids of clusters.

We can compare the centroids of clusters and check whether the centroids are significantly different from each other

```
> # Centers of clusters from K-means Model with K= 4
> KM_Model1$centers
     Recency  Frequency   Monetary
1 -0.5719567  0.7426322  1.4600967
2 -0.7496632  4.0453278  1.8326463
3  1.5205852 -0.4883279 -0.6109894
4 -0.4745218 -0.2890153 -0.3779794
```

*Figure 67: KM_Model1 - Centers of Clusters*

Comparing the centers of KM_Model1 (Kmeans clustering model with K as 4) clusters, there is no significant difference between the Recency values of cluster1, cluster2 and cluster4. There is also no significant difference between the Frequency values of cluster3 and cluster4.

```
> # Centers of clusters from K-means Model with K= 3
> KM_Model2$centers
     Recency  Frequency   Monetary
1 -0.4781634 -0.2581498 -0.3034692
2 -0.6310794  1.3699627  1.6535819
3  1.5098321 -0.4828600 -0.6003427
>
```

*Figure 68: KM_Model2 - Centers of Clusters*

Comparing the centers of KM_Model2 (Kmeans clustering model with K as 3) clusters, there is no significant difference between the Recency values of cluster1, cluster. There is also no significant difference between the Frequency values of cluster1 and cluster3.

```
> # Centers of clusters from Hierarchical clustering with K = 2
> apply (Clust_Data, 2, function (x) tapply (x, groups2, mean))
     Recency  Frequency   Monetary
1  0.1508643 -0.2854663 -0.4000176
2 -0.6497195  1.2294028  1.7227353
```

*Figure 69: HC_Model1 - Centers of Clusters*

Comparing the centers of HC_Model1(Hierarchical clustering model with K as 2) clusters, there is significant difference between the Recency, Frequency and Monetary values of cluster1 and cluster2.

```
> # Centers of clusters from Hierarchical clustering with K = 3
> apply (Clust_Data, 2, function (x) tapply (x, groups3, mean))
     Recency  Frequency   Monetary
1  1.1628303 -0.4485738 -0.4882853
2 -0.6497195  1.2294028  1.7227353
3 -0.5971588 -0.1649007 -0.3347721
>
```

*Figure 70: HC_Model2 - Centers of Clusters*

Comparing the centers of HC_Model2(Hierarchical clustering model with K as 3) clusters, there is no significant difference between the Recency values of cluster2 and cluster3. There is also no significant difference between Monetary values of cluster1 and cluster3.

Therefore, both from Internal validation measures and Manual evaluations, Hierarchical Clustering model with clusters 2 is the best model.

### 5.1.2. Identify best 10 Association rules for the products in the online retail

Association rules can be evaluated with the Rule evaluation metrics i.e., Support, Confidence and lift.

For association rule $\{\boldsymbol{X}\} \rightarrow \{\boldsymbol{Y}\}$

**Support**

Support measures how frequent the product set is in all the transactions.

$$\boldsymbol{Support}\ (\{\boldsymbol{X}\} \rightarrow \{\boldsymbol{Y}\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ Transactions}$$

**Confidence**

Confidence measures the likeliness of buying product Y, having product X in the Basket.

$$\boldsymbol{Confidence}\ (\{\boldsymbol{X}\} \rightarrow \{\boldsymbol{Y}\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

**Lift**

Lift is the rise in probability of buying product Y with product X in the basket over the probability of buying product Y without having the product X in the basket.

$$\boldsymbol{Lift}\ (\{\boldsymbol{X}\} \rightarrow \{\boldsymbol{Y}\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transcations\ containing\ Y}$$

Top 10 Association rules with their Support, Confidence and Lift values is as below

```
> # We evaluate the association rules with support, confidence, lift values
> interestMeasure(rules.pruned, c("support", "confidence", "lift"), transactional_data)
      support confidence      lift
1  0.03180834  0.8261438 16.203999
2  0.03009714  0.7816993 14.583647
3  0.03860285  0.7571570 14.125776
4  0.04152197  0.6773399  6.442349
5  0.03221098  0.6530612 12.346026
6  0.02944285  0.6270096  5.963645
7  0.03643867  0.6114865  5.816000
8  0.03422417  0.5787234  5.504383
9  0.03049977  0.5559633  7.058425
10 0.03226131  0.5035350  6.392803
```

*Figure 71: Association rules - Support, Confidence, Lift values*

In the given data 0.04 is the minimum support (best for the data), with which we can generate association rules.

The confidence of the top association rule is also good with 82.6%.

Lift value close to 1 indicates X and Y almost often appear together as expected, greater than 1 indicates, X and Y appear together more than expected and less than 1 indicates, X and Y appear less than expected. Greater lift values indicate stronger association.

The lift value of all the association rules is greater than 1 which indicates X and Y appear together more than expected. Expected is nothing but the Fraction of transactions containing only Y.

All the rules are generated with more than 50% confidence, lift value more than 1 and with minimum support 3% (best we got for the given data). We are considering these rules are good fit.

## 5.2. Results and Findings

### 5.2.1. Identify the Customers into different groups based on their behavior with the online retail.

- Interpreting the results

From the above evaluations, hierarchical clustering model with 2 clusters is the best model for customer segmentation of an online retail.

Divide the customers into 2 different groups using hierarchical clustering model.

The below are the size of the clusters.

```
> RFM_data['Cluster'] <- HC_data1$groups2
>
> Final_data <- data.frame(RFM_data)
>
> # Size of the clusters
> Final_data %>%
+   group_by(Cluster) %>%
+   summarise(count=n())
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 2 x 2
  Cluster count
    <int> <int>
1       1  3174
2       2   737
```

*Figure 72: Size of the final clusters*

Summary of the final data is as below

```
> summary(Final_data)
   CustomerID       Recency         Frequency         Monetary          Cluster
 Min.   :12348   Min.   :  0.00   Min.   :  1.00   Min.   :   3.75   Min.   :1.000
 1st Qu.:13868   1st Qu.: 22.00   1st Qu.: 15.00   1st Qu.: 283.51   1st Qu.:1.000
 Median :15348   Median : 58.00   Median : 35.00   Median : 588.22   Median :1.000
 Mean   :15338   Mean   : 99.42   Mean   : 61.52   Mean   : 886.65   Mean   :1.188
 3rd Qu.:16806   3rd Qu.:157.00   3rd Qu.: 79.00   3rd Qu.:1246.21   3rd Qu.:1.000
 Max.   :18287   Max.   :373.00   Max.   :970.00   Max.   :3692.28   Max.   :2.000
```

*Figure 73: Summary of Final data*

Some of the customers from group 1

```
> # Interpreting the results
> customers_1 <- head(RFM_data$CustomerID[RFM_data$Cluster == 1])
> filter(RFM_data, RFM_data$CustomerID %in% customers_1)
  CustomerID Recency Frequency Monetary Cluster
1      12348      75        31  1797.24       1
2      12350     310        17   334.40       1
3      12353     204         4    89.00       1
4      12354     232        58  1079.40       1
5      12355     214        13   459.40       1
6      12358       1        19  1168.06       1
>
```

*Figure 74: Customers of Group1*

For customer1, Recency and Frequency value is less whereas Monetary value is more compared to their respective means.

For customer2, Recency value is more whereas Frequency and Monetary values are less compared to their respective means.

From the above data, we can infer that Cluster 1 is of the customers who are either with More Recency or Less Frequency or Less Monetary in the online Retail.

Some of the customers from group 2

```
> customers_2 <- head(RFM_data$CustomerID[RFM_data$Cluster == 2])
> filter(RFM_data, RFM_data$CustomerID %in% customers_2)
  CustomerID Recency Frequency Monetary Cluster
1      12349      18        73  1757.55       2
2      12352      36        85  2506.04       2
3      12356      22        59  2811.43       2
4      12360      52       129  2662.06       2
5      12370      51       167  3545.69       2
6      12371      44        63  1887.96       2
```

*Figure 75: Customers of Group2*

For all the above customers Recency value is less, Frequency is more and Monetary is more compared to their respective means. Therefore, these are the customers who visited recently, frequently and spent good amount to the online retail.

Therefore, Group 2 customers are identified as Loyal customers.

## 5.2.2. Identify best 10 Association rules for the products in the online retail

The best 10 association rules for the products in the online retail are as below

```
> inspect(sort(rules.pruned, by='confidence'))
     lhs                                  rhs                                 support    confidence coverage   lift      count
[1]  {PINK REGENCY TEACUP AND SAUCER}  => {GREEN REGENCY TEACUP AND SAUCER} 0.03180834 0.8261438  0.03850219 16.203999 632
[2]  {PINK REGENCY TEACUP AND SAUCER}  => {ROSES REGENCY TEACUP AND SAUCER} 0.03009714 0.7816993  0.03850219 14.583647 598
[3]  {GREEN REGENCY TEACUP AND SAUCER} => {ROSES REGENCY TEACUP AND SAUCER} 0.03860285 0.7571570  0.05098394 14.125776 767
[4]  {JUMBO BAG PINK POLKADOT}         => {JUMBO BAG RED RETROSPOT}         0.04152197 0.6773399  0.06130152  6.442349 825
[5]  {ALARM CLOCK BAKELIKE GREEN}      => {ALARM CLOCK BAKELIKE RED}        0.03221098 0.6530612  0.04932307 12.346026 640
[6]  {JUMBO  BAG BAROQUE BLACK WHITE}  => {JUMBO BAG RED RETROSPOT}         0.02944285 0.6270096  0.04695757  5.963645 585
[7]  {JUMBO STORAGE BAG SUKI}          => {JUMBO BAG RED RETROSPOT}         0.03643867 0.6114865  0.05959032  5.816000 724
[8]  {JUMBO SHOPPER VINTAGE RED PAISLEY} => {JUMBO BAG RED RETROSPOT}       0.03422417 0.5787234  0.05913735  5.504383 680
[9]  {LUNCH BAG PINK POLKADOT}         => {LUNCH BAG RED RETROSPOT}         0.03049977 0.5559633  0.05485933  7.058425 606
[10] {LUNCH BAG  BLACK SKULL.}         => {LUNCH BAG RED RETROSPOT}         0.03226131 0.5035350  0.06406966  6.392803 641
```

*Figure 76: Best 10 Association rules*

Above are the association rules with min.support 0.0293 and min.confidence 0.1.

The support which we got is the best for the given data.

The confidence of the top association rule is good with 82.6%.

The lift value of all the association rules is greater than 1 which indicates X and Y appear together more than expected. Expected is nothing but the Fraction of transactions containing only Y.

The visual graph for the association rules is as below

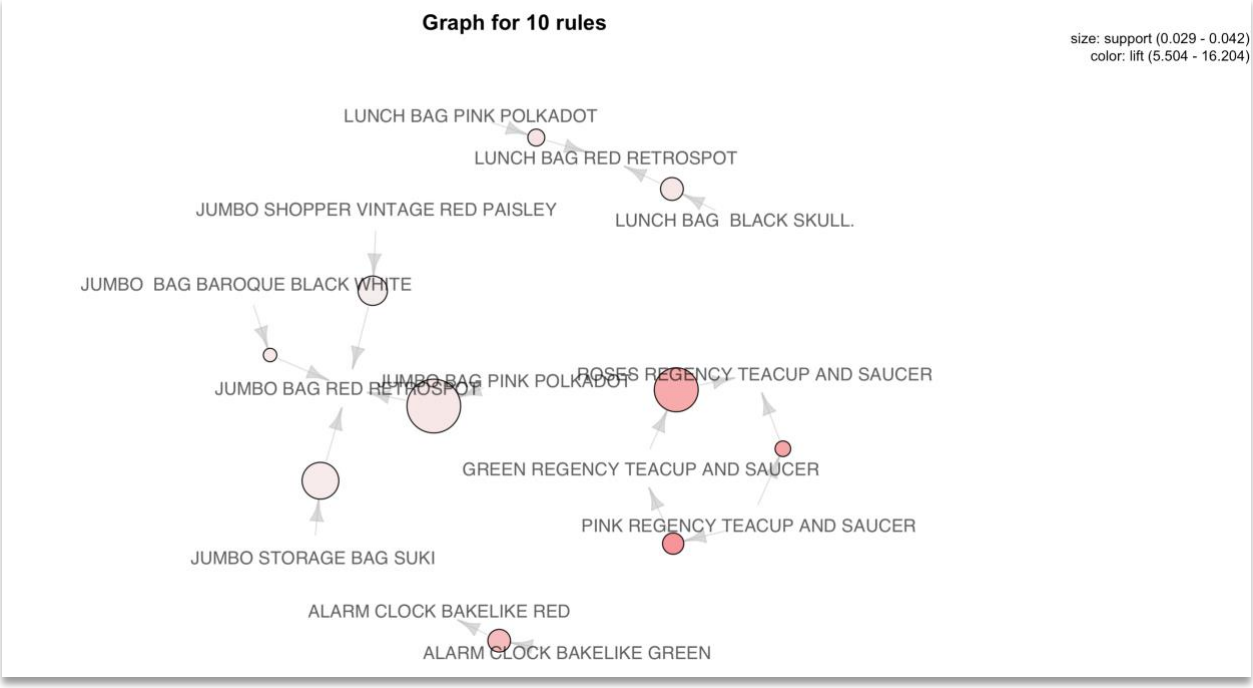


*Figure 77: Graph of Best 10 Association rules*

# 6. Conclusions and Future Work

## 6.1. Conclusions

- Customer Segmentation

Customers of online-retail have been segmented into two groups using their Recency, Frequency and Monetary values.

Group 1 are with the customers who are either not recent nor frequent nor spent good amount to the online retail.

Group 2 are the customers who are recent, frequent and spent good amount to the online retail.

Group 2 customers are identified as Loyal Customers.

- Market Basket Analysis

Following are the products that bought together along with their association ordered by confidence.

| | |
|---|---|
| {PINK REGENCY TEACUP AND SAUCER} | => {GREEN REGENCY TEACUP AND SAUCER} |
| {PINK REGENCY TEACUP AND SAUCER} | => {ROSES REGENCY TEACUP AND SAUCER} |
| {GREEN REGENCY TEACUP AND SAUCER} | => {ROSES REGENCY TEACUP AND SAUCER} |
| {JUMBO BAG PINK POLKADOT} | => {JUMBO BAG RED RETROSPOT} |
| {ALARM CLOCK BAKELIKE GREEN} | => {ALARM CLOCK BAKELIKE RED} |
| {JUMBO BAG BAROQUE BLACK WHITE} | => {JUMBO BAG RED RETROSPOT} |
| {JUMBO STORAGE BAG SUKI} | => {JUMBO BAG RED RETROSPOT} |
| {JUMBO SHOPPER VINTAGE RED PAISLEY} | => {JUMBO BAG RED RETROSPOT} |
| {LUNCH BAG PINK POLKADOT} | => {LUNCH BAG RED RETROSPOT} |
| {LUNCH BAG  BLACK SKULL.} | => {LUNCH BAG RED RETROSPOT} |

## 6.2. Limitations

- The conclusions made for the online retail are limited to the transactions that occurred between 01/12/2010 and 09/12/2011.
- The customer segmentation is limited to K-Means and Hierarchical clustering algorithms.
- The calculation of optimal number of clusters is limited to Elbow method and silhouette method.

## 6.3. Potential Improvements or Future Work

- Use Gap-Statistic Method to find the optimal number of clusters for clustering.
- Use Density based and K-medoids clustering algorithms for Customer segmentation.
- Inspect Association rules with confidence levels 0.5, 0.6 and so...Business can use these rules in designing lossless promotional offers for the products.