“Customer Segmentation with RFM Analysis and K-Means Clustering using the Online Retail Dataset"

1) Introduction:

The motivation behind the "Customer Segmentation with RFM Analysis and K-Means Clustering using the Online Retail Dataset" project is to use k-means clustering, an unsupervised algorithm, to gain insights into customer behavior and preferences, and group them into segments based on their purchasing behavior. The dataset used in this project is comprised of transactional data from an online retailer based in the UK between 2010 and 2011, which includes information on product descriptions, quantities sold, customer IDs, and purchase dates.
In summary, the project's motivation is to use k-means clustering techniques to gain insights into customer behavior and preferences, segment customers based on their purchasing habits, and develop targeted marketing strategies that can ultimately improve customer retention and loyalty for the online retailer.

The goal of the project is to analyze the purchasing behavior of each customer using the k-means clustering algorithm using three key metrics: frequency, recency, and monetary value. The aim is to identify different customer segments by grouping similar customers together based on their purchasing behavior. This analysis can then be used to develop targeted marketing strategies, improve customer retention, and increase sales and profitability for the retailer.
Overall, the goals of this project are to use data mining techniques to gain insights into customer behavior, identify opportunities for growth and optimization, and ultimately improve the profitability and sustainability of the online retailer.

This data mining project using the Online Retail dataset is interesting and important for several reasons:
Real-world application: This project uses a real-world dataset from an online retailer, which makes it relevant and applicable to businesses that want to gain insights into their customer behavior and sales patterns.
Business impact: By using data mining techniques, we can identify opportunities for growth and optimization that can ultimately have a positive impact on the retailer's profitability and sustainability.
Customer-centric approach: The project focuses on understanding the needs and preferences of customers, which is essential for developing effective marketing strategies and improving customer satisfaction and loyalty.
This project is interesting and important because it offers an opportunity to gain practical experience in data mining, demonstrate the ability to apply data-driven insights to real-world problems, and showcase skills that are valuable in a variety of industries.

Applying data mining techniques is necessary for the Online Retail dataset project because it is a large and complex dataset that contains a wealth of information about customer behavior and sales patterns. The dataset has information on over 5,41,909 transactions from customers across different countries, and it includes details on product descriptions, unit prices, and purchase dates.

By applying data mining techniques such as clustering, we can extract valuable insights and knowledge from the dataset that can help us optimize pricing strategies on the product based on their purchase, improve inventory management, and tailor marketing campaigns to specific customer groups. The information generated can then be used to develop effective marketing strategies for products to certain groups of customers for increasing sales and improve customer satisfaction. So, the application of the data mining technique is essential for extracting insights and value from the Online Retail dataset, which can ultimately lead to more informed decisions and improved business outcomes.

Keywords:

Customer Segmentation, K-Means Clustering, Purchasing Behavior, Targeted Marketing Strategies, Customer Retention, Real-world Application, Business Impact, Customer-centric Approach, Data-driven Insights, Clustering, Pricing Strategies, Inventory Management, Marketing Campaigns, Effective Marketing Strategies, Informed Decisions, Improved Business Outcomes.

2) Background:

The Online Retail dataset contains transactional data from an online retailer that specializes in selling gift items. The company operates mainly in the UK but also sells to other European countries. The dataset contains approximately 541,909 transactions made between 01/12/2010 and 09/12/2011.

The overall message of this project will be to analyze the transactional data and extract insights that can inform the company's business decisions. The information that needs to be conveyed includes:

Sales patterns: Understanding the sales patterns can help the company make decisions about inventory management, pricing strategies, and marketing campaigns.

Customer behavior**:** Analyzing customer behavior can help the company identify customer segments, tailor marketing campaigns, and improve customer retention.

Repeat purchases**:** Identifying customers who make repeat purchases can help the company increase revenue and develop strategies to encourage repeat purchases.

Business outcomes**:** The overall goal of the project is to identify sales patterns and insights that can help the company for better business decisions and improve its business.

The research question or problem to be addressed in this project can be as follows:

- Who are the best customers?
- Which of the customers could contribute to your churn rate?
- Who has the potential to become valuable customers?
- Which year has the highest transition of the product?
- Which month, days have the highest total sales in comparison?

The available dataset is sufficient to deliver the intended message for the above project. The dataset contains transactional data for one year from an online retailer, which can be analyzed to identify patterns and insights that can inform business decisions and improve revenue and the business can plan to avoid customer churn. The dataset includes information on customer demographics, products purchased, transaction dates, and transaction amounts, which are all relevant to the analysis.

3 . Dataset description:

The source of this dataset is from one of the public data repositories - kaggle
https://www.kaggle.com/datasets/vijayuv/onlineretail

The Online Retail dataset contains transactional data of a UK-based online retailer between 01/12/2010 and 09/12/2011. It includes about 5,41,909 transactions made by about 4373 customers and covers 4224 unique products. Each transaction record includes the Customer ID, Invoice Number, Product Description, Quantity, Invoice Date, Unit Price, and Country. We will use this dataset to segment the customer on the basis of Regency, frequency, and monetary (RFM) using K-means clustering.

For a data mining project, this dataset can be used to gain insights into customer behavior, trends in product sales, and patterns in transaction data. This can help businesses make informed decisions about marketing strategies, inventory management, and customer retention. For a data analyst or data scientist, this dataset can provide an opportunity to develop skills in data preprocessing, feature selection, feature engineering, and data visualization, data modeling. It will help me to gain experience in these things and in the future, the experience I gained using this dataset will help me to get success as a data scientist or data analyst. This dataset also be used to explore and apply various machine learning algorithms such as classification and association rule mining.

Now, we will explore the dataset step by step:

Loading dataset in Rstudio:

```
> #Loading dataset
> setwd("/Users/nirajkc/Desktop/module 7")
> retail <- read.csv("OnlineRetail.csv")
>
```

First, the working directory is set then, the dataset OnlineRetail.csv is loaded into the object named retail.

Loading required libraries:

```
> # • Loading require library.
> library(ggplot2)  # popular data visualization package
> library(dplyr)  #for data manipulation functions- mutate(),filter(),group_by(),summarize(), arrange()
>
```

The above R libraries ggplot2 is a powerful package for data visualization and
dplyr is a widely used package for data manipulation, it provides functions to select, filter, and group data.

printing top five rows of dataset:

```
> head(retail)
  InvoiceNo StockCode                          Description Quantity    InvoiceDate UnitPrice CustomerID
1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6 12/1/2010 8:26      2.55      17850
2    536365     71053                 WHITE METAL LANTERN        6 12/1/2010 8:26      3.39      17850
3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8 12/1/2010 8:26      2.75      17850
4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6 12/1/2010 8:26      3.39      17850
5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6 12/1/2010 8:26      3.39      17850
6    536365     22752        SET 7 BABUSHKA NESTING BOXES        2 12/1/2010 8:26      7.65      17850
         Country
1 United Kingdom
2 United Kingdom
3 United Kingdom
4 United Kingdom
5 United Kingdom
6 United Kingdom
>
```

head() function lists the top 5 rows of the dataset.

checking dimension dataset

```
> # •checking dimension  dataset
> dim(retail)
[1] 541909      8
>
```

By using the dim() function, we can list the number of rows and columns. The retail dataset contains 541909 rows and 8 columns.

structure of dataset:

```
> # •structure of dataset
> str(retail)
'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 UN
ION FLAG HOT WATER BOTTLE" ...
 $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
 $ InvoiceDate: chr  "12/1/2010 8:26" "12/1/2010 8:26" "12/1/2010 8:26" "12/1/2010 8:26" ...
 $ 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" ...
>
```

The "Online Retail" dataset contains transactional data from an online retailer based in the UK. The dataset includes 541,909 records of transactions that occurred between 01/12/2010 and 09/12/2011. Each transaction consists of the following variables:

InvoiceNo: A unique identifier for each transaction.
StockCode: A unique identifier for each product.
Description: A description of the product.
Quantity: The quantity of each product purchased in each transaction.
InvoiceDate: The date and time that each transaction occurred.
UnitPrice: The price of each product.
CustomerID: A unique identifier for each customer.
Country: The country where the customer resides.

All the variables were characters except Quantity, UnitPrice, and CustomerID, where UnitPrice was numerical whereas quantity and CustomerID were int variables.

#Total unique Products and Unique Customer

```
> #Total unique Product & Unique Customers
> length(unique(retail$Description))
[1] 4224
> length(unique(retail$CustomerID))
[1] 4373
```

To find the count of unique elements, we use the length() and unique() function in R, from the above code, we can see that the unique total customers were 4373 and unique products that were purchased were 4224.

Summary of the dataset:

```
> # •Summary of quantity and UnitPrice
> summary(retail$Quantity)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-80995.00      1.00      3.00      9.55     10.00  80995.00
> summary(retail$UnitPrice)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-11062.06      1.25      2.08      4.61      4.13  38970.00
>
```

The summary() function is used for numerical variables, it provides summary statistics which includes minimum, maximum, first quartile, median, mean, and third quartile for the numerical data.
The mean value for the Quantity and UnitPrice variables is more than the median, which states that the distribution of the data is right-skewed, also there are zero and negative values for Quantity and UnitPrice, there are several reasons why the Quantity and UnitPrice columns in the online retail dataset might contain negative values: may be due to cancellations and returns, or due to data entry errors, or may be due to discounts and promotions, but we will replace those values during the data cleaning process.

Frequency of unique countries:

```
> my_table <- table(retail$Country)
> my_sorted_table <- sort(my_table, decreasing = TRUE)
> my_sorted_table

      United Kingdom              Germany               France                EIRE                Spain 
              495478                 9495                 8557                 8196                 2533 
         Netherlands              Belgium          Switzerland             Portugal            Australia 
                2371                 2069                 2002                 1519                 1259 
              Norway               Italy      Channel Islands              Finland               Cyprus 
                1086                  803                  758                  695                  622 
              Sweden         Unspecified              Austria              Denmark                Japan 
                 462                  446                  401                  389                  358 
              Poland              Israel                  USA            Hong Kong            Singapore 
                 341                  297                  291                  288                  229 
             Iceland              Canada               Greece               Malta United Arab Emirates 
                 182                  151                  146                  127                   68 
  European Community                 RSA              Lebanon           Lithuania               Brazil 
                  61                   58                   45                   35                   32 
      Czech Republic              Bahrain         Saudi Arabia 
                  30                   19                   10 
> 
```

table() function is used to count the frequency of character variable, the above r code sorted the frequency of country in descending order. The United Kingdom has the highest frequency of occurrence, followed by Germany, then France.

Frequency of Description of product in descending order:

```
> # •Frequency of unique Description in descending order:
> my_table1 <- table(retail$Description)
> my_sorted_table1 <- sort(my_table1, decreasing = TRUE)
> my_sorted_table1

 WHITE HANGING HEART T-LIGHT HOLDER           REGENCY CAKESTAND 3 TIER            JUMBO BAG RED RETROSPOT
                               2369                               2200                               2159
                      PARTY BUNTING            LUNCH BAG RED RETROSPOT       ASSORTED COLOUR BIRD ORNAMENT
                               1727                               1638                               1501
  SET OF 3 CAKE TINS PANTRY DESIGN                                      PACK OF 72 RETROSPOT CAKE CASES
                               1473                               1454                               1385
            LUNCH BAG  BLACK SKULL.     NATURAL SLATE HEART CHALKBOARD                            POSTAGE
                               1350                               1280                               1252
           JUMBO BAG PINK POLKADOT              HEART OF WICKER SMALL           JAM MAKING SET WITH JARS
                               1251                               1237                               1229
            JUMBO STORAGE BAG SUKI     PAPER CHAIN KIT 50'S CHRISTMAS    JUMBO SHOPPER VINTAGE RED PAISLEY
                               1214                               1210                               1202
               LUNCH BAG CARS BLUE         LUNCH BAG SPACEBOY DESIGN             JAM MAKING SET PRINTED
                               1197                               1192                               1182
    RECIPE BOX PANTRY YELLOW DESIGN                    SPOTTY BUNTING            LUNCH BAG SUKI DESIGN
                               1180                               1172                               1139
   ROSES REGENCY TEACUP AND SAUCER            LUNCH BAG PINK POLKADOT   WOODEN PICTURE FRAME WHITE FINISH
                               1138                               1137                               1129
       SET OF 4 PANTRY JELLY MOULDS         ALARM CLOCK BAKELIKE RED      GREEN REGENCY TEACUP AND SAUCER
                               1111                               1107                               1085
            LUNCH BAG APPLE DESIGN     VICTORIAN GLASS HANGING T-LIGHT        RED RETROSPOT CHARLOTTE BAG
                               1084                               1072                               1068
                LUNCH BAG WOODLAND                RABBIT NIGHT LIGHT         ALARM CLOCK BAKELIKE GREEN
                               1061                               1051                               1024
SET/20 RED RETROSPOT PAPER NAPKINS                   JUMBO BAG APPLES        WOODEN FRAME ANTIQUE WHITE
                               1015                               1009                                999
```

The above code shows the frequency of the highest-selling product in descending order. The top selling product is a white hanging heart t-light holder followed by regency cakestand 3 tier.

Following are the steps I did for cleaning and customization in order to fit the k-means clustering. The dataset contains null values, which were removed. In this dataset, we have analyzed the recency, frequency, and monetary value of customers. All these steps are as follows:

#Checking Null Values

```
> #Checking Null values
> colSums(is.na(retail))
  InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice  CustomerID     Country
          0           0           0           0           0           0      135080           0
>
```

We can check the null values in the dataset using is.na() function and CustomerID contains 135080 null values. It is a good practice to remove null values as it can help ensure the accuracy of analysis and modeling, prevent errors in data visualization, and improve overall data quality.

Removing null values:

```
> # Removing null values
> retail <- na.omit(retail)
> colSums(is.na(retail))
  InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice  CustomerID     Country
          0           0           0           0           0           0           0           0
>
```

In R, we can remove null values using na.omit() finction.

Filtering UnitPrice & Quantity value > 0:

```
> # Removing UnitPrice & Quantity value <= 0
> retail <- retail %>% filter(UnitPrice > 0, Quantity >0)
> min(retail$Quantity)
[1] 1
> min(retail$UnitPrice)
[1] 0.001
>
```

This will create a new dataset called "retail", using filter() function from the dplyr library, that only contains rows where both the "UnitPrice" and "Quantity" values are greater than 0.

It is a good practice to remove any negative or invalid values from a dataset, including the "UnitPrice" and "Quantity" columns in the "retail" dataset. This is because negative values do not make sense for these variables and may indicate errors in the data entry process or other issues.

checking unique rows of dataset:

```
> # checking unique rows of dataset
> dim(unique(retail))[1]
[1] 392692
>
```

The above code will show the number of unique rows in the dataset. As seen above, there are 392692 unique rows in the dataset after removing the negative and zero values for price and quantity.

separate date, month, year, daysOfWeek column from InvoiceDate:

```
> # separate date, month, year, daysOfWeek column from InvoiceDate
> retail$InvoiceDate <- as.POSIXct(retail$InvoiceDate, format="%m/%d/%Y %H:%M") #converting string InvoiceDate to date and time format
>
> library(lubridate)#for manipulating dates & times
> retail$month <- month(retail$InvoiceDate)
> retail$year <- year(retail$InvoiceDate)
> retail$dayOfWeek <- wday(retail$InvoiceDate, label=TRUE)
>
```

The first line of code converts the string data type of InvoiceDate to date time format using as.POSIXct() function,
And extracting month, year, and days using the month(), year(), and wday() functions respectively from using the lubridate library.

Change into factors for newly created column using as.factor() function:

```
> # Change into factors  for newly created column using as.factor() function:
> retail$month <- as.factor(retail$month)
> retail$dayOfWeek <- as.factor(retail$dayOfWeek)
> retail$year <- as.factor(retail$year)
> retail$Country <- as.factor(retail$Country)
> str(retail)
'data.frame':   397884 obs. of  11 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 HO
T WATER BOTTLE" ...
 $ Quantity   : int  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 : int  17850 17850 17850 17850 17850 17850 17850 17850 17850 13047 ...
 $ Country    : Factor w/ 37 levels "Australia","Austria",..: 35 35 35 35 35 35 35 35 35 35 ...
 $ month      : Factor w/ 12 levels "1","2","3","4",..: 12 12 12 12 12 12 12 12 12 12 ...
 $ year       : Factor w/ 2 levels "2010","2011": 1 1 1 1 1 1 1 1 1 1 ...
 $ dayOfWeek  : Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 4 4 4 4 4 4 4 4 4 4 ...
 - attr(*, "na.action")= 'omit' Named int [1:135080] 623 1444 1445 1446 1447 1448 1449 1450 1451 1452 ...
  ..- attr(*, "names")= chr [1:135080] "623" "1444" "1445" "1446" ...
>
```

In R, it is often more efficient to store categorical variables as factors rather than as strings, because factors use less memory and are faster to process, also when building models or performing statistical analyses, it is often necessary to convert categorical variables to factors, because many statistical methods require numeric input, and factors can be useful for creating plots and visualizations, because they can be easily mapped to colors, shapes, and other graphical elements.

From the above codes, we have converted columns : month, dayOfWeek, year, and Country into factors using as.factor() function.

Creating new column Total Sales by multiplying Quantity*UnitPrice

```
> #Creating new column Total Sales by multiplying Quantity*UnitPrice
> retail = mutate(retail, TotalSales = Quantity*UnitPrice)
>
```

In the above code, mutate() function is used to create a new column in dataset retail by multiplying quantity and unitprice.

creating Diff column (by subtracting latest date minus InvoiceDate column)

```
> # creating Diff column (by subtracting latest date minus InvoiceDate column)
> max_date <- max(retail$InvoiceDate) # max date
> retail$Diff =  difftime(max_date, retail$InvoiceDate, units = "days")
> retail$Diff <- floor(retail$Diff)
>
```

In the above code difftime() is a built-in R function that calculates the difference between two dates, floor() is a built-in function that rounds a numeric value down to the nearest integer.
The first line of code calculate the maximum date or latest date from InvoiceDate column and "Diff" column indicates how many days it has been customer has purchased the product.

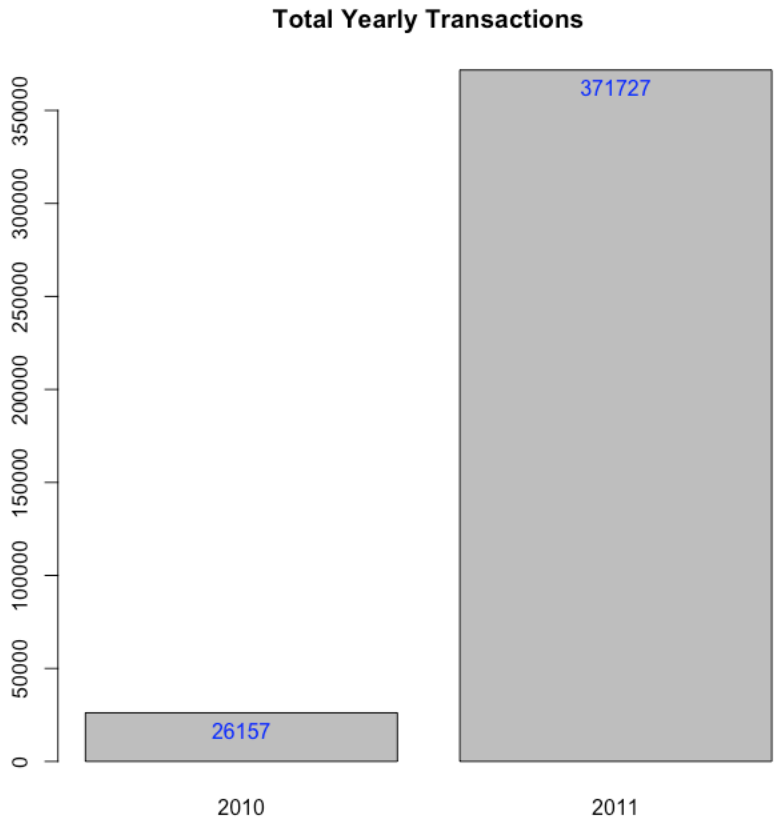
#RFM (Recency, Frequency, Monetary) analysis:
RFM analysis is a data analysis technique used to segment customers based on their purchasing behavior. It stands for:
Recency: how recently a customer has made a purchase.
Frequency: how often a customer makes purchases.
Monetary Value: how much a customer spends on each purchase.

```
> # RFM (Recency, Frequency, Monetary) analysis:
> RFM <- summarise(group_by(retail,CustomerID),Frequency = n(), Monetary = sum(TotalSales), Recency = min(Diff))
> RFM$Recency <- as.numeric(RFM$Recency)
> summary(RFM)
   CustomerID      Frequency          Monetary            Recency
 Min.   :12346   Min.   :   1.00   Min.   :     3.75   Min.   :  0.00
 1st Qu.:13813   1st Qu.:  17.00   1st Qu.:   307.42   1st Qu.: 17.00
 Median :15300   Median :  41.00   Median :   674.48   Median : 50.00
 Mean   :15300   Mean   :  91.72   Mean   :  2054.27   Mean   : 91.61
 3rd Qu.:16779   3rd Qu.: 100.00   3rd Qu.:  1661.74   3rd Qu.:141.00
 Max.   :18287   Max.   :7847.00   Max.   :280206.02   Max.   :373.00
>
```

The first code groups the retail dataset by CustomerID by using groupby(), and then uses the summarize() function from the dplyr package to calculate the Frequency (number of purchases), Monetary (total spending), and Recency (number of days since the last purchase) for each customer, sum() is used to calculate the total monetary value of purchases for each customer, and min() is used to calculate the recency (number of days since last purchase) for each customer, and also we have converted recency column into numeric from datetime. The results are stored in a new data frame called RFM.

Now, visualizing the results:
Transactions By Year Analysis

```
> # Transactions By Year Analysis
> year_counts <- table(retail$year)
> barplot(year_counts,
+         main = "Yearly transactions",
+         xlab = "Year",
+         ylab = "Total number",
+         col = "blue")
> text(x = barplot(year_counts),y = year_counts,labels = year_counts,pos = 1, col = "blue")
> title("Total Yearly Transactions")
>
```

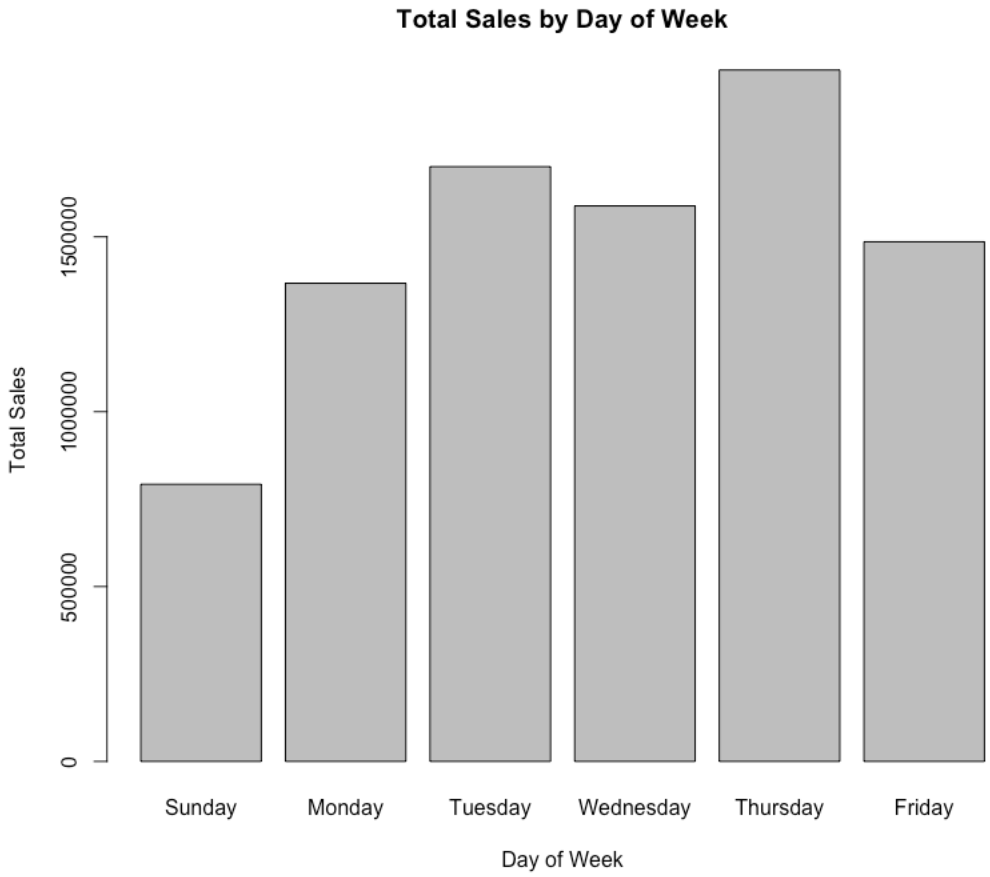


The above code plots bar chart of the total transition made in the year 2010 and 2011, we can see that the total transition in 2011 is much higher.

Revenue By day of the week Analysis:

```
> # Aggregate TotalSales by dayOfWeek
> sales_by_day <- retail %>% group_by(dayOfWeek) %>% summarise(TotalSales = sum(TotalSales))
>
> # Plot bar chart
> barplot(sales_by_day$TotalSales, names.arg = c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"),
+         xlab = "Day of Week", ylab = "Total Sales", main = "Total Sales by Day of Week")
>
```

The above code groups the "retail" dataset by the "dayOfWeek" column and calculates the total sales for each day of the week using the summarise() function. The %>% "pipe" operator is used to chain together multiple functions in a single expression, allowing for more concise code.
The resulting dataset, "sales_by_day", has two columns: "dayOfWeek" and "TotalSales", names.arg argument is used to specify the labels for the X-axis (i.e. the days of the week) and barplot() function plots the bar diagram.

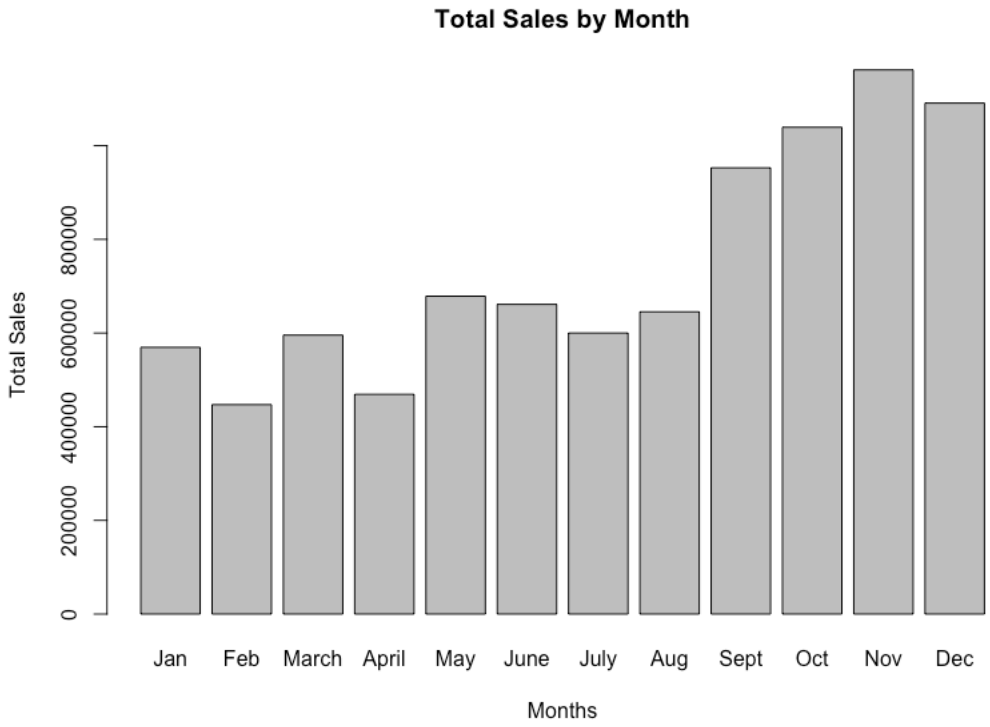


From the above diagram, the total revenue generated on Thursday is more than any other day and least on Sunday.

Revenue By the month of the month Analysis

```
> # Revenue By month of the year Analysis
> sales_by_mnth <- retail %>% group_by(month) %>% summarise(TotalSales = sum(TotalSales))
> # Plot bar chart
> barplot(sales_by_mnth$TotalSales, names.arg = c("Jan", "Feb", "March", "April", "May", "June","July", "Aug", "Sept", "Oct", "Nov", "Dec"),
+         xlab = "Months", ylab = "Total Sales", main = "Total Sales by Month")
>
```

Similarly, as earlier codes, the above codes create a bar plot and compare the months' wise total sales.

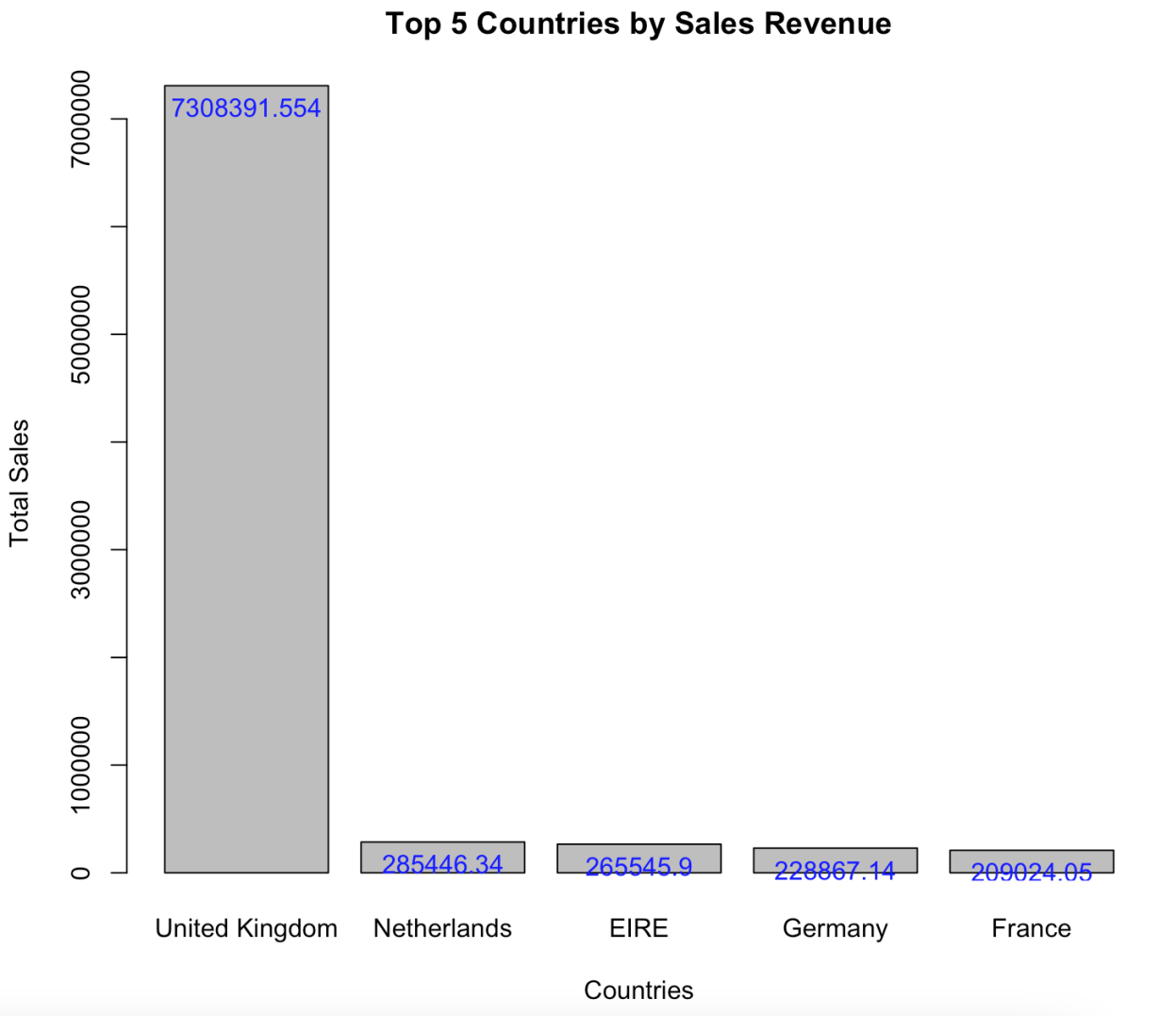


As seen in above diagram, the months September, October, November, and December have more sales.

Revenue country-wise:

```
>
> # Revenue country wise:
> options(scipen = 999) #removes scientific notation
> sales_by_country <- retail %>%
+   group_by(Country) %>%
+   summarise(TotalSales = sum(TotalSales)) %>%
+   arrange(desc(TotalSales)) %>%
+   head(5)
> # Plot bar chart
> bp <- barplot(sales_by_country$TotalSales,
+         names.arg = sales_by_country$Country,
+         xlab = "Countries",
+         ylab = "Total Sales",
+         main = "Top 5 Countries by Sales Revenue")
> # Add labels on top of each bar
> text(x = bp, y = sales_by_country$TotalSales, labels = sales_by_country$TotalSales, pos = 1, col="blue")
>
```

Similarly, as in earlier codes, the total revenue of the top 5 countries is calculated using similar codes. The text() function in R is used to add text labels to a plot, It takes several arguments that specify the position, content, and formatting of the text labels, where labels = sales_by_country$TotalSales: specifies the text labels to be added to the bars, pos = 3: specifies the labels will be positioned above the bars.



From the bar diagram, the united kingdom has the highest total sales, followed by the Netherlands, Ireland, Germany, and France respectively.

4. Model's Baseline : K-Means Clustering:

K-Means clustering is a widely used unsupervised learning algorithm that is used for clustering data points into groups based on their similarities. The algorithm partitions the data into K clusters, where K is a user-specified parameter. The algorithm aims to minimize the sum of squared distances between the data points and the centroid of each cluster. One of the main reasons why K-Means is preferred is its simplicity, speed, and efficiency, especially for large datasets. It is also easy to interpret the results and it can be used for a variety of applications like customer segmentation, image processing, and document clustering.
I used K-means clustering as a data mining algorithm for the project since by using this clustering method, we can group the customers on the basis of different parameters, in this project I have made the clusters of the customers on the basis of their purchasing behavior like recency, frequency and their monetary value. By analyzing these parameters, retailers can make strategies based on the result to enhance their business as well as avoid customer churn by providing discounts to these customers.

The steps I performed for the K-Means clustering are as follows:

Scaling the data

```
> # IV. Model's Baseline
> #K-Means Clustering
> # Scaling the data
> RFM <- data.frame(RFM)
> row.names(RFM) <- RFM$CustomerID
> RFM <- RFM[,-1]
> RFM_scaled <- scale(RFM)
> head(RFM_scaled)
        Frequency    Monetary    Recency
12346 -0.39653199  8.35770470  2.3331552
12347  0.39460347  0.25093734 -0.9058455
12348 -0.26540457 -0.02859271 -0.1660737
12349 -0.08182617 -0.03300799 -0.7358979
12350 -0.32659736 -0.19132522  2.1732046
12352 -0.02937520  0.05025720 -0.5659503
>
```

In the first code, RFM is first converted to a data frame, then, the row names are set as the CustomerID column using row.names() function, which allows for easier referencing of specific customers in the dataset. It assigns the unique CustomerID value to each row as its name, so we can refer to a specific row in the dataset by its CustomerID instead of its row number, then the RFM data frame is then modified to remove the CustomerID column using RFM <- RFM[,-1] as

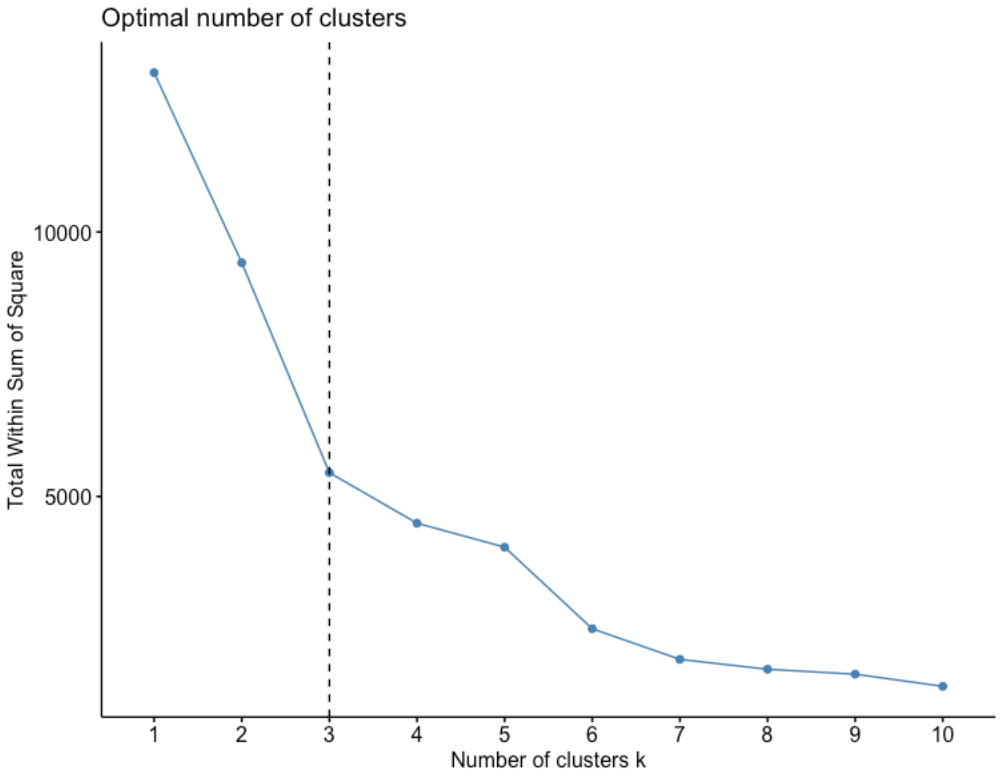
CustomerID is the first column. Next, the RFM data frame is standardized using scale() function to center and scale the data to have zero mean and unit variance. Scaling is a common technique used in data pre-processing to transform the original features into a common scale to make them more comparable. The scale() function in R by default uses the "Z-score" method for standardization, which is also known as "standard score" or "unit score". In this method, the values are scaled to have a mean of 0 and a standard deviation of 1.

Determining Optimal Cluster:
To determine the optimal number of clusters, we can use the **Elbow** and **Silhouette** methods. The Elbow method involves plotting the sum of squared errors (SSE) against the number of clusters and identifying the "elbow" point where further clustering does not significantly improve the SSE.

1. Elbow Method:

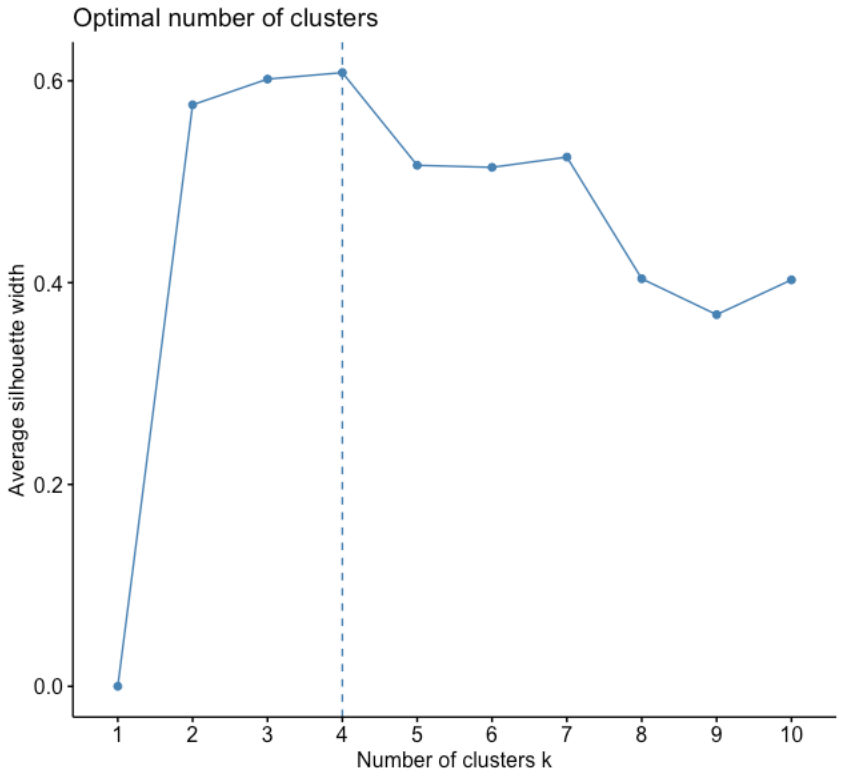
```
> # Determining Optimal Cluster:
> # 1.  Elbow Method:
> set.seed(123)
> fviz_nbclust(RFM_scaled, kmeans, method = "wss") + geom_vline(xintercept = 3, linetype = 2)
>
```

In this code, set.seed(123) sets the random seed to a fixed value (123 in this case) to ensure reproducibility of the analysis and the within-cluster sum of squares (wss) is plotted against different numbers of clusters (k) for k-means clustering using the fviz_nbclust() function from the factoextra package. A vertical line is added at k=3 using the geom_vline() function to help identify the elbow point, which is the optimal number of clusters where the wss starts to level off.



2. Average Silhoute Method:
   The Silhouette method involves calculating the mean silhouette coefficient for each cluster and choosing the number of clusters that maximizes the coefficient.

```
> # 2.  Average Silhoute Method:
> set.seed(123)
> fviz_nbclust(RFM_scaled, kmeans, method = "silhouette")
>
```



The code uses the fviz_nbclust() function from the factoextra package to plot the average silhouette width against different numbers of clusters (k) for k-means clustering, the optimal number of clusters is determined based on the highest average silhouette width i.e., k=4 as seen above diagram.

visualize kmeans clusters using both k=3 and k=4:

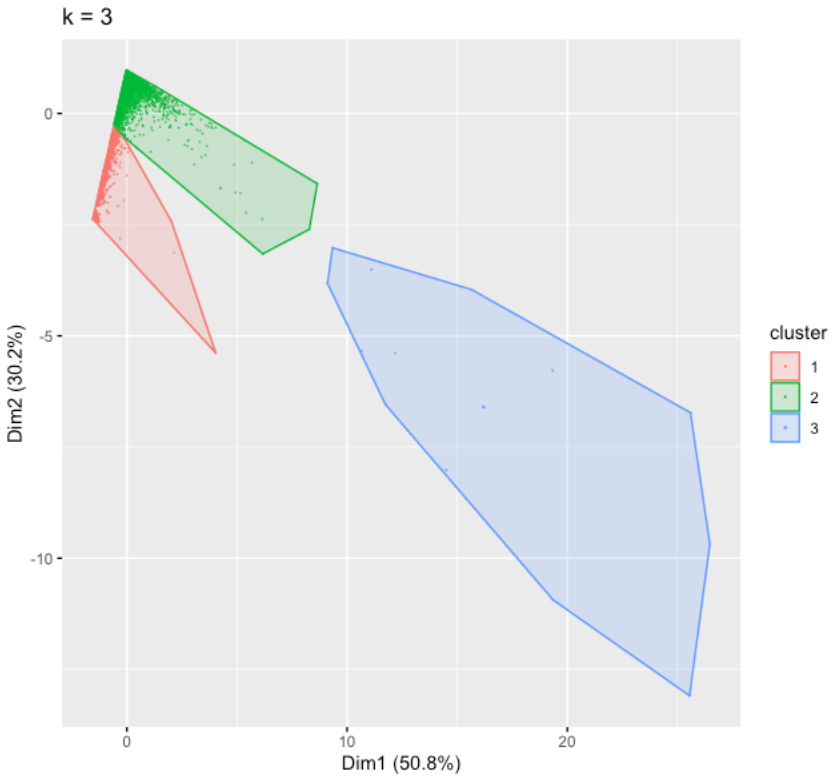
```
> #  visualize kmeans clusters using both k=3 and k=4:
> set.seed(123)
> k3 <- kmeans(RFM_scaled, centers = 3, nstart = 25)
> k4 <- kmeans(RFM_scaled, centers = 4, nstart = 25)
>
```

In the above R code, kmeans() function is used to perform k-means clustering on the standardized data RFM_scaled. The centers parameter specifies the number of clusters to form, which is set to 3 in this case. The nstart parameter specifies the number of times to run the k-means algorithm with different random starting points, which is set to 25 in the first case and assigned to object k3, similarly clustering is done by specifying 4 clusters in the second code and has been assigned to object k4.

K=3:

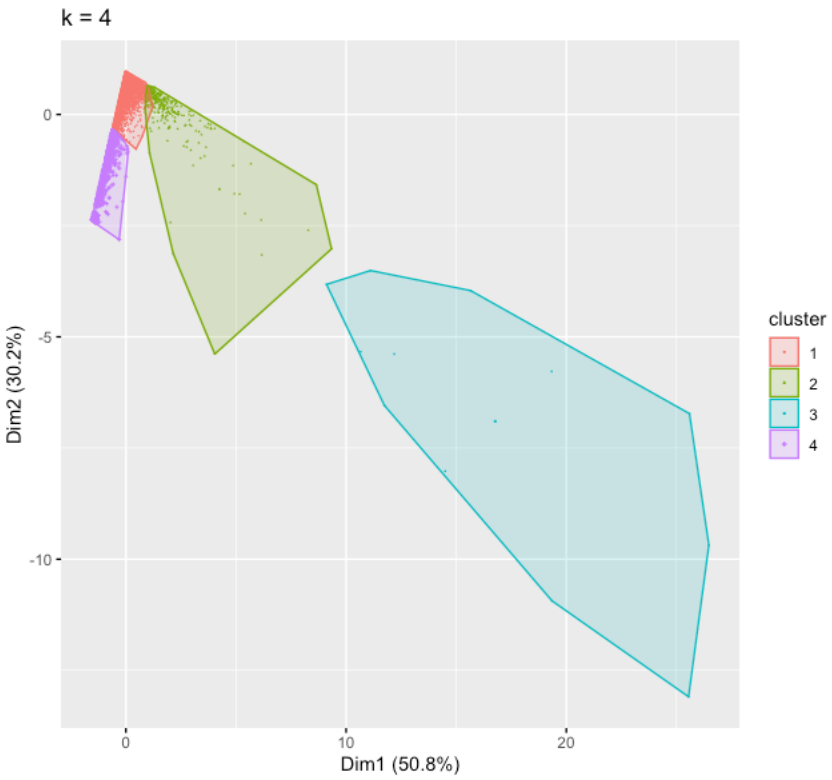
```
> # k=3
> fviz_cluster(k3, geom = "point", data = RFM_scaled, pointsize = 0.2) + ggtitle("k = 3")
>
```

After cluster is formed, visualization is done using the fviz_cluster() function from the factoextra package, which plots the data points colored by their assigned cluster, and includes a title "k=3" using the ggtitle() function. The geom = "point" creates a point plot. The pointsize argument controls the size of the points in the plot.



k = 4:

```
> # k=4
> fviz_cluster(k4, geom = "point", data = RFM_scaled, pointsize = 0.2) + ggtitle("k = 4")
>
```



Similarly, visualization is done for the number of cluster 4.

Based on the results of the clustering algorithms, we compared, we have determined that K=3 is the most suitable and optimal number of clusters. This decision was made because we observed some overlap between clusters when K=4 was used.

summary statistics of each cluster for each of the variables:
Now, after cluster is formed, we will visualize the summary statistics of each cluster for each of the variables in the dataset. After performing clustering, we want to see how each cluster differs from the others in terms of the variables used for clustering.

```
> # summary statistics of each cluster for each of the variables
> res <- cbind(RFM, ClusterId = k3$cluster)
> res <- as.data.frame(res)
> head(res)
      Frequency Monetary Recency ClusterId
12346         1 77183.60     325         1
12347       182  4310.00       1         2
12348        31  1797.24      75         2
12349        73  1757.55      18         2
12350        17   334.40     309         1
12352        85  2506.04      35         2
>
```

The above R codes create a new data frame called res.
The cbind() function is used to combine the RFM dataset with the ClusterId column obtained from performing clustering k3.

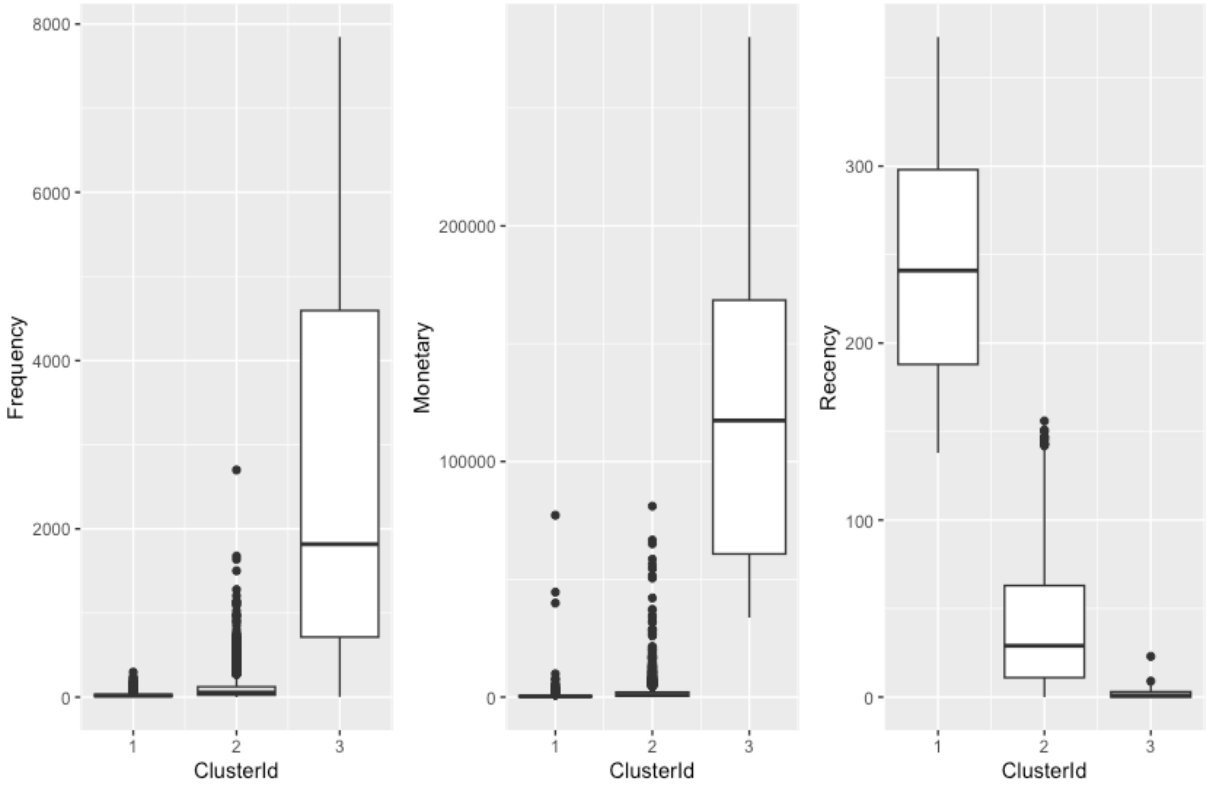
From head() function, we can see that dataframe res has one more column named ClusterId than original dataframe RFM, the column ClusterId gives the information about the cluster group each of the variables in the dataset.

Boxplots are a useful way to visualize the distribution of each variable in each cluster.

```
> library(gridExtra)
> a <- ggplot(res, aes(x = ClusterId, y = Frequency, group = ClusterId )) + geom_boxplot(show.legend = FALSE)
> b <- ggplot(res, aes(x = ClusterId, y = Monetary, group = ClusterId)) + geom_boxplot(show.legend = FALSE)
> c <- ggplot(res, aes(x = ClusterId, y = Recency, group = ClusterId)) + geom_boxplot(show.legend = FALSE)
> grid.arrange(a,b,c, ncol = 3)
>
```

In the above code first we loaded the library gridExtra for multiple grid-based plots.

The code creates a box plot using ggplot2 library for the variable "Frequency" , Monetary and Recency against the "ClusterId" groupings.
aes() function maps the "ClusterId" to the x-axis and "Frequency", Monetary, and Recency to the y-axis, and also sets the grouping.
grid.arrange() helps in arranging multiple plots into a grid in the last code and ncol parameter specifies the number of columns in the grid.



The customers are divided into 3 groups.
From the frequency plot, we can see that cluster 3 people are the ones who spend more.
Also, the last group from the cluster is more frequent customers.
The less the recent value the recent customer purchased product, even here the last group has fewer values.

K-Means clustering is often preferred for several reasons:
For customer segmentation, K-Means clustering is one of the popular clustering algorithms, with its help, we can segment the customers having similar purchasing behavior. By getting this insight into the customers from the dataset, it helps to create target marketing strategies. Moreover, K-Means is chosen for its simplicity as it is easy to implement and interpret using few codes in R, also it is able to handle large datasets making it suitable for large datasets. Besides customer segmentation, K_means clustering algorithms can also be used for image processing, document clustering, etc. Furthermore, it provides accurate results when the data has clear patterns and distinct clusters.

K-means clustering has several advantages which are as follows:

- It is relatively simple to implement.
- It is scalable to large datasets.
- K-means clustering guarantees convergence, which means that the algorithm will eventually reach a stable solution. The algorithm alternates between assigning data points to the closest cluster centroid and updating the centroids until they no longer change.

- K-means can warm-start the positions of centroids, allowing for pre-defined initial values that can help the algorithm converge more quickly and produce more consistent results. These advantages make K-means a reliable and efficient clustering algorithm.
- K-means can easily adapt to new examples and generalize to clusters of different shapes and sizes, including elliptical clusters.

K-means clustering has several disadvantages.

- Firstly, choosing the optimal number of clusters manually can be difficult, but a "Loss vs. Clusters" plot can be used to help find the best value of k.
- Secondly, K-means is dependent on the initial values of the centroids and can have trouble clustering data of varying sizes and densities. To mitigate this, you can run the algorithm several times with different initial values or use advanced versions of K-means.
- Additionally, K-means can have trouble clustering outliers and can scale poorly with an increasing number of dimensions. To address these challenges, outlier removal, dimensionality reduction, and alternative clustering algorithms can be used.

5. Conclusion:

In this project, we have used RFM analysis (Recency, Frequency, Monetary) for segmenting the customer in the online retail dataset based on their purchase behavior. The dataset contains approximately 541,909 transactions made between 01/12/2010 and 09/12/2011. The features included in the dataset were InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country. For the data mining algorithm, we have used K-Means clustering, which further refines customer segmentation and identify distinct groups of customers with similar purchase behavior and preferences. By identifying these clusters, the company can target their marketing efforts more effectively and tailor their strategies to meet the specific needs and preferences of each customer segment. Overall, this analysis provides valuable insights into customer behavior and can help the company to improve customer retention and increase revenue." So, k-means clustering can be used to segment customers into groups based on their purchasing behavior and other attributes. One of the main benefits of clustering is that instead of examining the whole client base as a whole, it is fruitful to segment them into clusters and engage them with relevant deals and avoid the churn rate. Also, companies (retailers) can identify their loyal customers and provide discounts for them, and engage with them for their services so that overall satisfaction will increase. also, they can identify less potential customers which will help businesses tailor their marketing strategies and improve customer engagement.

The main struggles and difficulties faced while using R-studio to build a K-Means clustering model include dealing with large datasets that require extensive processing power and memory. As my dataset is large with 541,909 observations, sometimes I had to restart R-studio again due to processing issues. Another challenge I faced using R-studio is to find the appropriate R codes for data visualization, there are lots of articles on the internet for Python as compared to R. For segmenting the customer, deciding which features will be appropriate was quite challenging, and finding the appropriate R code to do that. Also, interpreting the results of the K-Means clustering model can be a challenge, as the results may not always be intuitive or easy to explain. It is important to carefully analyze the clustering results and identify any patterns or insights that can be gained from the data.

There are several softwares that can work properly with K-Means Clustering Model.

A) Python: Python is one of the most widely used programming languages for data mining and has several libraries that support K-means clusterings, such as Scikit-learn and PyClustering. It is also easy to use and has a large community that provides support and resources.

B) MATLAB : MATLAB is another software f which is proficient in supporting various data mining algorithms, including K-means clustering. It offers an intuitive user interface and provides a diverse range of data analysis and visualization tools and is easy to use.

C) SAS: SAS is a software that is commonly used for statistical analysis and business intelligence. It offers a broad range of tools that are useful for cluster analysis, including K-means clustering. It is well-regarded for its strong, precise, and secure features.

References:

https://rpubs.com/sheng0628/273249

https://www.kaggle.com/code/sanjeet41/online-retail-data-analysis-in-r

https://rstudio-pubs-static.s3.amazonaws.com/430563_d38c12b53d724fa6852949b1f3e4ffbf.html

https://rstudio-pubs-static.s3.amazonaws.com/958718_a4fdb15aba0a4ce8bb7ac153d26e052b.html

https://rpubs.com/Dikshita24/Clustering

https://medium.com/web-mining-is688-spring-2021/exploring-customers-segmentation-with-rfm-analysis-and-k-means-clustering-118f9ffcd9f0