HarvardX:PH125.9x Data Science - Capstone: Customer Segmentation Project

Richard Jonyo

06 March 2022

Contents

1. Executive Summary

The goal of this project is to identify segments of customers based on common characteristics or patterns. Segmentation of customer can take many forms, based on demographic, geographic, interest, behavior or a combination of these characteristics. Through analyzing customer purchases and product sales history, we can group customers into groups that behave similarly, and use the insights to drive decision making especially targeted marketing strategies.

For this project we shall employ K-mean Clustering which is the essential algorithm for clustering. We shall have to format the data in a way the algorithm can process, and we shall let it determine the customer segments.

This project is part of the HarvardX:PH125.9x Data Science: Capstone course and we use the E-Commerce Dataset from Kaggel.com repository. The dataset lists purchases made by over 4,000 customers over a period of one year (from 2010/12/01 to 2011/12/09).

Exploratory analysis was conducted on the data using R and R Studio, a language and a software environment for statistical computing. R Markdown, a simple formatting syntax for authoring HTML, PDF, and MS Word documents and a component of R Studio was used to compile the report.

2. Methods and Exploratory Analysis

This section helps us to understand the structure of the E-Commerce Dataset for us to gain insights that will aid in a better analysis for customer segmentation. It explains the process and techniques used, including data cleaning, exploration, visualization, insights gained, and the segmentation approach used.

2.1 Required Libraries

The project utilized and loaded several CRAN libraries to assist with the analysis. The libraries were automatically downloaded and installed during code execution. These included: ggplot2, dplyr, tidyr, DataExplorer, lubridate, heatmaply, dlookr, highcharter, purrr, factoextra, and scales libraries.

```
# Note: this process could take a couple of minutes
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-
project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-
project.org")
if(!require(tidyr)) install.packages("tidyr", repos = "http://cran.us.r-
project.org")if(!require(lubridate)) install.packages("lubridate", repos =
"http://cran.us.r-project.org")
if(!require(dataexplorer)) install.packages("dataexplorer", repos =
"http://cran.us.r-project.org")
if(!require(heatmaply)) install.packages("heatmaply", repos =
"http://cran.us.r-project.org")
```

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

library(ggplot2)
library(dplyr)
library(tidyr)
library(DataExplorer)
library(lubridate)
library(heatmaply)
library(dlookr)
library(highcharter)
library(purrr)
library(factoextra)
library(scales)
```

2.2 The E-Commerce Dataset

The E-Commerce Dataset contains 541,909 transactions by 4,373 unique customers. Each customer is represented by a CustomerID and customers can have multiple transactions. The dataset is loaded locally as a CSV file.

```r
# the E-Commerce Dataset:
# https://www.kaggle.com/fabiendaniel/customer-segmentation/data

#dataset downloaded and loaded locally
customer_data <- read.csv("dataset/data.csv")

head(customer_data)#Preview customer_data

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

The following is a brief description of the data.

- **InvoiceNo:** A 6-digit number uniquely assigned to each transaction. The letter C indicates a cancellation.
- **StockCode:** A 5-digit number that is uniquely assigned to each product.
- **Description:** Product name.
- **Quantity:** The quantities of each product per transaction.
- **InvoiceDate:** Invoice date and time. Includes the day and time when a transaction was generated.
- **UnitPrice:** Product price per unit.
- **CustomerID:** A 5-digit number uniquely assigned to each customer.
- **Country:** The name of the customer's country.

The dataset has 541,909 observations and 8 columns.

```
dim(customer_data) #541,909 observations and 8 columns

## [1] 541909      8
```

We see three data types: character, integer, and number.

```
str(customer_data) #view datatypes

## 'data.frame':    541909 obs. of  8 variables:
##  $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr  "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL
LANTERN" "CREAM CUPID HEARTS COAT HANGER" "KNITTED UNION FLAG HOT WATER
BOTTLE" ...
##  $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: chr  "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 dataset has 8 columns namely: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country.

```
names(customer_data)#view column names

## [1] "InvoiceNo"   "StockCode"   "Description" "Quantity"    "InvoiceDate"
## [6] "UnitPrice"   "CustomerID"  "Country"
```

The dataset has 4,373 unique customers.

```
length(unique(customer_data$CustomerID))#4,373 unique customers

## [1] 4373
```

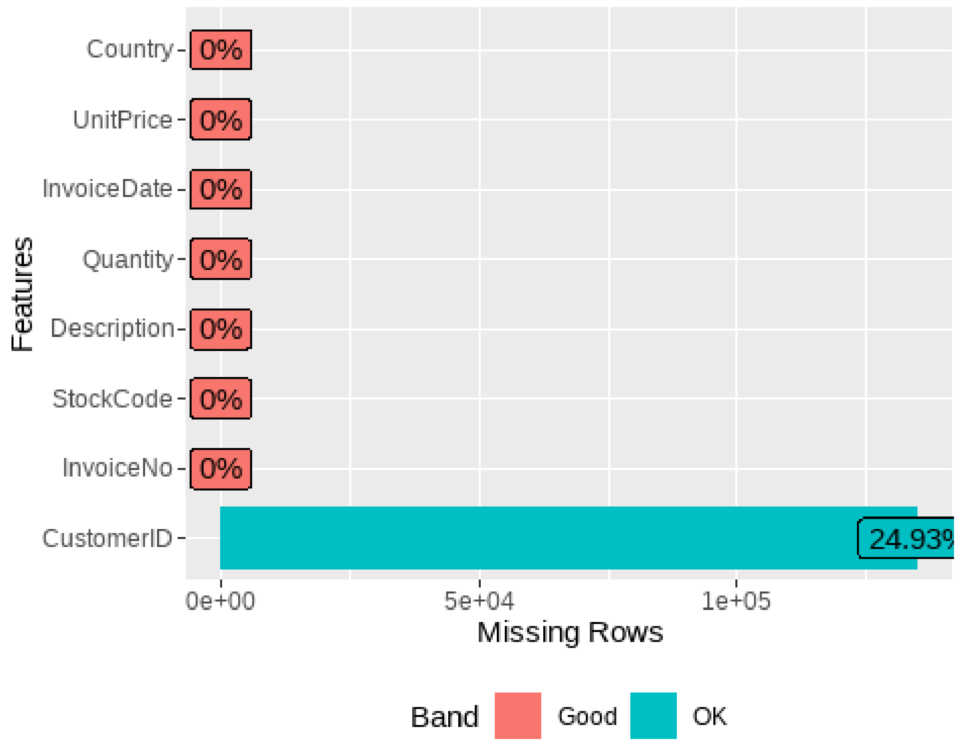
2.3 Missing values

There are some missing values in the dataset on the CustomerID column.

```
anyNA(customer_data) #check missing values - results to TRUE

## [1] TRUE
```

We plot missing values and discover that customerID has many missing values. Over 25% of the entries are not assigned to a particular customer.

```
#We plot missing values
plot_missing(customer_data)
```



The Quantity and UnitPrice seem to be the most important in the dataset. We notice that both variables have some negative values. We check their summaries as below:

```
summary(customer_data$Quantity)#Quantity summary

##       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -80995.00      1.00      3.00      9.55     10.00  80995.00

summary(customer_data$UnitPrice)#UnitPrice summary

##       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -11062.06      1.25      2.08      4.61      4.13  38970.00
```

We notice that we have negative quantities within the dataset which reflect cancelled orders. These are indicated with the C letter in front of the Invoice Number.

```
#Checking Quantities with negative values
quantityCheck <- customer_data %>%
  filter(Quantity < 0) %>%
  arrange(Quantity)
head(quantityCheck, 5)

##   InvoiceNo StockCode                         Description Quantity
## 1   C581484     23843         PAPER CRAFT , LITTLE BIRDIE   -80995
## 2   C541433     23166      MEDIUM CERAMIC TOP STORAGE JAR   -74215
## 3    556690     23005        printing smudges/thrown away    -9600
## 4    556691     23005        printing smudges/thrown away    -9600
## 5   C536757     84347 ROTATING SILVER ANGELS T-LIGHT HLDR    -9360
##         InvoiceDate UnitPrice CustomerID        Country
## 1  12/9/2011 9:27      2.08      16446 United Kingdom
## 2 1/18/2011 10:17      1.04      12346 United Kingdom
## 3 6/14/2011 10:37      0.00         NA United Kingdom
## 4 6/14/2011 10:37      0.00         NA United Kingdom
## 5 12/2/2010 14:23      0.03      15838 United Kingdom
```

2.4 Negative values

We replace all negative quantities with NA so that they don't not adversely affect our results. In addition, we delete all transactions that do not have a customerID and we remain with 397,884 observations.

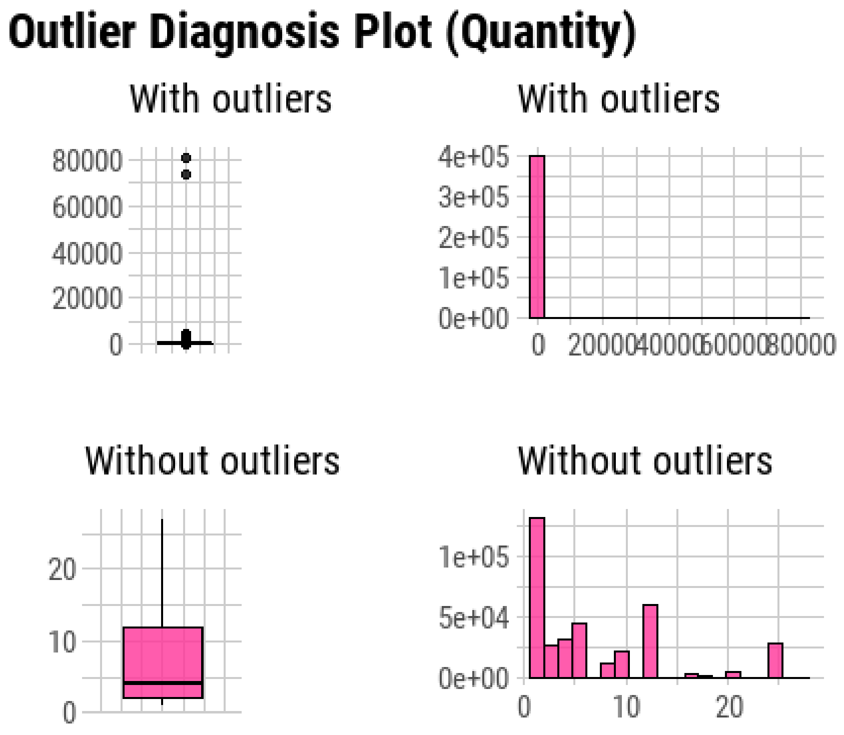
```
#Replace all negative Quantity and Price with NA
customer_data <- customer_data %>%
  mutate(Quantity = replace(Quantity, Quantity<=0, NA),
         UnitPrice = replace(UnitPrice, UnitPrice<=0, NA))
customer_data <- customer_data %>%
  drop_na() #Delete any customerID with NA
dim(customer_data) #we remain with 397,884 observations

## [1] 397884      8
```

2.5 Outliers

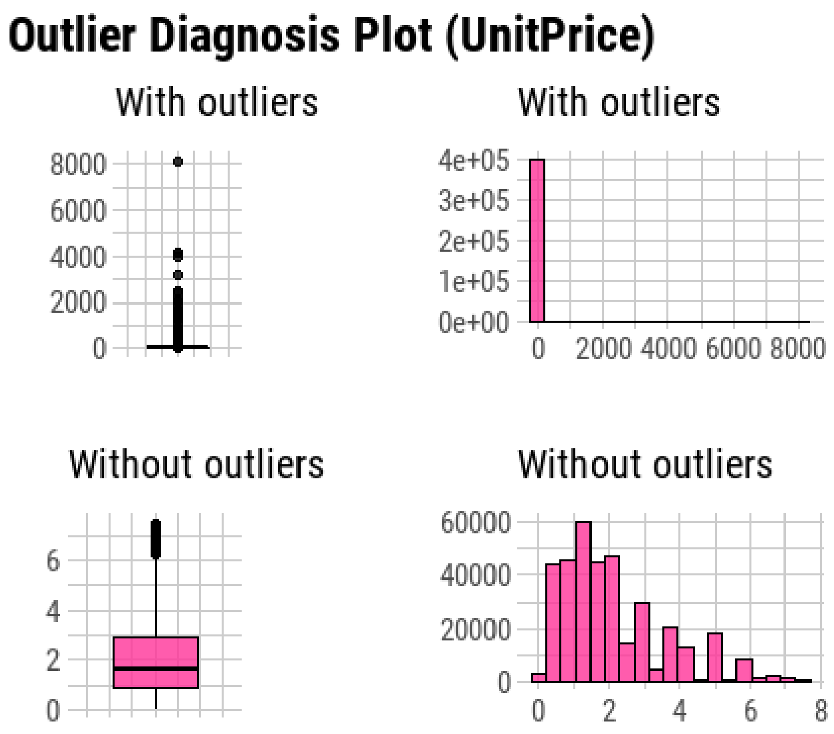
We check for the presence of outliers. We notice that most of the products that are being sold are mostly a low priced. Some of these outliers could be cancelled or wrong orders that got returned and thus were assigned with a negative value. We leave the outliers as removing them will undermined the analysis.

A charts for the outliers *(using the dlookr package)* confirms that there are some negative values in our data for Quantity, as well as some zero value inputs.

```
#We check again the presence of outliers
plot_outlier(customer_data, Quantity, col = "#FF3399")
```



```
plot_outlier(customer_data, UnitPrice, col = "#FF3399")
```



2.6 Creating date variables

We separate date and time components from the invoice date.

```
#We separate date and time components of invoice date
customer_data$date <- sapply(customer_data$InvoiceDate, FUN = function(x)
{strsplit(x, split = '[ ]')[[1]][1]})
customer_data$time <- sapply(customer_data$InvoiceDate, FUN = function(x)
{strsplit(x, split = '[ ]')[[1]][2]})
```

We create time, month, year and hour of day variables.

```
#we create month, year and hour of day columns
customer_data$month <- sapply(customer_data$date, FUN = function(x)
{strsplit(x, split = '[/]')[[1]][1]})
customer_data$year <- sapply(customer_data$date, FUN = function(x)
{strsplit(x, split = '[/]')[[1]][3]})
customer_data$hourOfDay <- sapply(customer_data$time, FUN = function(x)
{strsplit(x, split = '[:]')[[1]][1]})
tmp <- customer_data %>% select(CustomerID, Country, date, time, month, year,
hourOfDay)
head(tmp)
```

```
##   CustomerID        Country      date time month year hourOfDay
## 1      17850 United Kingdom 12/1/2010 8:26    12 2010         8
## 2      17850 United Kingdom 12/1/2010 8:26    12 2010         8
## 3      17850 United Kingdom 12/1/2010 8:26    12 2010         8
## 4      17850 United Kingdom 12/1/2010 8:26    12 2010         8
## 5      17850 United Kingdom 12/1/2010 8:26    12 2010         8
## 6      17850 United Kingdom 12/1/2010 8:26    12 2010         8
```

We convert date column to date format and create a new column for day of the week, using the *wday function* from the lubridate package.

```
#We convert date column to date format
#We can create day of the week column
customer_data$date <- as.Date(customer_data$date, "%m/%d/%Y")
str(customer_data)

## 'data.frame':    397884 obs. of  13 variables:
##  $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr  "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL
LANTERN" "CREAM CUPID HEARTS COAT HANGER" "KNITTED UNION FLAG HOT WATER
BOTTLE" ...
##  $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: chr  "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" ...
##  $ date       : Date, format: "2010-12-01" "2010-12-01" ...
##  $ time       : chr  "8:26" "8:26" "8:26" "8:26" ...
##  $ month      : chr  "12" "12" "12" "12" ...
##  $ year       : chr  "2010" "2010" "2010" "2010" ...
##  $ hourOfDay  : chr  "8" "8" "8" "8" ...

customer_data$dayOfWeek <- wday(customer_data$date, label=TRUE)
tmp <- customer_data %>% select(CustomerID, Country, date, time, month, year,
hourOfDay, dayOfWeek)
head(tmp)

##   CustomerID        Country       date time month year hourOfDay dayOfWeek
## 1      17850 United Kingdom 2010-12-01 8:26    12 2010         8       Wed
## 2      17850 United Kingdom 2010-12-01 8:26    12 2010         8       Wed
## 3      17850 United Kingdom 2010-12-01 8:26    12 2010         8       Wed
## 4      17850 United Kingdom 2010-12-01 8:26    12 2010         8       Wed
## 5      17850 United Kingdom 2010-12-01 8:26    12 2010         8       Wed
## 6      17850 United Kingdom 2010-12-01 8:26    12 2010         8       Wed
```

2.7 Creating a TotalCost Column

We add TotalCost column by multiplying Quantity and UnitPrice.

```r
#add TotalCost column
customer_data <- customer_data %>% mutate(TotalCost = Quantity * UnitPrice)
```

We convert appropriate columns (month, year, hourOfDay, dayOfWeek, and Country) into factors.

```r
#we turn the appropriate variables into factors
customer_data$month <- as.factor(customer_data$month)
customer_data$year <- as.factor(customer_data$year)
levels(customer_data$year) <- c(2010,2011)
customer_data$hourOfDay <- as.factor(customer_data$hourOfDay)
customer_data$dayOfWeek <- as.factor(customer_data$dayOfWeek)
customer_data$Country <- as.factor(customer_data$Country)
str(customer_data)
```

```
## 'data.frame':    397884 obs. of  15 variables:
##  $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr  "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL
LANTERN" "CREAM CUPID HEARTS COAT HANGER" "KNITTED UNION FLAG HOT WATER
BOTTLE" ...
##  $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: chr  "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    : Factor w/ 37 levels "Australia","Austria",..: 35 35 35 35
35 35 35 35 35 35 ...
##  $ date       : Date, format: "2010-12-01" "2010-12-01" ...
##  $ time       : chr  "8:26" "8:26" "8:26" "8:26" ...
##  $ month      : Factor w/ 12 levels "1","10","11",..: 4 4 4 4 4 4 4 4 4 4
...
##  $ year       : Factor w/ 2 levels "2010","2011": 1 1 1 1 1 1 1 1 1 1 ...
##  $ hourOfDay  : Factor w/ 15 levels "10","11","12",..: 14 14 14 14 14 14
14 14 14 14 ...
##  $ dayOfWeek  : Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 4 4 4 4 4 4 4
4 4 4 ...
##  $ TotalCost  : num  15.3 20.3 22 20.3 20.3 ...
```

2.8 Exploratory Analysis

We have a better dataset to start performing analyses. We employ Exploratory Data Analysis (EDA) to conduct an initial investigation inside the dataset and observe common patterns, spot anomalies and retrieve useful information about the data in a graphical way. We need to understand the dataset before starting to develop the models hence an exploratory analysis is significant.

Below is a quick summary of the dataset:

```
summary(customer_data)

##   InvoiceNo          StockCode          Description          Quantity     
##  Length:397884      Length:397884      Length:397884      Min.   :    1.00
##  Class :character   Class :character   Class :character   1st Qu.:    2.00
##  Mode  :character   Mode  :character   Mode  :character   Median :    6.00
##                                                           Mean   :   12.99
##                                                           3rd Qu.:   12.00
##                                                           Max.   :80995.00
##                                                                           
##  InvoiceDate          UnitPrice          CustomerID                Country      
##  Length:397884      Min.   :   0.001   Min.   :12346   United 
Kingdom:354321  
##  Class :character   1st Qu.:   1.250   1st Qu.:13969   Germany       :  
9040  
##  Mode  :character   Median :   1.950   Median :15159   France        :  
8341  
##                     Mean   :   3.116   Mean   :15294   EIRE          :  
7236  
##                     3rd Qu.:   3.750   3rd Qu.:16795   Spain         :  
2484  
##                     Max.   :8142.750   Max.   :18287   Netherlands   :  
2359  
##                                                        (Other)       : 
14103  
##       date                time               month           year       
##  Min.   :2010-12-01   Length:397884      11     : 64531   2010: 26157  
##  1st Qu.:2011-04-07   Class :character   10     : 49554   2011:371727  
##  Median :2011-07-31   Mode  :character   12     : 43461                
##  Mean   :2011-07-10                      9      : 40028                
##  3rd Qu.:2011-10-20                      5      : 28320                
##  Max.   :2011-12-09                      6      : 27185                
##                                          (Other):144805                
##    hourOfDay     dayOfWeek    TotalCost       
##  12     :72065   Sun:62773   Min.   :     0.00  
##  13     :64026   Mon:64893   1st Qu.:     4.68  
##  14     :54118   Tue:66473   Median :    11.80  
##  11     :49084   Wed:68885   Mean   :    22.40  
##  15     :45369   Thu:80035   3rd Qu.:    19.80  
##  10     :37997   Fri:54825   Max.   :168469.60  
##  (Other):75225   Sat:    0
```

The dataset has 4338 unique customerIDs and 18532 unique invoice numbers.

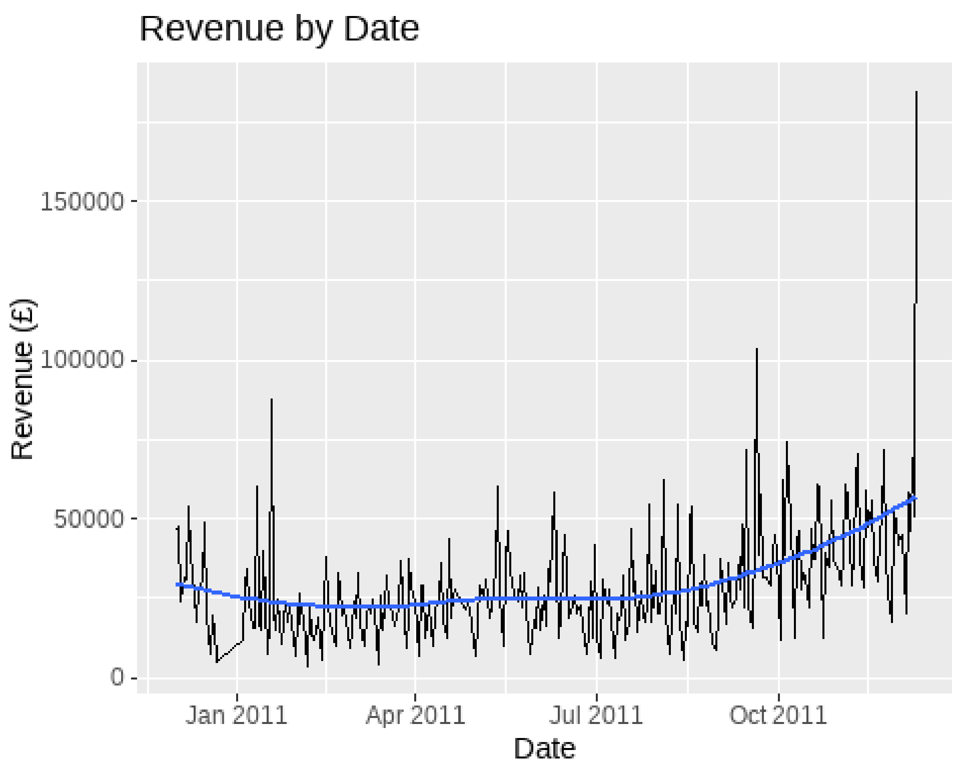
```
length(unique(customer_data$CustomerID)) #4,373 unique customerIDs

## [1] 4338

length(unique(customer_data$InvoiceNo)) #18,532 unique invoice no.s
```

```
## [1] 18532
```

2.8.1 Revenue Summaries

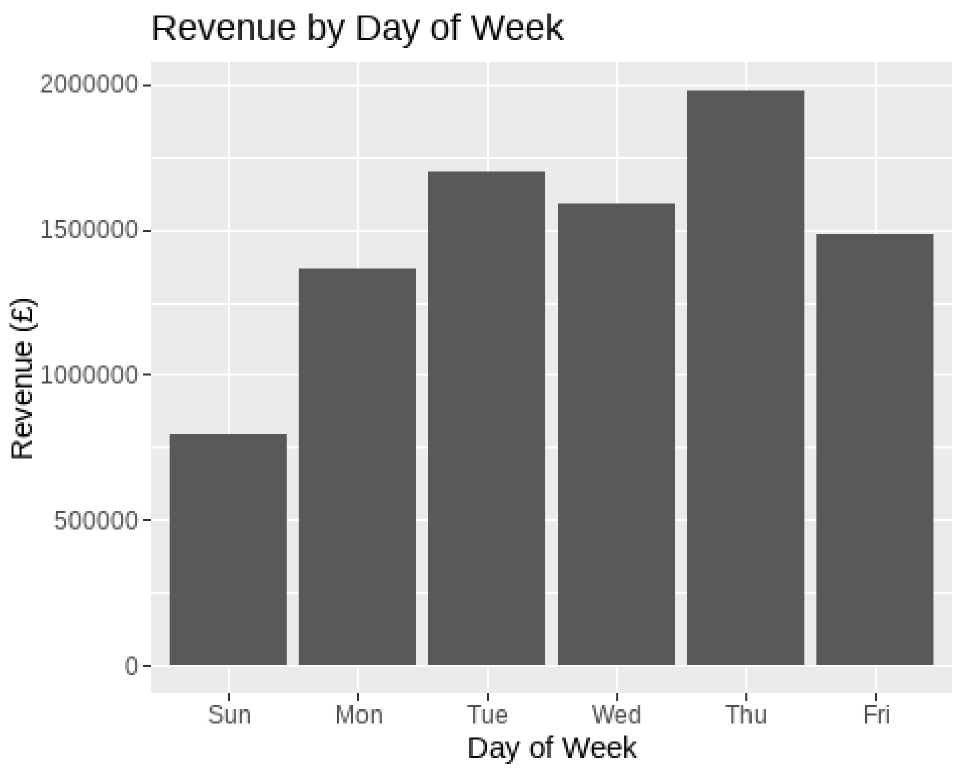
From the chart below there seem to be a steady positive increase in revenue over time with the highest peak in September 2011.

```
#Plot revenues over time
options(repr.plot.width=8, repr.plot.height=3)
customer_data %>%
  group_by(date) %>%
  summarise(revenue = sum(TotalCost)) %>%
  ggplot(aes(x = date, y = revenue)) +
  geom_line() +
  geom_smooth(method = 'auto', se = FALSE) +
  labs(x = 'Date', y = 'Revenue (£)', title = 'Revenue by Date')
```



The chart below shows the Revenue by Day of Week. Most revenue is generated o Thursdays and Tuesdays and the least revenue is generated on Sundays.

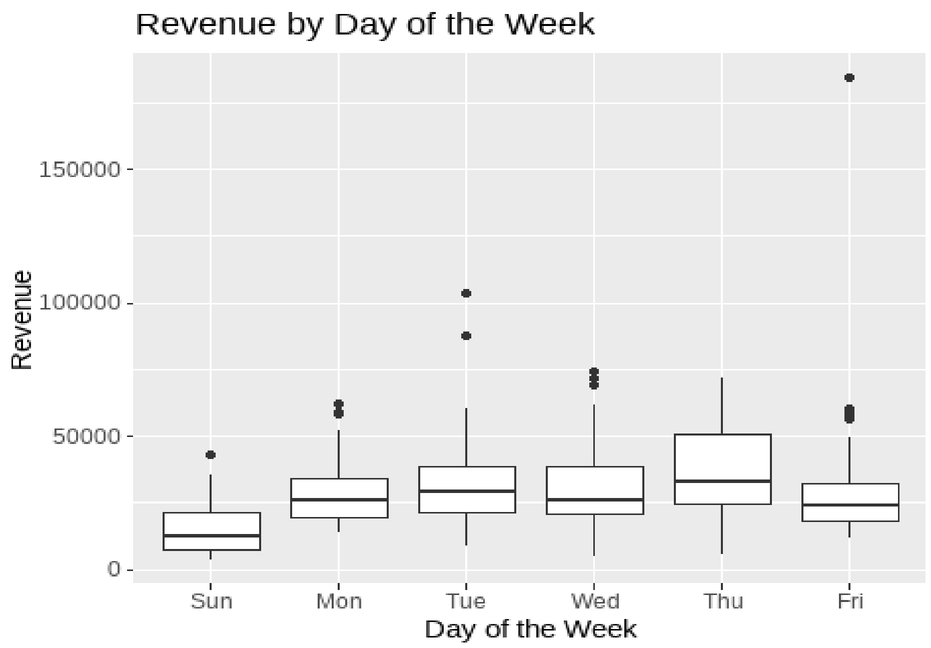
```
# Plot Revenue by Day of Week
customer_data %>%
  group_by(dayOfWeek) %>%
  summarise(revenue = sum(TotalCost)) %>%
  ggplot(aes(x = dayOfWeek, y = revenue)) +
  geom_col() +
  labs(x = 'Day of Week', y = 'Revenue (£)', title = 'Revenue by Day of
Week')
```



```
#Summary of revenue generated on each weekday
weekdaySummary <- customer_data %>%
  group_by(date, dayOfWeek) %>%
  summarise(revenue = sum(TotalCost), transactions = n_distinct(InvoiceNo))
%>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup()
```

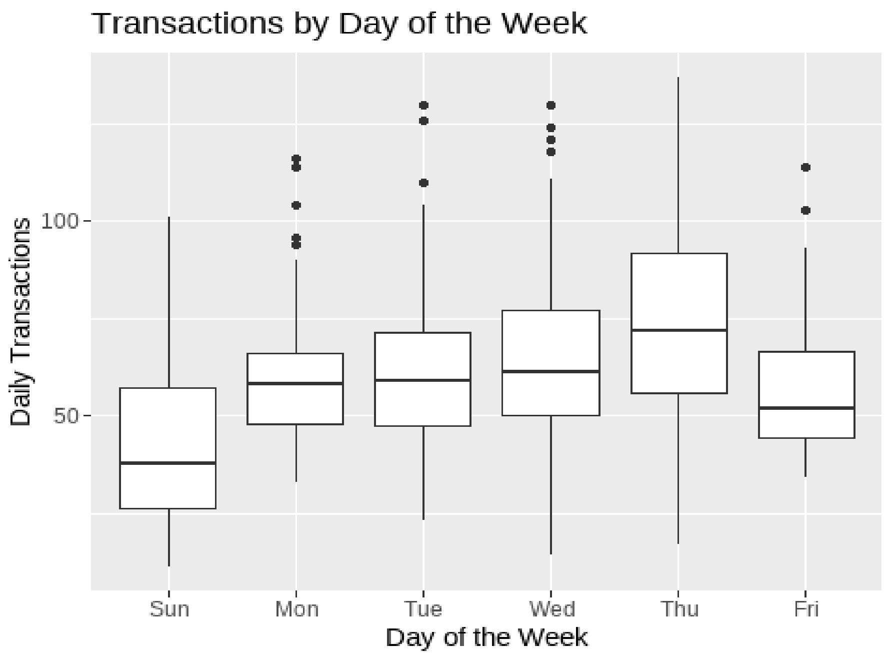
The plot below shows the Revenue by Day of the Week.

```
#Plot of Revenue by Day of the Week
ggplot(weekdaySummary, aes(x = dayOfWeek, y = revenue)) +
  geom_boxplot() +
  labs(x = 'Day of the Week', y = 'Revenue', title = 'Revenue by Day of the
Week')
```



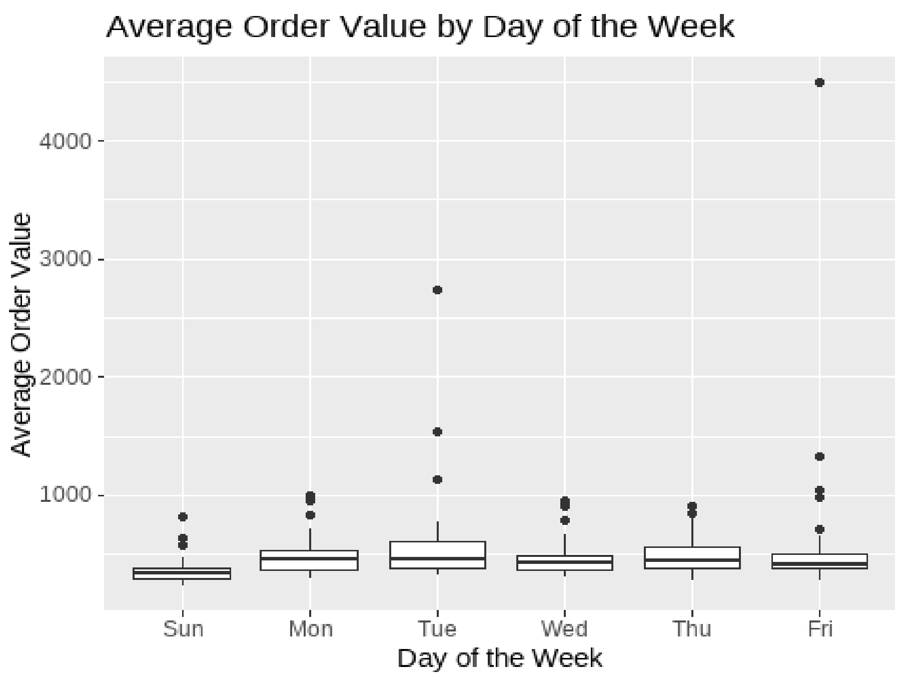
The chart below shows the transactions by day of the Week. It is evident that there are more transactions on Thursdays which is also when we have the most revenue generated. the same trend applies to Sunday which has the least transactions and the least revenue generated.

```
#Plot of Transactions by Day of the Week
ggplot(weekdaySummary, aes(x = dayOfWeek, y = transactions)) +
  geom_boxplot() + labs(x = 'Day of the Week', y = 'Daily Transactions',
title = 'Transactions by Day of the Week')
```



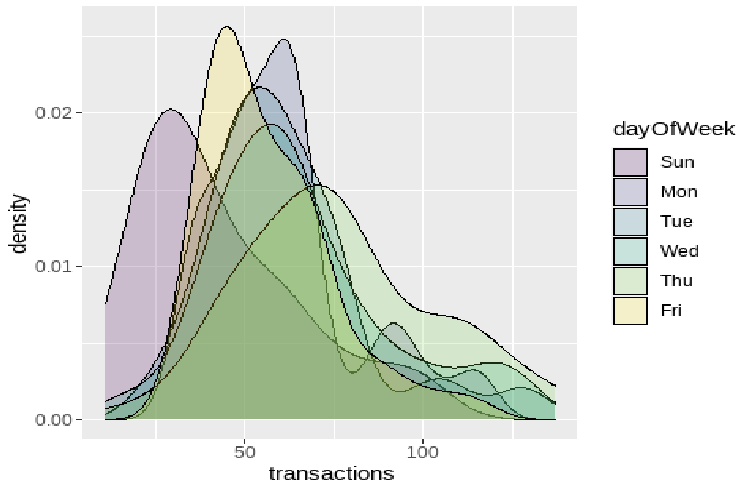
The differences in the amount of revenue on each day of the week is driven by a difference in the no. of transactions, rather than the average order value as is evident in the chart below.

```
#Plot of Average Order Value by Day of the Week
ggplot(weekdaySummary, aes(x = dayOfWeek, y = aveOrdVal)) +
  geom_boxplot() + labs(x = 'Day of the Week', y = 'Average Order Value',
title = 'Average Order Value by Day of the Week')
```



The chart below shows that there is skewness in our distributions with the least number of transactions leaning towards Sunday and Friday.

```
ggplot(weekdaySummary, aes(transactions, fill = dayOfWeek)) +
  geom_density(alpha = 0.2)
```



2.8.2 Country Summaries

We now examine the data summaries of the countries. United Kingdom has the most revenue and the most transactions while United Arab Emirates has the least transactions and revenue.

```
countrySummary <- customer_data %>%
  group_by(Country) %>%
  summarise(revenue = sum(TotalCost), transactions = n_distinct(InvoiceNo))
%>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup() %>%
  arrange(desc(revenue))
head(countrySummary, n = 10)
```

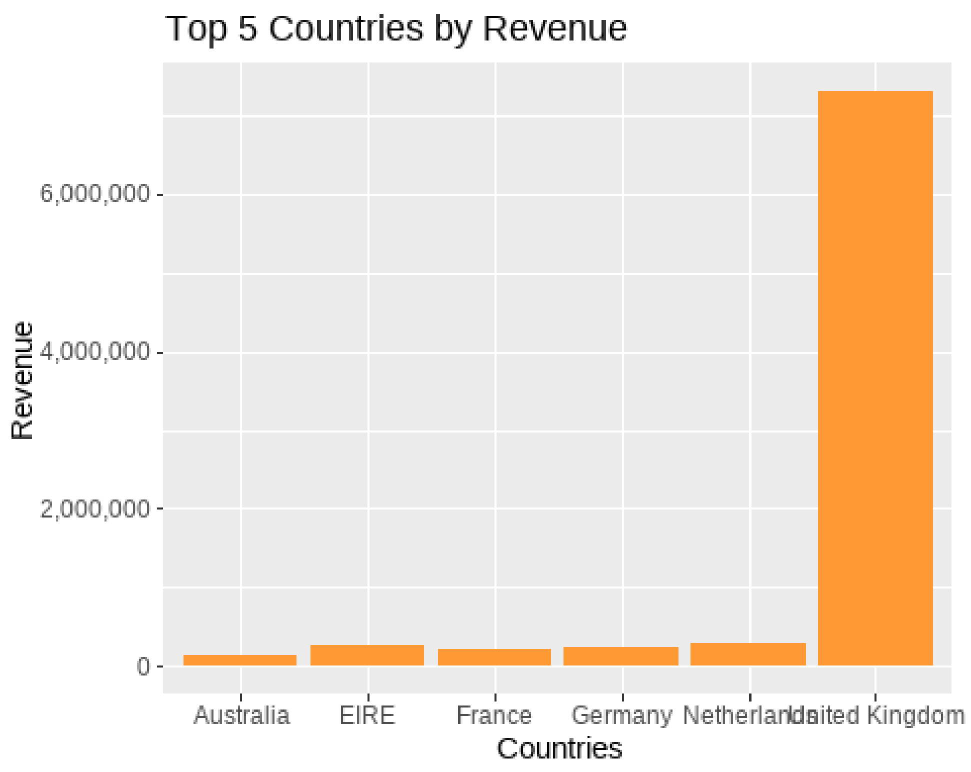
```
## # A tibble: 10 x 4
##    Country          revenue transactions aveOrdVal
##    <fct>              <dbl>        <int>     <dbl>
##  1 United Kingdom 7308392.        16646      439.
##  2 Netherlands     285446.           94     3037.
##  3 EIRE            265546.          260     1021.
##  4 Germany         228867.          457      501.
##  5 France          209024.          389      537.
##  6 Australia       138521.           57     2430.
##  7 Spain            61577.           90      684.
##  8 Switzerland      56444.           51     1107.
##  9 Belgium          41196.           98      420.
## 10 Sweden           38378.           36     1066.
```

The chart below shows the top five countries in terms of revenue contribution.

```
#Top five countries in revenue contribution
top5Countries <- customer_data %>%
  filter(Country == 'United Kingdom' | Country == 'Netherlands' | Country ==
'EIRE' | Country == 'Germany' | Country == 'France' | Country == 'Australia')
```

A table of top five countries in terms of revenue generation

```
#dataframe of top 5 coutries
top_5 <- top5Countries %>%
  group_by(Country) %>%
  dplyr::summarise(revenue = sum(TotalCost), transactions =
n_distinct(InvoiceNo),
                   customers = n_distinct(CustomerID)) %>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  arrange(desc(revenue))

#Plot countries vs. revenue
top_5 %>%
  ggplot(aes(x=Country, y=revenue))+
  geom_bar(stat = 'identity', fill = '#FF9933') +
  ggtitle('Top 5 Countries by Revenue') +
  xlab('Countries') +
  ylab('Revenue')+
  scale_y_continuous(labels = comma)
```



We repeat the above step without United Kingdom to remove bias and to see the other countries revenue clearly. According to the chart below, Netherlands and EIRE also have significant revenue generation. Germany and France represent significant opportunities but at a lower level.

```
#Plot top 5 country revenue summary
#
top5Countries <- customer_data %>%
  filter(Country == 'Netherlands' | Country == 'EIRE' | Country == 'Germany'
| Country == 'France' | Country == 'Australia')
top_5 <- top5Countries %>%
  group_by(Country) %>%
  dplyr::summarise(revenue = sum(TotalCost), transactions =
n_distinct(InvoiceNo),
                   customers = n_distinct(CustomerID)) %>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  arrange(desc(revenue))
top_5 %>%
  group_by(Country) %>%
  dplyr::summarise(revenue = sum(revenue)) %>%
  hchart('treemap', hcaes(x = 'Country', value = 'revenue', color =
'revenue')) %>%
  hc_title(text=" Top 5 Countries by Revenue (excluding United Kingdom)")

top_5

## # A tibble: 5 x 5
##   Country     revenue transactions customers aveOrdVal
##   <fct>         <dbl>        <int>     <int>     <dbl>
## 1 Netherlands 285446.           94         9     3037.
## 2 EIRE        265546.          260         3     1021.
## 3 Germany     228867.          457        94      501.
## 4 France      209024.          389        87      537.
## 5 Australia   138521.           57         9     2430.
```

2.9 Customer segmentation

If we can identify a group of customers who make purchases regularly, we can target this group with dedicated marketing campaigns which may reinforce their loyalty. Here we use the CustomerID to look for differences between customers.

We summarize the top customers by revenue.

```
custSummary_1 <- customer_data %>%
  group_by(CustomerID) %>%
  summarise(revenue = sum(TotalCost), transactions = n_distinct(InvoiceNo))
%>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup() %>%
  arrange(desc(revenue))

head(custSummary_1)

## # A tibble: 6 x 4
##   CustomerID revenue transactions aveOrdVal
##        <int>   <dbl>        <int>     <dbl>
## 1      14646 280206.           73     3838.
## 2      18102 259657.           60     4328.
## 3      17450 194551.           46     4229.
## 4      16446 168472.            2    84236.
## 5      14911 143825.          201      716.
## 6      12415 124915.           21     5948.
```

The summary below shows that there seems to be quite a lot of high-quantity sales and refunds as well.

```
#summarize customers with high revenues/sales
custSummary_2 <- customer_data %>%
  group_by(CustomerID, InvoiceNo) %>%
  summarise(revenue = sum(TotalCost), transactions = n_distinct(InvoiceNo))
```

```
%>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup() %>%
  arrange(revenue) %>%
  mutate(cumsum=cumsum(revenue))

head(custSummary_2)

## # A tibble: 6 x 6
##   CustomerID InvoiceNo revenue transactions aveOrdVal cumsum
##        <int> <chr>       <dbl>        <int>     <dbl>  <dbl>
## 1      14800 570554       0.38            1      0.38   0.38
## 2      16669 567869       0.4             1      0.4    0.78
## 3      14744 542736       0.55            1      0.55   1.33
## 4      16554 540945       0.85            1      0.85   2.18
## 5      12748 538669       0.95            1      0.95   3.13
## 6      17230 539645       0.95            1      0.95   4.08
```

It seems many of the large transactions are refunded, so if we sum the revenue, we should be working with some reasonable numbers.

```
#many large transactions are refunded
#we sum the revenues
custSummary_2 <- customer_data %>%
  group_by(InvoiceNo, CustomerID, Country, date, month, year, hourOfDay,
dayOfWeek) %>%
  summarise(orderVal = sum(TotalCost)) %>%
  mutate(recent = Sys.Date() - date) %>%
  ungroup()

custSummary_2$recent <- as.character(custSummary_2$recent)
custSummary_2$recentDays <- sapply(custSummary_2$recent, FUN = function(x)
{strsplit(x, split = '[ ]')[[1]][1]})
custSummary_2$recentDays <- as.integer(custSummary_2$recentDays)
head(custSummary_2, n = 5)

## # A tibble: 5 x 11
##   InvoiceNo CustomerID Country date       month year  hourOfDay dayOfWeek
##   <chr>          <int> <fct>   <date>     <fct> <fct> <fct>     <ord>
## 1 536365         17850 United~ 2010-12-01 12    2010  8         Wed
## 2 536366         17850 United~ 2010-12-01 12    2010  8         Wed
## 3 536367         13047 United~ 2010-12-01 12    2010  8         Wed
## 4 536368         13047 United~ 2010-12-01 12    2010  8         Wed
## 5 536369         13047 United~ 2010-12-01 12    2010  8         Wed
## # ... with 3 more variables: orderVal <dbl>, recent <chr>, recentDays
<int>
```

The dataframe can provide us with the order value and date and time information for each transaction, that can be grouped by customerID.

```
customerSummary_3 <- custSummary_2 %>%
  group_by(CustomerID, Country) %>%
  summarise(orders = n_distinct(InvoiceNo), revenue = sum(orderVal),
meanRevenue = round(mean(orderVal), 2), medianRevenue = median(orderVal),
            mostDay = names(which.max(table(dayOfWeek))), mostHour =
names(which.max(table(hourOfDay))),
            recency = min(recentDays))%>% #the amount of days that a customer
has remained inactive
  ungroup()
head(customerSummary_3)

## # A tibble: 6 x 9
##   CustomerID Country orders revenue meanRevenue medianRevenue mostDay
mostHour
##        <int> <fct>    <int>   <dbl>       <dbl>         <dbl> <chr>
<chr>
## 1      12346 United~      1  77184.      77184.        77184. Tue     10
## 2      12347 Iceland      7   4310         616.          585. Tue     14
## 3      12348 Finland      4   1797.        449.          338. Tue     10
## 4      12349 Italy        1   1758.       1758.         1758. Mon     9
## 5      12350 Norway       1    334.        334.          334. Wed     16
## 6      12352 Norway       8   2506.        313.          281. Tue     14
## # ... with 1 more variable: recency <int>
```

We filter orders greater than 1 and revenue greater than 50 pounds. Our dataframe gives us a list of repeat customers and tells us their country, how many orders they have made, total revenue and average order value as well as the day of the week and the time of the day they most frequently place orders.

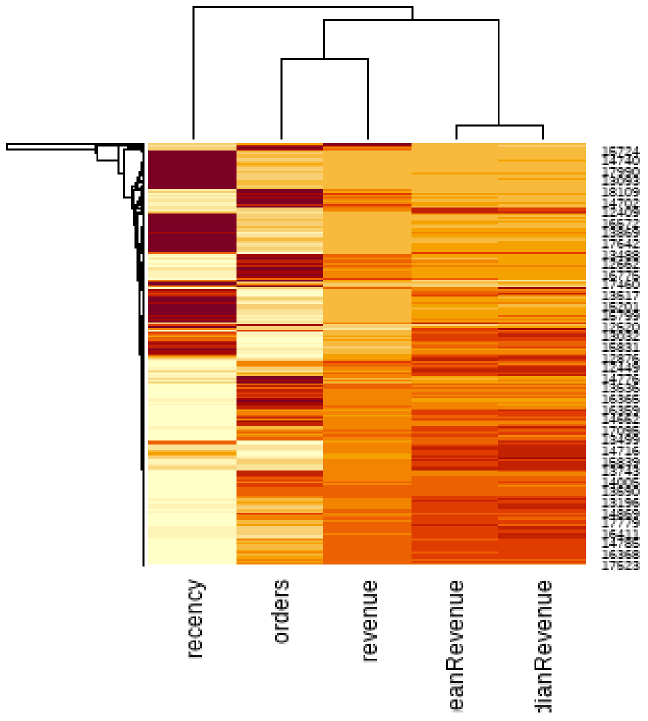
```
customerSummary_3Sum <- customerSummary_3 %>%
  filter(orders > 1, revenue > 50)
head(customerSummary_3Sum)

## # A tibble: 6 x 9
##   CustomerID Country orders revenue meanRevenue medianRevenue mostDay
mostHour
##        <int> <fct>    <int>   <dbl>       <dbl>         <dbl> <chr>
<chr>
## 1      12347 Iceland      7   4310         616.          585. Tue     14
## 2      12348 Finland      4   1797.        449.          338. Tue     10
## 3      12352 Norway       8   2506.        313.          281. Tue     14
## 4      12356 Portug~      3   2811.        937.          481. Tue     12
## 5      12358 Austria      2   1168.        584.          584. Tue     10
## 6      12359 Cyprus       4   6373.       1593.         1474. Mon     12
## # ... with 1 more variable: recency <int>

dim(customerSummary_3Sum) #We remain with a small subset (2,845)

## [1] 2845    9
```

We now remain with a small subset of 2,845 customers. From this, we are in a better position to answer a number of questions about the customers that we could use for targeted marketing strategies.

```
custTargets <- customerSummary_3Sum %>%
  select(recency, revenue, meanRevenue, medianRevenue, orders) %>%
  as.matrix()
rownames(custTargets) <- customerSummary_3Sum$CustomerID
head(custTargets)

##       recency revenue meanRevenue medianRevenue orders
## 12347    3742 4310.00      615.71       584.910      7
## 12348    3815 1797.24      449.31       338.500      4
## 12352    3776 2506.04      313.26       281.375      8
## 12356    3762 2811.43      937.14       481.460      3
## 12358    3741 1168.06      584.03       584.030      2
## 12359    3797 6372.58     1593.14      1474.115      4
```

By analyzing how customers cluster, we discover groups of customers that behave in similar ways. This level of customer segmentation is useful in marketing to these groups of customers appropriately. A marketing campaign that works for a group of customers that places low value orders frequently may not be appropriate for customers who place sporadic, high value orders for example. We use a heat map to visualize the customers recency, order and revenue scores. Higher scores are indicated by the darker areas in the heatmap.

```
#Generate a heatmap
options(repr.plot.width=20, repr.plot.height=14)
heatmap(scale(custTargets), cexCol = 0.9)
```



Recency refers to the number of days before the reference date when a customer made the last purchase. The lesser the value of recency, higher the likelihood the customer will visit the store.

Our clustering algorithm aims to keep the distance between data points in a cluster as little as possible relative to the distance between two clusters. Members of separate groups are very distinct whereas individuals of one group are quite similar.

From the heatmap above, it is evident that the total revenue clusters with the number of orders as we would expect. The mean and median order values cluster together, again this is expected, and lastly the order recency sits in its own group. The significant point here is how the customers rows cluster. We are able to uncover groups of customers that behave in similar ways. We have an idea about the clusters, and we now proceed to employ K-means Algorithm as our segmentation approach.

3. Results

This section presents the segmentation approach that was employed and the results obtained.

3.1 Segmentation Approach (Using K-means Algorithm)

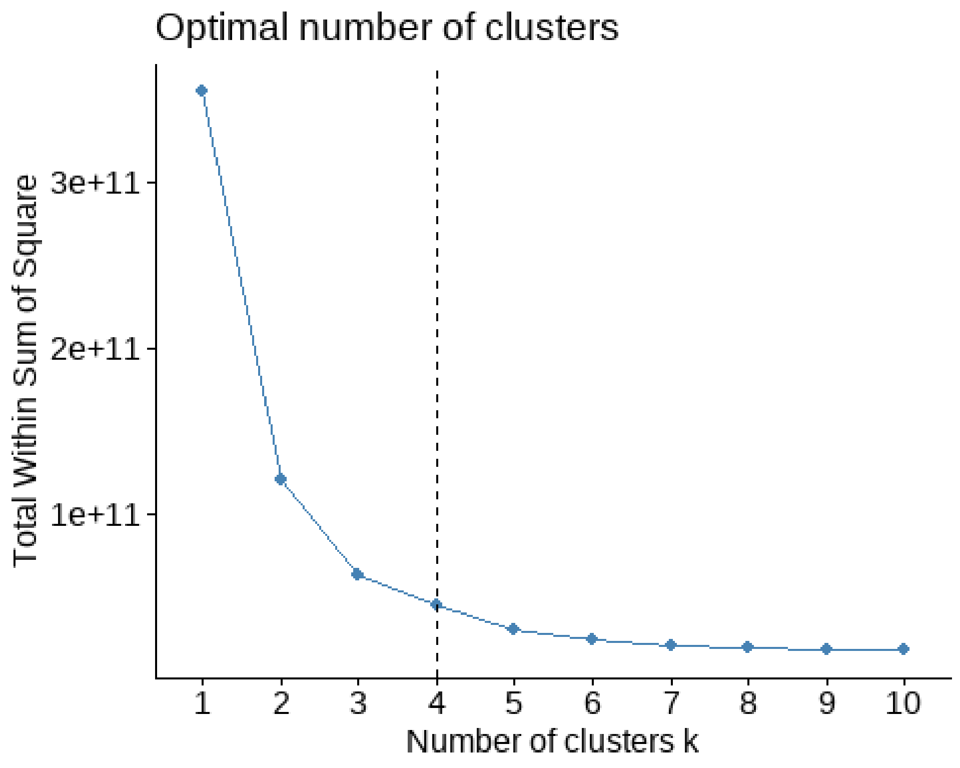
There are 3 main steps in K-Means algorithm (known also as Lloyd's algorithm):

- Split samples into initial groups by using seed points. The nearest samples to these seed point will create initial clusters.
- Calculate samples distances to groups' central points (centroids) and assign the nearest samples to their cluster.
- The third step is to calculate newly created (updated) cluster centroids.

Using Elbow Method

The main goal behind cluster partitioning methods like k-means is to define the clusters that maintain the intra-cluster variation at a minimum. The plot below denotes the appropriate number of clusters required in our model. In the plot, the location of a bend or a knee is the indication of the optimum number of clusters.

```r
set.seed(1, sample.kind="Rounding") #using R version 4.0.4`
#We plot optimal number of clusters using factoextra package
fviz_nbclust(custTargets, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2)
```



We use the kmeans function to come up with 4 clusters. We then attach the results to CustomersID to identify each customer's cluster.

```
clusters <- kmeans(scale(custTargets[,1:5]), 4, nstart = 1)
#Performing kmeans with 4 clusters. nstart > 1 is often recommended.
#Attaching the results to CustomersID
custTargets$Cluster <- as.factor(clusters$cluster)
custTargetsDf <- as.data.frame(custTargets) #convert matrix to dataframe
```

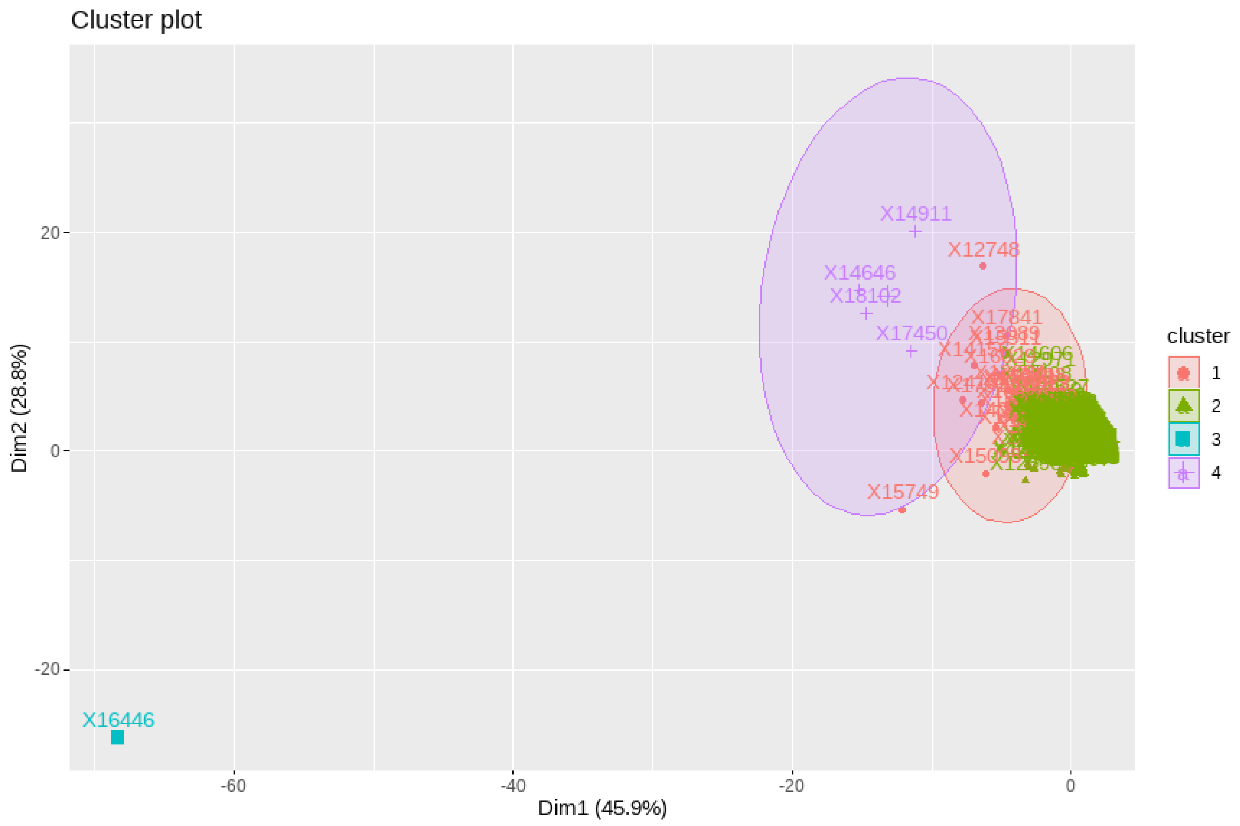
Based on the Elbow chart produced we conclude that 4 is the optimal number of clusters since it seems to be appearing at the bend in the elbow plot and this is our K value or an optimal number of clusters. Below are the cluster sizes:

```
#cluster sizes
clusters$size

## [1] 2336    1   25  483
```

Below are the cluster means.

```
#cluster means
clusters$centers

##      recency     revenue meanRevenue medianRevenue       orders
## 1 -0.3848591 -0.05000465 -0.02479490   -0.02075438 -0.003488075
## 2 -0.8309982 15.18328905 50.53173542   51.37302095 -0.440603035
## 3 -0.7503004  7.30808171  0.78941258    0.40998637  6.967489723
## 4  1.9019035 -0.16785606 -0.02556142   -0.02720589 -0.342854029
```

```
fviz_cluster(clusters, data=as.data.frame(custTargets)[, -6], ellipse.type =
"norm")
```



It is evident that from the results from the table and the chart above:

- Cluster 1 consist of 29 customers with high revenue.
- Cluster 2 represents 4 customers having a highest number of orders.
- Cluster 3 represents 1 customer who received very huge refunds (not significant).
- Cluster 4 comprises of 2,811 customers with highest recency.

4. Conclusion

We managed to identify three main segments of customers according to their revenue patterns, number of orders and recency which will helps target the customers based on their habits. One cluster which comprised of a single customer was not found to be significant for our case.

Through Kmeans clustering were able to segment customers to get a better understanding of them which in turn could be used to increase a company's revenue. Other clustering algorithms such as Silhouette and Gap statistic methods could be used to identify the the optimal number of clusters. For this project we restricted ourselves to the Elbow method only.

Further analysis using other segmentation approaches such as RFM Analysis, Principal Component Analysis (PCA) etc. can be conducted on the clusters to identify more narrowed characteristics of the customers, understand relationship between cluster and types of product purchased or predicting each cluster and customers lifetime value.