Application of Unsupervised in Business Cases

Obrein Telly

29/03/2022

CARREFOUR MARKETING STRATEGY

1. Introduction & Overview

a) Background

Carrefour Kenya is reviewing its marketing strategies and would like a data science driven approach to help realize the highest numbers of sales(total price tax included)

b) Approach

To answer this question, we will explore the data and leveraging the various unsupervised learning techniques derive insights and propose recommendations to the marketing team.
The project will be broken down into 4 parts. The first and second parts will deal with Dimensionality Reduction and Feature Selection.

c) Metric for Success
d) Implementation Design

2. Loading the Relevant Packages & Libraries

```
#install.packages(c("tidyverse", "ggplot2", "plotly", "dplR","DataExplorer","mice","VIM",
 #                  "lubridate", "Hmisc","GGally","moments", "ggcorrplot", "data.tables", "corrplot",
#"mlbench", "caret", "vqv/ggbiplot", "FSelector", 'arules', "Rtsne", "anomalize","tibbletime", "timetk"
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(DataExplorer)
library(ggplot2)
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(mice)
```

```
##
## Attaching package: 'mice'

## The following object is masked from 'package:stats':
##
##     filter

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(VIM)
```

```
## Loading required package: colorspace

## Loading required package: grid

## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(Hmisc)
```

```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following object is masked from 'package:plotly':
##
##     subplot

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(moments)
library(ggcorrplot)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(mlbench)
library(caret)
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(dplyr)
library(devtools)
```

```
## Loading required package: usethis
```

```
library(ggbiplot)
```

```
## Loading required package: plyr

## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:Hmisc':
##
##     is.discrete, summarize
```

```
## The following objects are masked from 'package:plotly':
##
##     arrange, mutate, rename, summarise

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact

## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(Rtsne)
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
library(anomalize)
```

```
## == Use anomalize to improve your Forecasts by 50%! =============================
## Business Science offers a 1-hour course - Lab #18: Time Series Anomaly Detection!
## </> Learn more at: https://university.business-science.io/p/learning-labs-pro </>
```

```r
library(tibbletime)
```

```
##
## Attaching package: 'tibbletime'

## The following object is masked from 'package:stats':
##
##     filter
```

```r
library(timetk)
```

```
##
## Attaching package: 'timetk'

## The following object is masked from 'package:data.table':
##
##     :=
```

```r
library(kdensity)


install.packages("dplyr")
library(clustvarsel)
```

```
## Loading required package: mclust

## Package 'mclust' version 5.4.9
## Type 'citation("mclust")' for citing this R package in publications.

##
## Attaching package: 'mclust'

## The following object is masked from 'package:VIM':
##
##     diabetes

## The following object is masked from 'package:purrr':
##
##     map

## Package 'clustvarsel' version 2.3.4

## Type 'citation("clustvarsel")' for citing this R package in publications.
```

```r
#install.packages("RapidMiner")

#remove.packages("FSelector")
#install.packages("FSelector", dependencies = TRUE)
```

3. Loading the Dataset

```r
# load the dataset using fread() function and instantiating it

df <- fread('http://bit.ly/CarreFourDataset')
```

4. Data Exploration

```r
#Previewing the first 6 records in the dataset
head(df)
```

```
##     Invoice ID Branch Customer type Gender           Product line Unit price
## 1: 750-67-8428      A        Member Female      Health and beauty      74.69
## 2: 226-31-3081      C        Normal Female Electronic accessories      15.28
## 3: 631-41-3108      A        Normal   Male     Home and lifestyle      46.33
## 4: 123-19-1176      A        Member   Male      Health and beauty      58.22
## 5: 373-73-7910      A        Normal   Male      Sports and travel      86.31
## 6: 699-14-3026      C        Normal   Male Electronic accessories      85.39
##    Quantity     Tax      Date  Time     Payment   cogs gross margin percentage
## 1:        7 26.1415  1/5/2019 13:08     Ewallet 522.83                4.761905
## 2:        5  3.8200  3/8/2019 10:29        Cash  76.40                4.761905
## 3:        7 16.2155  3/3/2019 13:23 Credit card 324.31                4.761905
## 4:        8 23.2880 1/27/2019 20:33     Ewallet 465.76                4.761905
## 5:        7 30.2085  2/8/2019 10:37     Ewallet 604.17                4.761905
## 6:        7 29.8865 3/25/2019 18:30     Ewallet 597.73                4.761905
##    gross income Rating    Total
## 1:      26.1415    9.1 548.9715
## 2:       3.8200    9.6  80.2200
## 3:      16.2155    7.4 340.5255
## 4:      23.2880    8.4 489.0480
## 5:      30.2085    5.3 634.3785
## 6:      29.8865    4.1 627.6165
```

```r
#Checking the columns names
colnames(df)
```

```
##  [1] "Invoice ID"              "Branch"
##  [3] "Customer type"           "Gender"
##  [5] "Product line"            "Unit price"
##  [7] "Quantity"                "Tax"
##  [9] "Date"                    "Time"
## [11] "Payment"                 "cogs"
## [13] "gross margin percentage" "gross income"
## [15] "Rating"                  "Total"
```

```r
## checking the Data types of each variable using sapply() function
sapply(df, class)
```

```
##              Invoice ID                  Branch           Customer type
##             "character"             "character"             "character"
##                  Gender            Product line              Unit price
##             "character"             "character"               "numeric"
##                Quantity                     Tax                    Date
##               "integer"               "numeric"             "character"
##                    Time                 Payment                    cogs
##             "character"             "character"               "numeric"
## gross margin percentage            gross income                  Rating
##               "numeric"               "numeric"               "numeric"
##                   Total
##               "numeric"
```

```r
#Previewing Columns & confirming values are expected

#Gender
unique(df$Gender)
```

```
## [1] "Female" "Male"
```

```r
#Quality
unique(df$Quantity)
```

```
##  [1]  7  5  8  6 10  2  3  4  1  9
```

```r
unique(df$`Product line`)
```

```
## [1] "Health and beauty"      "Electronic accessories" "Home and lifestyle"
## [4] "Sports and travel"      "Food and beverages"     "Fashion accessories"
```

```r
unique(df$GROSS_MARGIN_PERCENTAGE)
```

```
## NULL
```

```r
#Values are expected
```

5. Data Cleaning

```r
#Create Uniform Case of the columns

names(df) <- toupper(names(df))
# replace the spaces with underscores
names(df) <- gsub(" ","_", names(df))
str(df)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  16 variables:
##  $ INVOICE_ID             : chr  "750-67-8428" "226-31-3081" "631-41-3108" "123-19-1176" ...
##  $ BRANCH                 : chr  "A" "C" "A" "A" ...
##  $ CUSTOMER_TYPE          : chr  "Member" "Normal" "Normal" "Member" ...
##  $ GENDER                 : chr  "Female" "Female" "Male" "Male" ...
##  $ PRODUCT_LINE           : chr  "Health and beauty" "Electronic accessories" "Home and lifestyle" "I
##  $ UNIT_PRICE             : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ QUANTITY               : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ TAX                    : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ DATE                   : chr  "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ TIME                   : chr  "13:08" "10:29" "13:23" "20:33" ...
##  $ PAYMENT                : chr  "Ewallet" "Cash" "Credit card" "Ewallet" ...
##  $ COGS                   : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ GROSS_MARGIN_PERCENTAGE: num  4.76 4.76 4.76 4.76 4.76 ...
##  $ GROSS_INCOME           : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ RATING                 : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
##  $ TOTAL                  : num  549 80.2 340.5 489 634.4 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
#Checking for Duplicates

any(duplicated(df))
```

```
## [1] FALSE
```

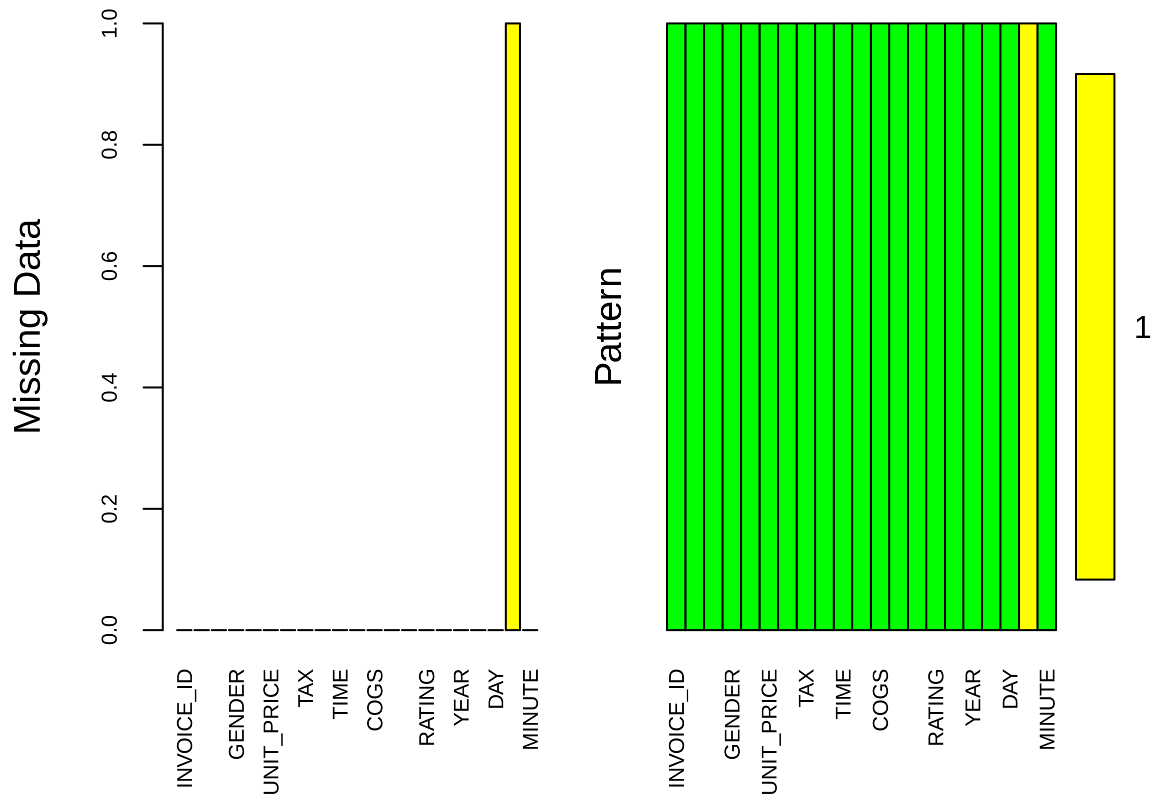
```r
#There are no duplicates in the dataset


#Creating Year, Month and Day Columns
df$DATE <- as.Date(df$DATE, "%m/%d/%Y")


df$YEAR <- year(ymd(df$DATE))

df$MONTH <- month(ymd(df$DATE))

df$DAY <- day(ymd(df$DATE))


#Unique values in Year
unique(df$YEAR)
```

```
## [1] 2019
```

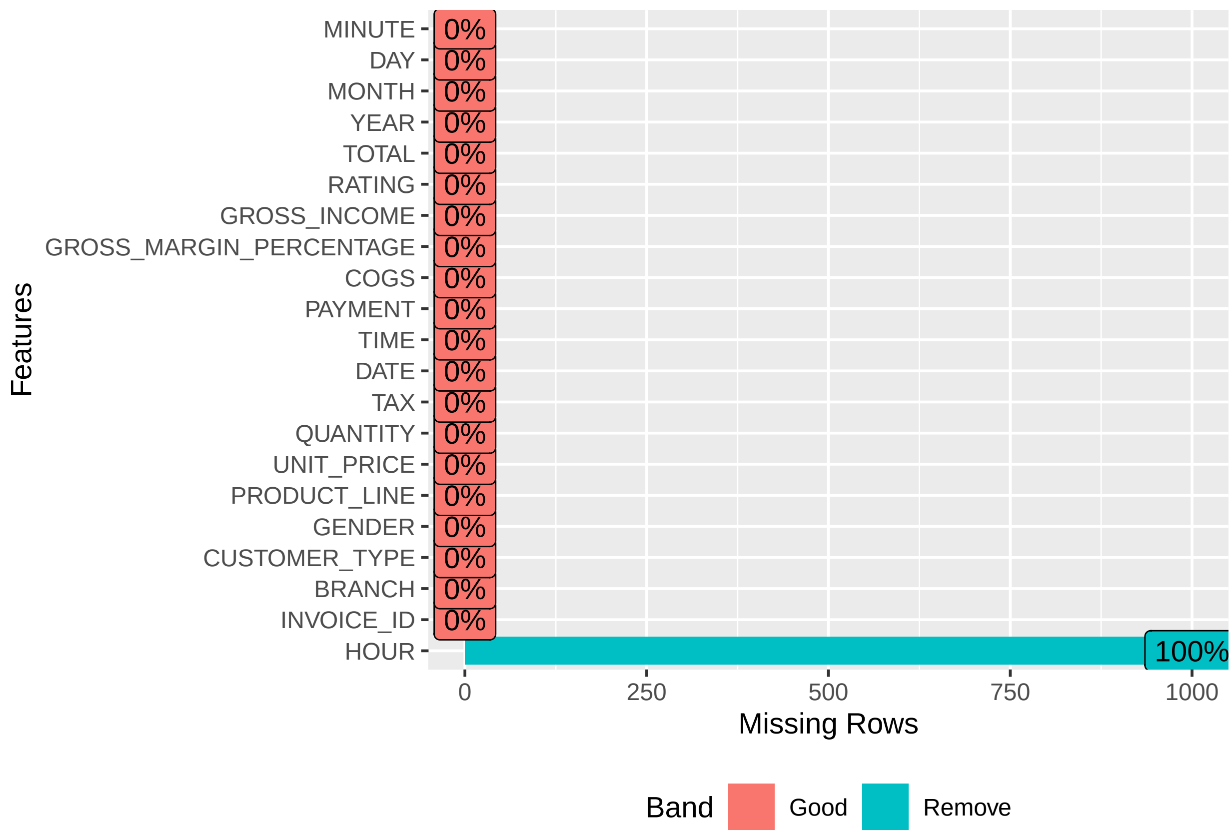
```r
# All the datapoints relate to 2019.

#Separate Hours and Minutes
df$HOUR <- format(strptime(df$TIME, "%H:M"), "%H")

df$MINUTE <- format(strptime(df$TIME, "%H:%M"), "%M")

str(df)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  21 variables:
##  $ INVOICE_ID             : chr  "750-67-8428" "226-31-3081" "631-41-3108" "123-19-1176" ...
##  $ BRANCH                 : chr  "A" "C" "A" "A" ...
##  $ CUSTOMER_TYPE          : chr  "Member" "Normal" "Normal" "Member" ...
##  $ GENDER                 : chr  "Female" "Female" "Male" "Male" ...
##  $ PRODUCT_LINE           : chr  "Health and beauty" "Electronic accessories" "Home and lifestyle" "
##  $ UNIT_PRICE             : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ QUANTITY               : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ TAX                    : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ DATE                   : Date, format: "2019-01-05" "2019-03-08" ...
##  $ TIME                   : chr  "13:08" "10:29" "13:23" "20:33" ...
##  $ PAYMENT                : chr  "Ewallet" "Cash" "Credit card" "Ewallet" ...
##  $ COGS                   : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ GROSS_MARGIN_PERCENTAGE: num  4.76 4.76 4.76 4.76 4.76 ...
##  $ GROSS_INCOME           : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ RATING                 : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
##  $ TOTAL                  : num  549 80.2 340.5 489 634.4 ...
##  $ YEAR                   : int  2019 2019 2019 2019 2019 2019 2019 2019 2019 2019 ...
##  $ MONTH                  : int  1 3 3 1 2 3 2 2 1 2 ...
##  $ DAY                    : int  5 8 3 27 8 25 25 24 10 20 ...
##  $ HOUR                   : chr  NA NA NA NA ...
##  $ MINUTE                 : chr  "08" "29" "23" "33" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
#Checking for Missing Values using the aggr() function
mp <- aggr(df, col= c('green', 'yellow'),
numbers= TRUE, sortvars=TRUE, labels= names(df), cex.axis =.7, gap=3, ylab=c("Missing Data", "Pattern"))
```




```r
#Distribution of Missing Values using plot_missing function in DataExplorer

plot_missing(df)
```




```
#There are missing values in the Hour Column. We drop it alongside DATE & TIME as we have
```

```
#Split the original values into other columns. We also drop the YEAR column as all entries are
#in 2019


df <- df <- select(df, -c(INVOICE_ID, HOUR, MINUTE, DATE, TIME, YEAR, GROSS_MARGIN_PERCENTAGE))


#Checking there are no missing values

colSums(is.na(df),)
```

```
##        BRANCH CUSTOMER_TYPE        GENDER  PRODUCT_LINE    UNIT_PRICE
##             0             0             0             0             0
##      QUANTITY           TAX       PAYMENT          COGS  GROSS_INCOME
##             0             0             0             0             0
##        RATING         TOTAL         MONTH           DAY
##             0             0             0             0
```

```
#There are no missing values in the dataset



#Factorizing erroneously classed dtypes
```

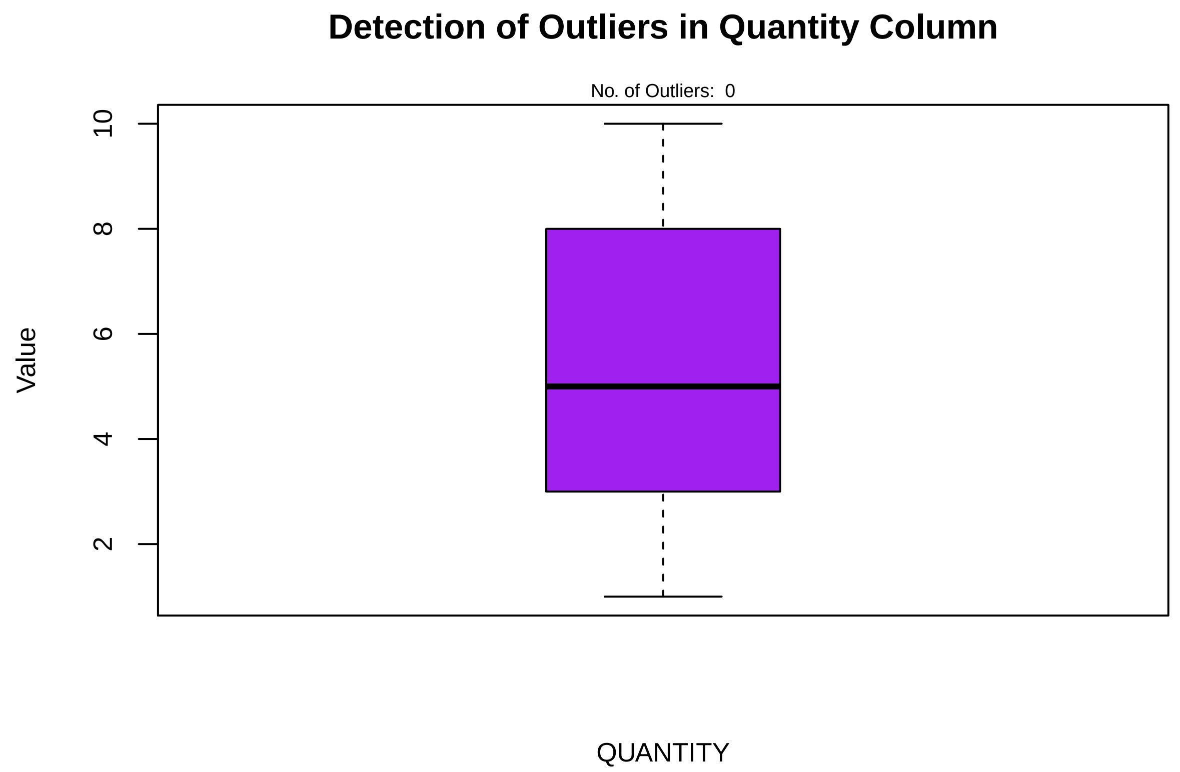
```r
df$BRANCH <- factor(df$BRANCH)

df$GENDER <- factor(df$GENDER)

df$PRODUCT_LINE <- factor(df$PRODUCT_LINE)

df$QUANTITY <- as.integer(factor(df$QUANTITY))

df$PAYMENT <- factor(df$PAYMENT)

df$RATING <- as.integer(factor(df$RATING))

df$CUSTOMER_TYPE <- factor(df$CUSTOMER_TYPE)


#Deriving the Numerical Variables

str(df)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  14 variables:
##  $ BRANCH       : Factor w/ 3 levels "A","B","C": 1 3 1 1 1 3 1 3 1 2 ...
##  $ CUSTOMER_TYPE: Factor w/ 2 levels "Member","Normal": 1 2 2 1 2 2 1 2 1 1 ...
##  $ GENDER       : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 2 1 1 1 1 ...
##  $ PRODUCT_LINE : Factor w/ 6 levels "Electronic accessories",..: 4 1 5 4 6 1 1 5 4 3 ...
##  $ UNIT_PRICE   : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ QUANTITY     : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ TAX          : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ PAYMENT      : Factor w/ 3 levels "Cash","Credit card",..: 3 1 2 3 3 3 3 3 2 2 ...
##  $ COGS         : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ GROSS_INCOME : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ RATING       : int  52 57 35 45 14 2 19 41 33 20 ...
##  $ TOTAL        : num  549 80.2 340.5 489 634.4 ...
##  $ MONTH        : int  1 3 3 1 2 3 2 2 1 2 ...
##  $ DAY          : int  5 8 3 27 8 25 25 24 10 20 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

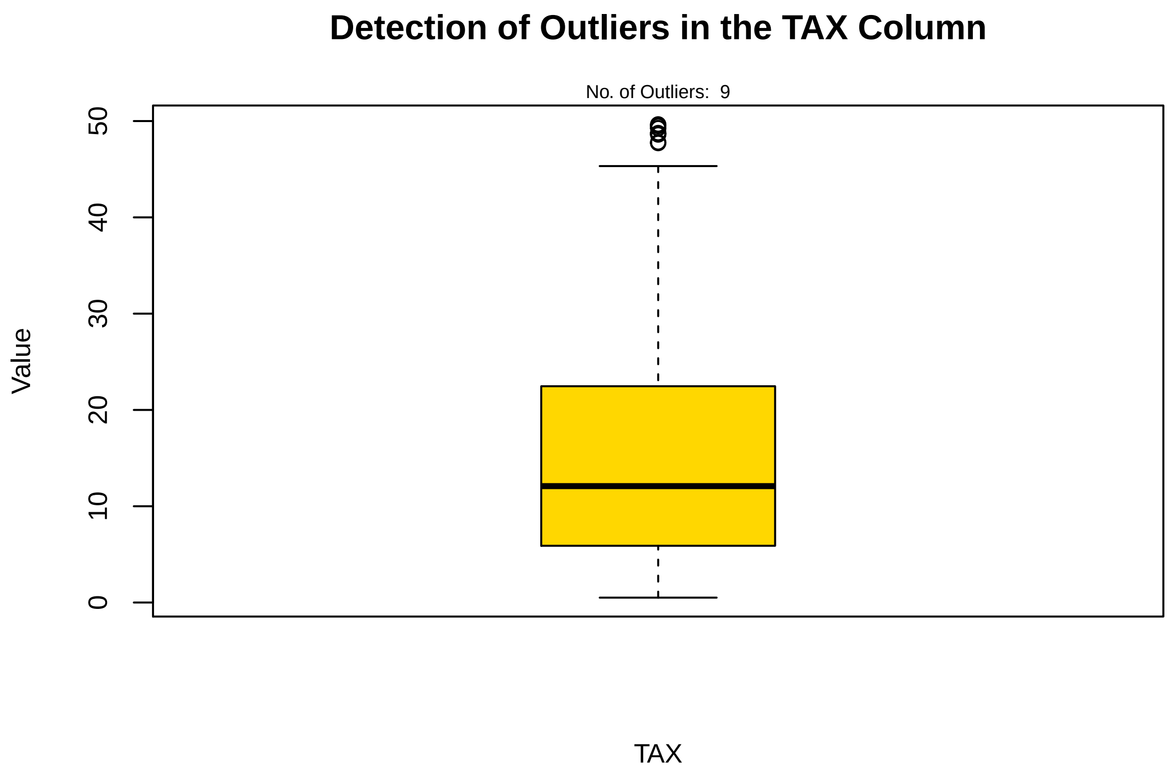
```r
num.cols <- select(df, c(UNIT_PRICE, TAX, COGS, TOTAL, GROSS_INCOME ))

categ.cols <- select(df, c(BRANCH, CUSTOMER_TYPE, GENDER, PRODUCT_LINE, PAYMENT))

str(categ.cols)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  5 variables:
##  $ BRANCH       : Factor w/ 3 levels "A","B","C": 1 3 1 1 1 3 1 3 1 2 ...
##  $ CUSTOMER_TYPE: Factor w/ 2 levels "Member","Normal": 1 2 2 1 2 2 1 2 1 1 ...
##  $ GENDER       : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 2 1 1 1 1 ...
##  $ PRODUCT_LINE : Factor w/ 6 levels "Electronic accessories",..: 4 1 5 4 6 1 1 5 4 3 ...
##  $ PAYMENT      : Factor w/ 3 levels "Cash","Credit card",..: 3 1 2 3 3 3 3 3 2 2 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

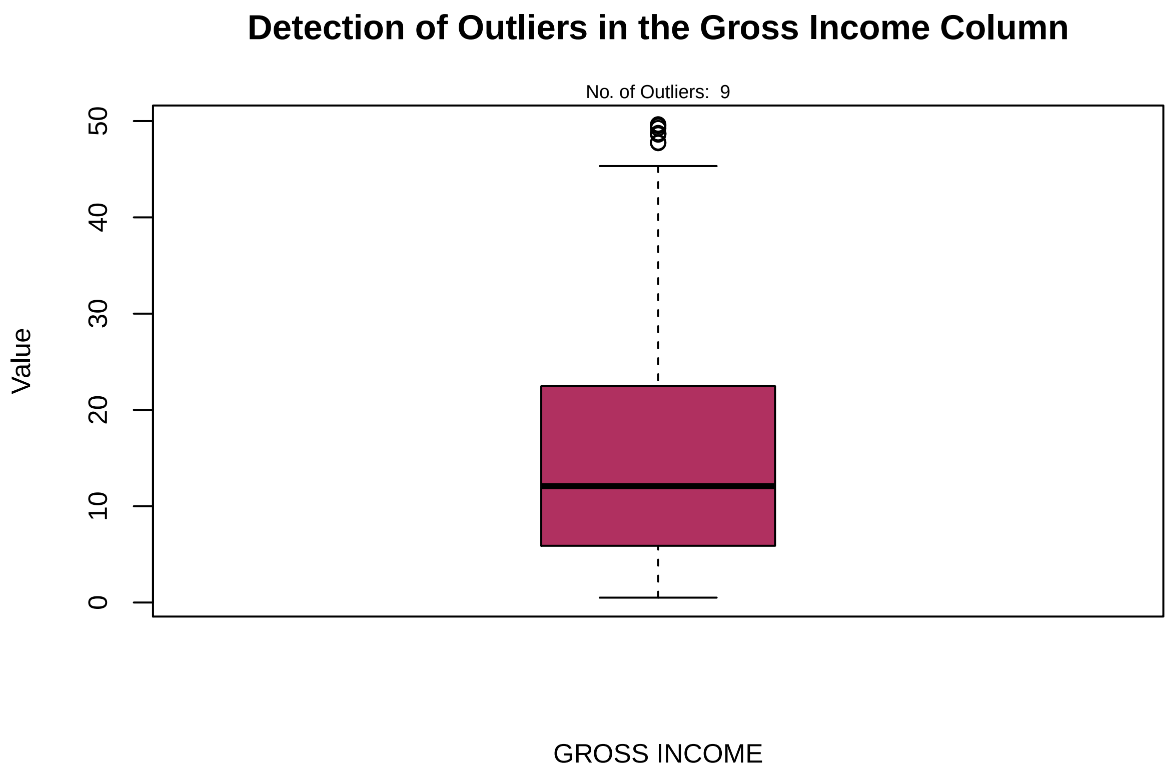
```r
#Outlier Detection in the Quantity Attribute using Boxplot

options(repr.plot.width = 11, repr.plot.height =9)

boxplot(df$QUANTITY, main="Detection of Outliers in Quantity Column", xlab = "QUANTITY", ylab = "Value"

quantity_outlier <- boxplot.stats(df$QUANTITY)$out
mtext(paste("No. of Outliers: ", paste(length(quantity_outlier), collapse=", ")), cex=0.6)
```



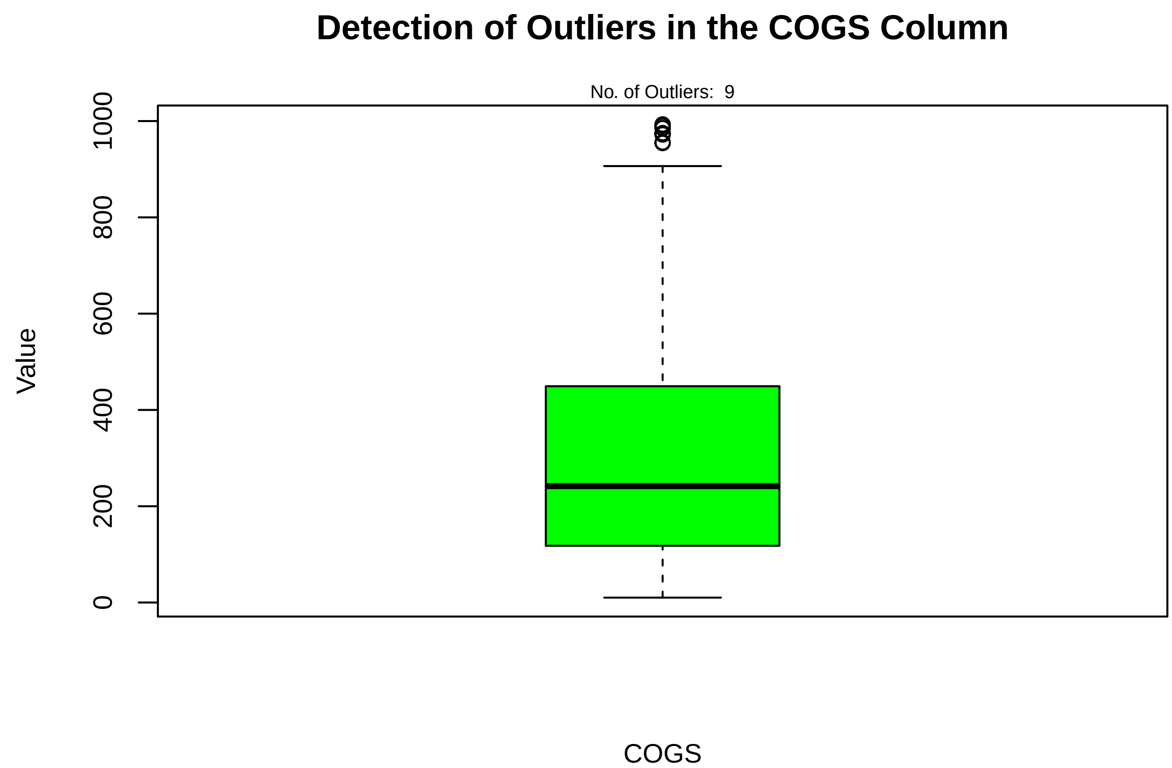

```r
#NO Outliers in this COlumn

#Outlier Detection in the TAX Attribute using Boxplot

options(repr.plot.width = 12, repr.plot.height =10)

boxplot(df$TAX, main="Detection of Outliers in the TAX Column", xlab = "TAX", ylab = "Value", boxwex=0.

tax_outlier <- boxplot.stats(df$TAX)$out
mtext(paste("No. of Outliers: ", paste(length(tax_outlier), collapse=", ")), cex=0.6)
```



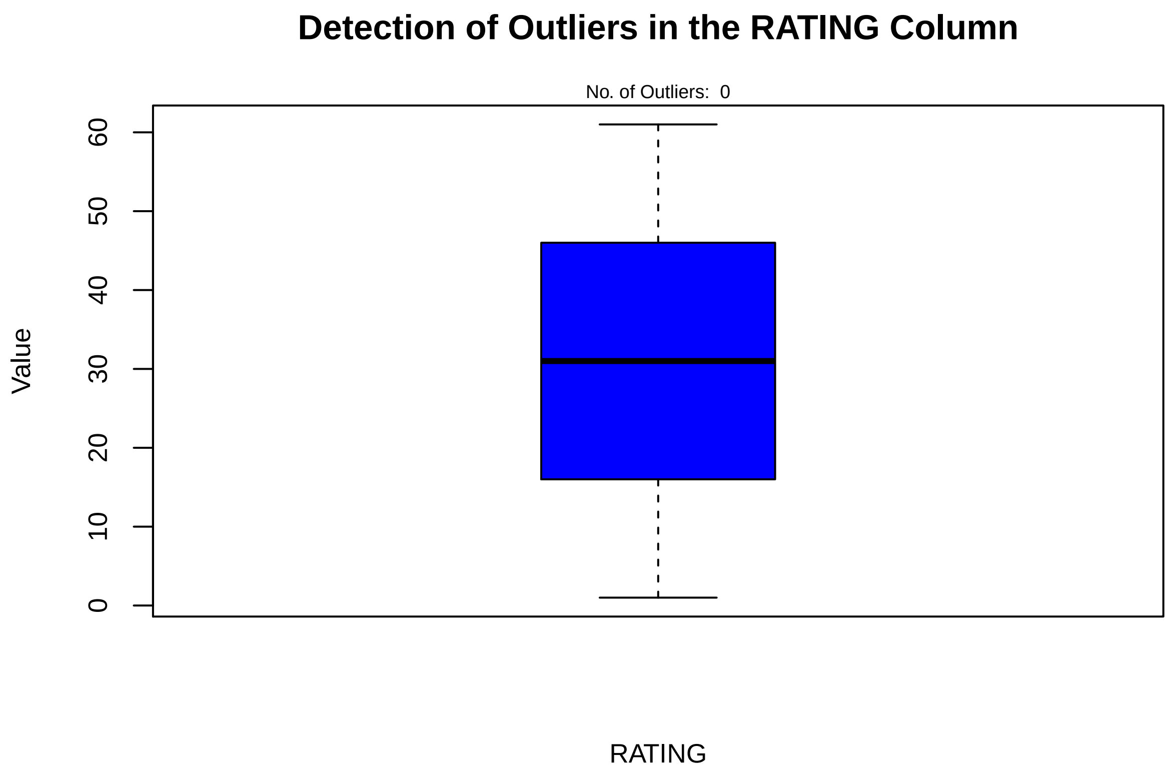

```r
#There are 9 Outliers in this column


#Outlier Detection in the Gross Income Attribute using Boxplot & length()

options(repr.plot.width = 12, repr.plot.height =10)

boxplot(df$GROSS_INCOME, main="Detection of Outliers in the Gross Income Column", xlab = "GROSS INCOME"

income_outlier <- boxplot.stats(df$GROSS_INCOME)$out
mtext(paste("No. of Outliers: ", paste(length(income_outlier), collapse=", ")), cex=0.6)
```



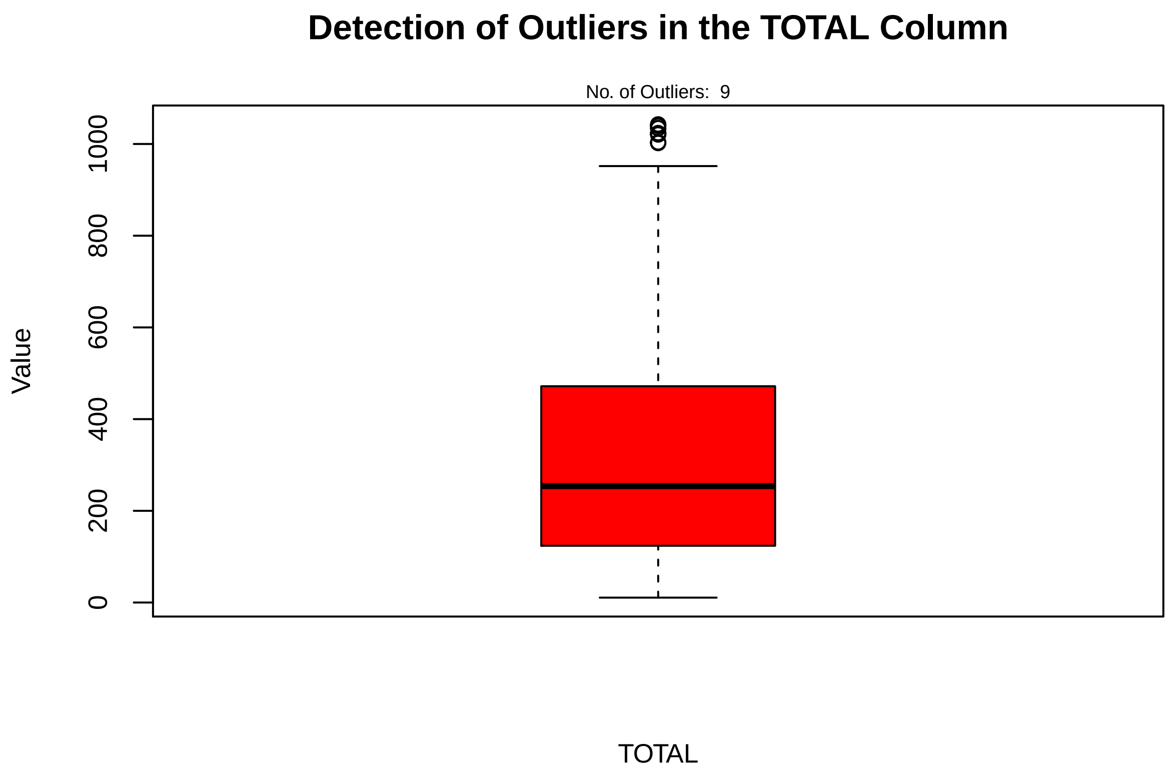

```
#There are about 9 outlier data points


#Outlier Detection in the COGS Attribute using Boxplot

options(repr.plot.width = 12, repr.plot.height =10)

boxplot(df$COGS, main="Detection of Outliers in the COGS Column", xlab = "COGS", ylab = "Value", boxwex=

cogs_outlier <- boxplot.stats(df$COGS)$out
mtext(paste("No. of Outliers: ", paste(length(cogs_outlier), collapse=", ")), cex=0.6)
```




```
# There are 9 outliers in the COGS COlumn


#Outlier Detection in the RATING Attribute using Boxplot

options(repr.plot.width = 12, repr.plot.height =10)

boxplot(df$RATING, main="Detection of Outliers in the RATING Column", xlab = "RATING", ylab = "Value", 

rating_outlier <- boxplot.stats(df$RATING)$out
mtext(paste("No. of Outliers: ", paste(length(rating_outlier), collapse=", ")), cex=0.6)
```




```
#No Outliers in the Ratings Column



#Outlier Detection in the TOTAL Column using Boxplot

options(repr.plot.width = 12, repr.plot.height =10)

boxplot(df$TOTAL, main="Detection of Outliers in the TOTAL Column", xlab = "TOTAL", ylab = "Value", boxw

total_outlier <- boxplot.stats(df$TOTAL)$out
mtext(paste("No. of Outliers: ", paste(length(total_outlier), collapse=", ")), cex=0.6)
```




```
#We have around 9 Outliers in the Totals Column


#Despite the presence of Outliers in some of the numerical COlumns, we still keep them in the
#dataset as they represent actual datapoints and we do not have a good reason to drop them.
```

```
#Apportioning the MOnth Column into features that can be analyzed
df$TIME_MONTH = ifelse(df$DAY >= 1 & df$DAY <= 10, "Early-Month",
               ifelse(df$DAY >= 11 & df$DAY <= 20, "Mid-Month", "End-Month"))


str(df)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  15 variables:
##  $ BRANCH       : Factor w/ 3 levels "A","B","C": 1 3 1 1 1 3 1 3 1 2 ...
##  $ CUSTOMER_TYPE: Factor w/ 2 levels "Member","Normal": 1 2 2 1 2 2 1 2 1 1 ...
##  $ GENDER       : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 2 1 1 1 1 ...
##  $ PRODUCT_LINE : Factor w/ 6 levels "Electronic accessories",..: 4 1 5 4 6 1 1 5 4 3 ...
##  $ UNIT_PRICE   : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ QUANTITY     : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ TAX          : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ PAYMENT      : Factor w/ 3 levels "Cash","Credit card",..: 3 1 2 3 3 3 3 3 2 2 ...
##  $ COGS         : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ GROSS_INCOME : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ RATING       : int  52 57 35 45 14 2 19 41 33 20 ...
```

```
##  $ TOTAL        : num  549 80.2 340.5 489 634.4 ...
##  $ MONTH        : int  1 3 3 1 2 3 2 2 1 2 ...
##  $ DAY          : int  5 8 3 27 8 25 25 24 10 20 ...
##  $ TIME_MONTH   : chr  "Early-Month" "Early-Month" "Early-Month" "End-Month" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
unique(df$DAY)
```

```
##  [1]  5  8  3 27 25 24 10 20  6  9 12  7 29 15 11  1 21 17  2 22 28 23  4 16 19
## [26] 14 13 26 18 30 31
```

```
unique(df$MONTH)
```

```
## [1] 1 3 2
```

```
unique(df$TIME_MONTH)
```

```
## [1] "Early-Month" "End-Month"   "Mid-Month"
```

```
unique(df$MINUTE)
```

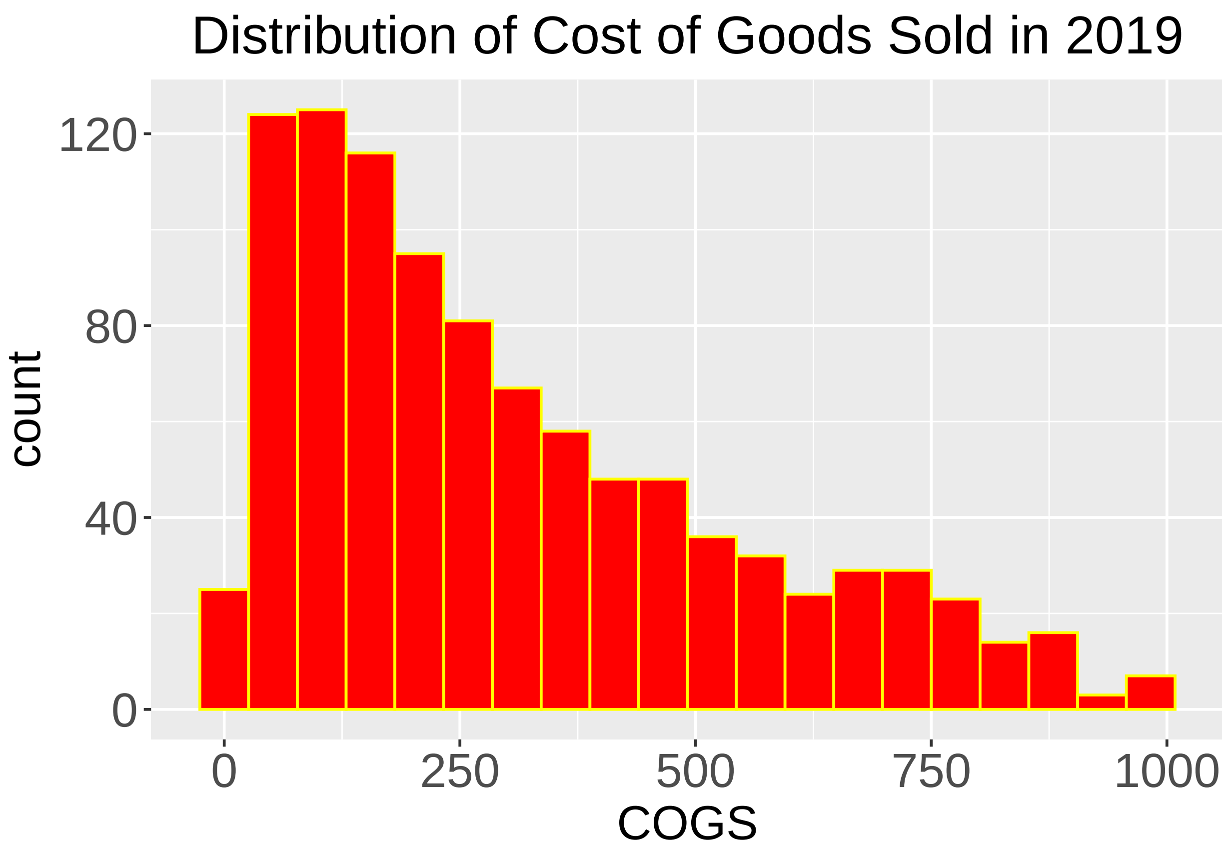
```
## NULL
```

6. EXPLORATORY DATA ANALYSIS (EDA)

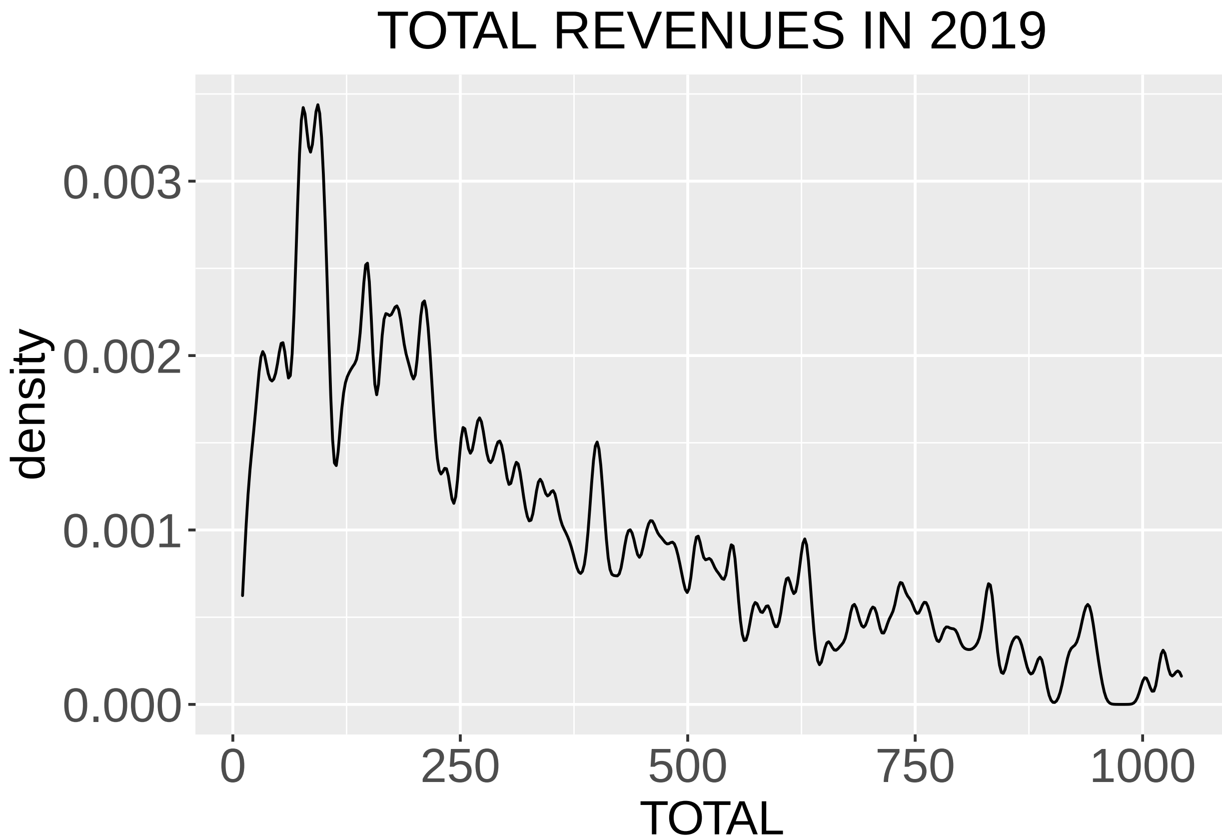
```
#Distribution of Branches in the Dataset
#
# branch_perc <- df %>%
#   filter(BRANCH != "NA") %>%
#   group_by(BRANCH) %>%
#   count() %>%
#   ungroup() %>%
#   arrange(desc(BRANCH)) %>%
#   mutate(percentage = round(n / sum(n), 4)*100 , lab.pos = cumsum(percentage)- 0.5 * percentage)
# ggplot(branch_perc, aes(x = "", y= percentage, fill = BRANCH)) +
#   geom_bar(stat = "identity") +
#   coord_polar("y", start = 200) +
#   geom_text(aes(y = lab.pos, label = paste(percentage,"%", sep = "")), col = "black") +
#   theme_void() + scale_fill_brewer(palette = "PuOr") + labs(title= "Distribution of Branches in the D
#   theme(plot.title = element_text(hjust = 0.4, size = 20))
#
# # Distribution of Gender in the Dataset(round off to 4 dec points)
# gender_perc <- df %>%
#   filter(GENDER != "NA") %>%
#   group_by(GENDER) %>%
#   count() %>%
#   ungroup() %>%
#   arrange(desc(GENDER)) %>%
#   mutate( percentage = round(n/sum(n), 4)*100, lab.pos = cumsum(percentage)- 0.5 * percentage)
# ggplot(gender_perc, aes(x = "", y= percentage, fill = GENDER)) +
```

```r
#   geom_bar(stat = "identity")+
#   coord_polar("y", start = 200) +
#   geom_text(aes(y = lab.pos, label = paste(percentage,"%", sep = "")), col = "black") +
#   theme_void() + scale_fill_brewer(palette = "PRGn") + labs(title= "Distribution of Gender in the Dat
#   theme(plot.title = element_text(hjust = 0.4, size = 20))
#
# #Distribution of Payments in the 2019 Datset
# payment_perc <- df %>%
#   filter(PAYMENT != "NA") %>%
#   group_by(PAYMENT) %>%
#   count() %>%
#   ungroup() %>%
#   arrange(desc(PAYMENT)) %>%
#   mutate( percentage = round(n/sum(n), 4)*100, lab.pos = cumsum(percentage)- 0.5 * percentage)
# ggplot(payment_perc, aes(x = "", y= percentage, fill = PAYMENT)) +
#   geom_bar(stat = "identity")+
#   coord_polar("y", start = 200) +
#   geom_text(aes(y = lab.pos, label = paste(percentage,"%", sep = "")), col = "black") +
#   theme_void() + scale_fill_brewer(palette = "Spectral") + labs(title= "Distribution of Payment Types
#   theme(plot.title = element_text(hjust = 0.4, size = 20))
#
# #Distribution of Product Lines in the Dataset
# productline_perc <- df %>%
#   filter(PRODUCT_LINE != "NA") %>%
#   group_by(PRODUCT_LINE) %>%
#   count() %>%
#   ungroup() %>%
#   arrange(desc(PRODUCT_LINE)) %>%
#   mutate(percentage = round(n/sum(n), 4)*100, lab.pos = cumsum(percentage)- 0.5 * percentage)
# ggplot(productline_perc, aes(x = "", y= percentage, fill = PRODUCT_LINE)) +
#   geom_bar(stat = "identity")+
#   coord_polar("y", start = 200) +
#   geom_text(aes(y = lab.pos, label = paste(percentage,"%", sep = "")), col = "black") +
#   theme_void() + scale_fill_brewer(palette = "Set1") + labs(title= "Distribution of Product Lines in
#   theme(plot.title = element_text(hjust = 0.4, size = 20))
#
```

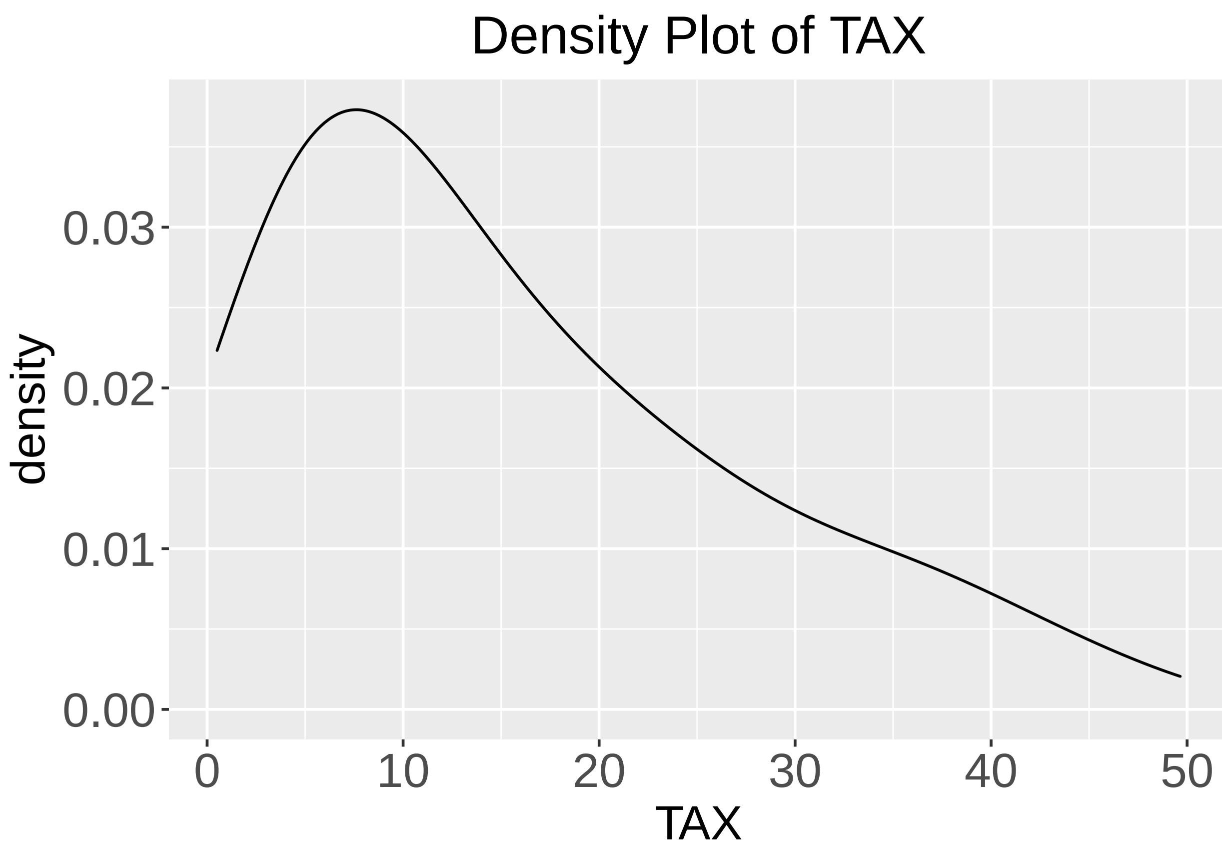
```r
# DIstribution of Cost of Goods Sold
ggplot(df, aes(x = COGS)) +

geom_histogram( bins = 20, color='yellow', fill = 'red') +

ggtitle("Distribution of Cost of Goods Sold in 2019") +

 theme(axis.text = element_text(size=18),
          axis.title = element_text(size = 18),
          plot.title = element_text(hjust = 0.5, size = 20))
```



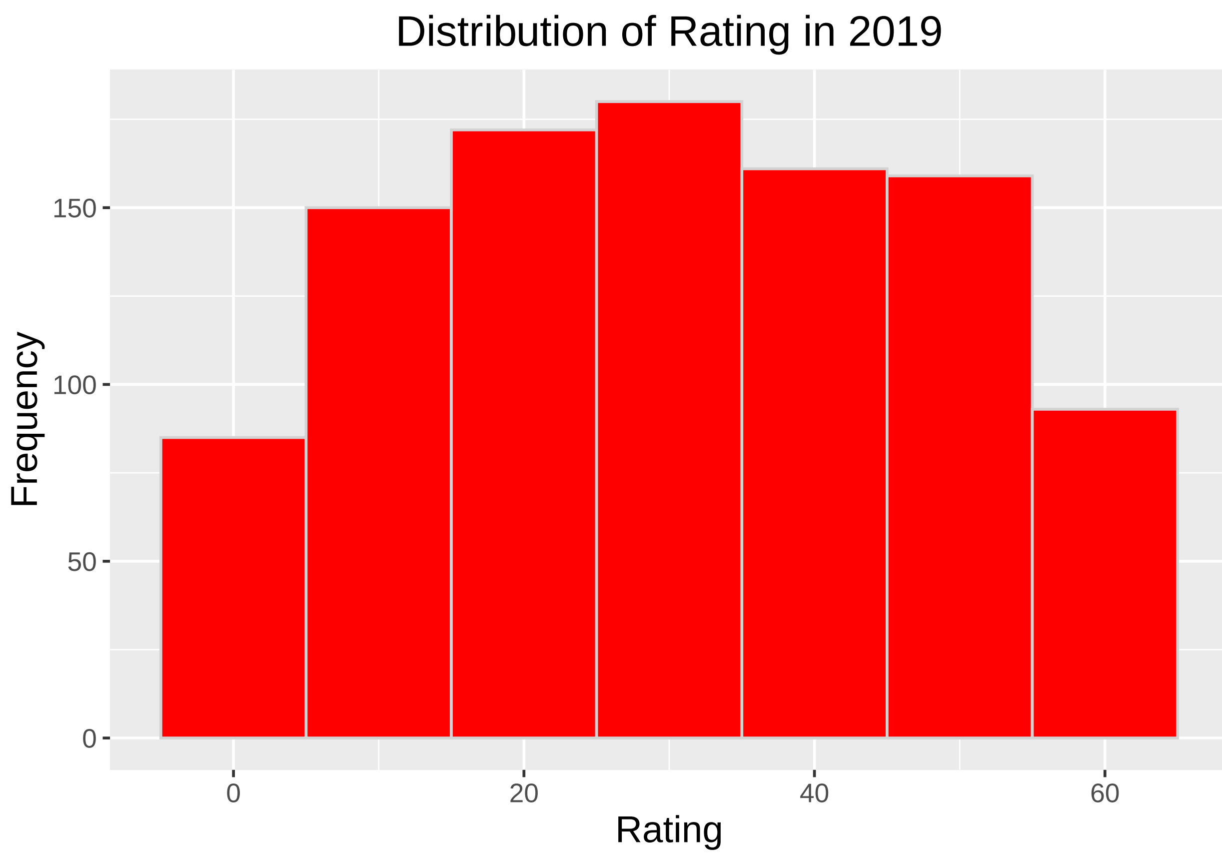

```
#Density Plot Distribution of the Total Column
ggplot(df, aes(x= TOTAL)) +
geom_density(bw = 5) +
ggtitle("TOTAL REVENUES IN 2019") +
 theme(axis.text = element_text(size=18),
          axis.title = element_text(size = 18),
          plot.title = element_text(hjust = 0.5, size = 20))
```



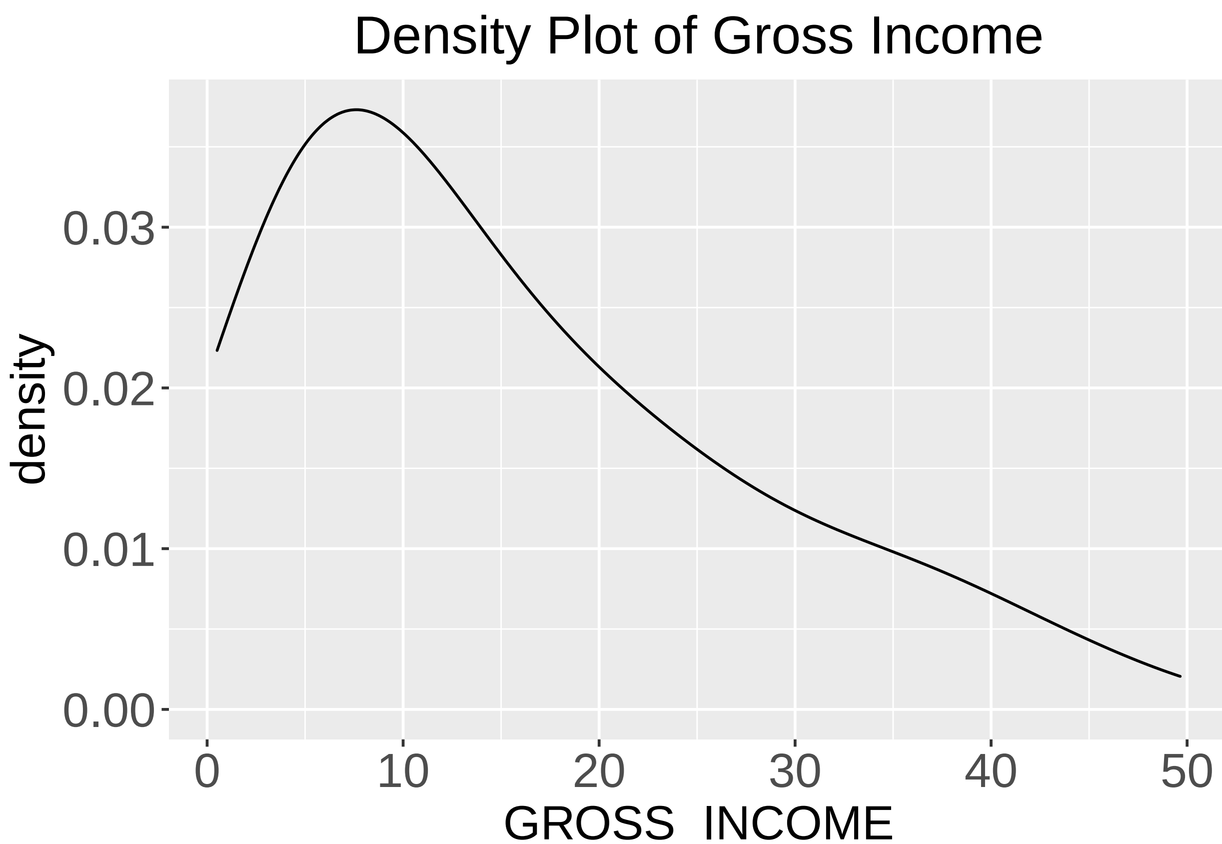

```
#Density Plot Distribution of the Tax Column
ggplot(df, aes(x= TAX)) +
geom_density(bw = 5) +
ggtitle("Density Plot of TAX") +
 theme(axis.text = element_text(size=18),
          axis.title = element_text(size = 18),
          plot.title = element_text(hjust = 0.5, size = 20))
```



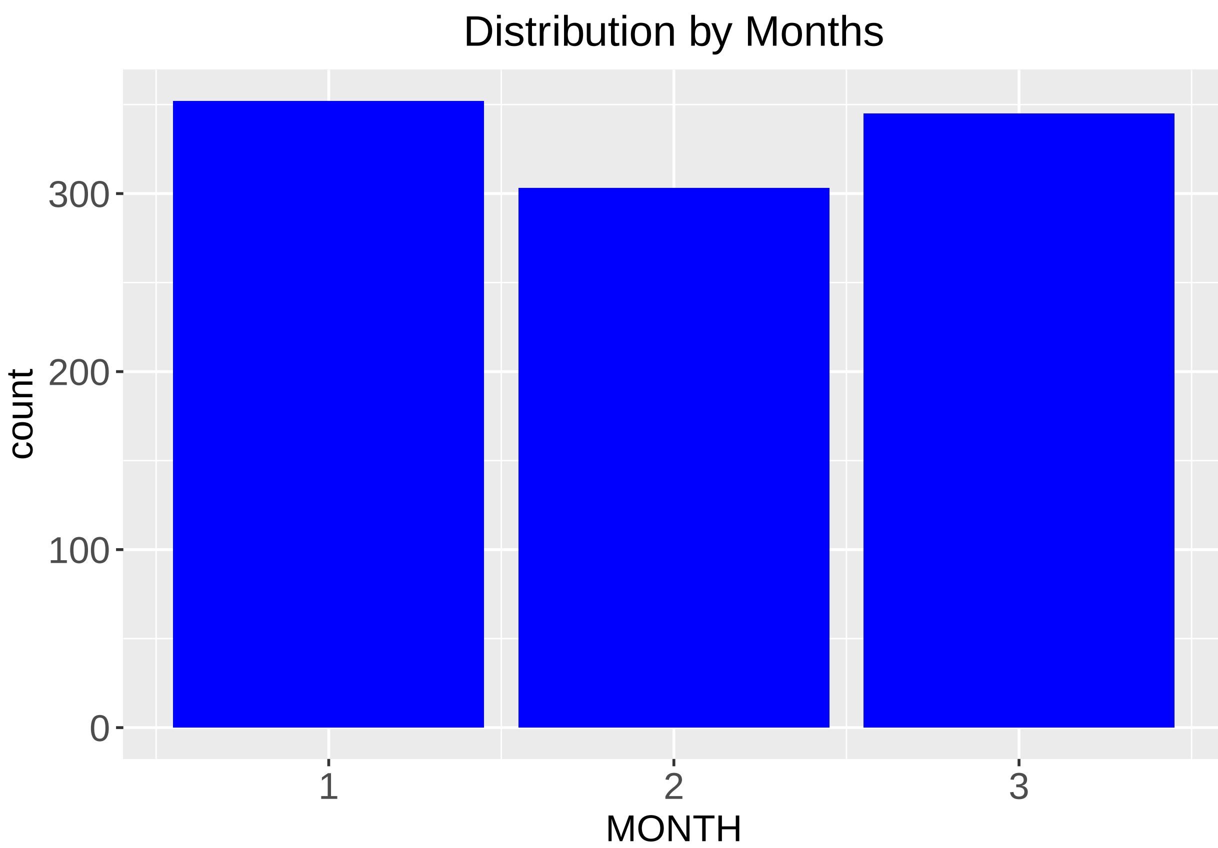

```r
# Histogram illustrating the distribution of values in the Rating column

options(repr.plot.width = 11, repr.plot.height = 9)
df %>% ggplot(aes(x = RATING )) +

geom_histogram(color="lightgray", fill="red", binwidth = 10) +
    labs(title = "Distribution of Rating in 2019", x = "Rating", y = "Frequency") +
    theme(axis.title = element_text(size = 14),
          axis.text = element_text(size=10),
          plot.title = element_text(hjust = 0.5, size = 16))
```




```
#Density Plot Distribution of the Income Column
ggplot(df, aes(x= GROSS_INCOME)) +
geom_density(bw = 5) +
ggtitle("Density Plot of Gross Income") +
 theme(axis.text = element_text(size=18),
          axis.title = element_text(size = 18),
          plot.title = element_text(hjust = 0.5, size = 20))
```



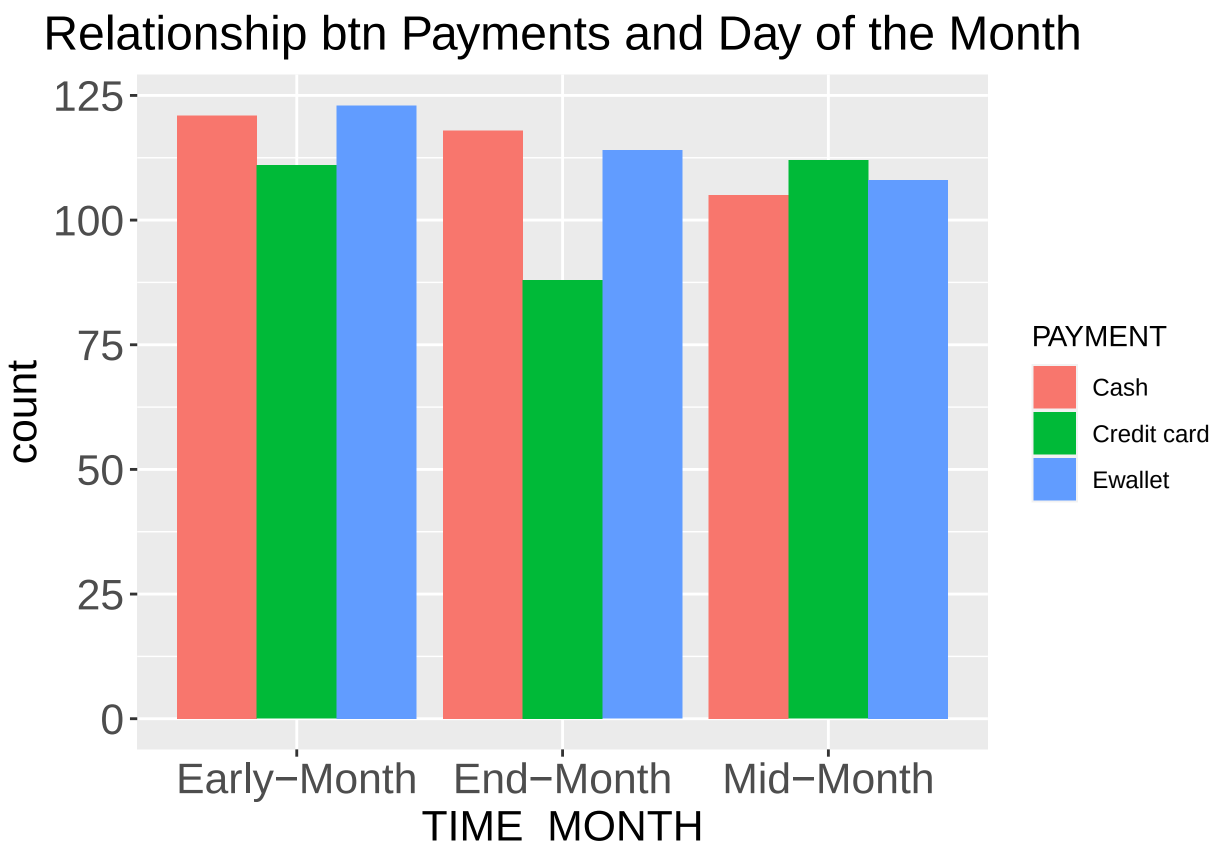

```r
# DIstribution by Months
ggplot(df, aes(x = MONTH)) +
  geom_bar( position= "dodge", fill= 'blue') +

  ggtitle("Distribution by Months") +
   theme(axis.text = element_text(size=14),
          axis.title = element_text(size = 14),
          plot.title = element_text(hjust = 0.5, size = 16))
```



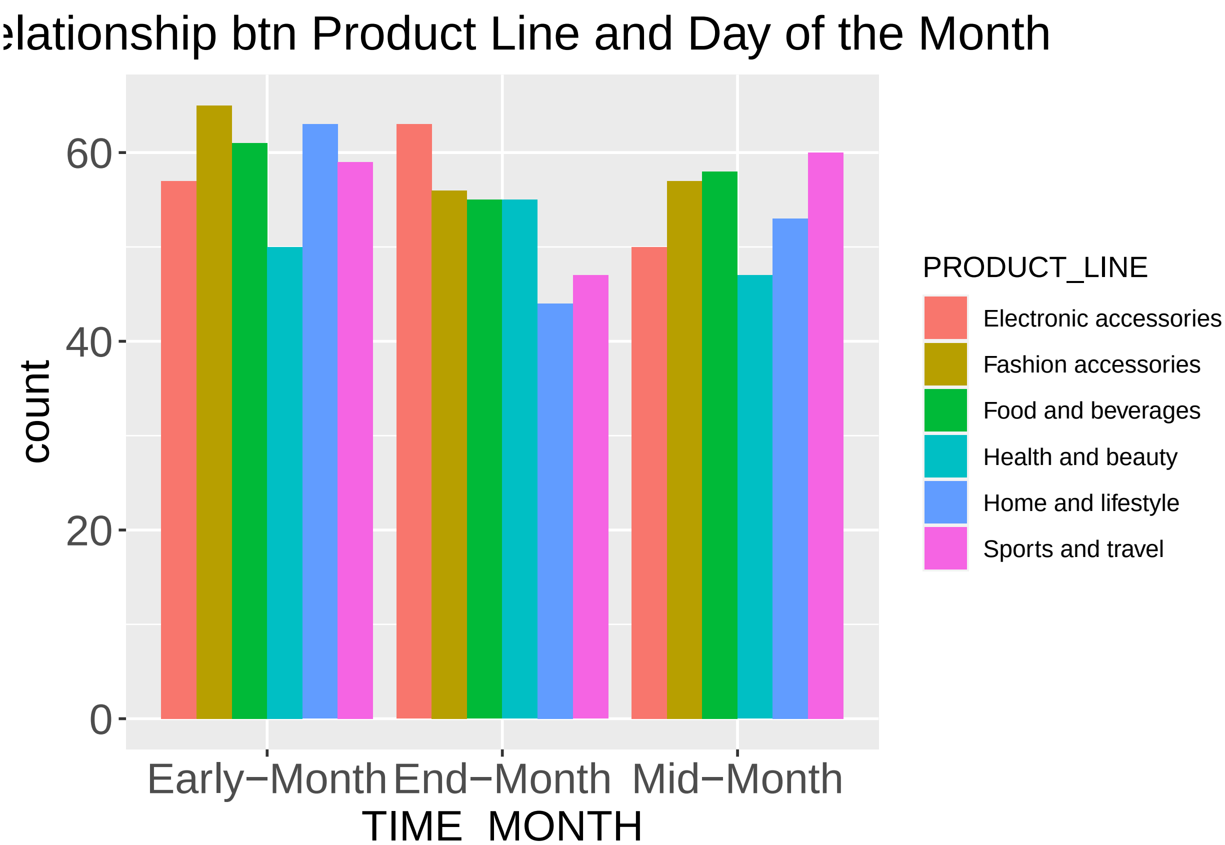

```
#Month One was the most active followed by Month 3
```

BIVARIATE ANALYSIS

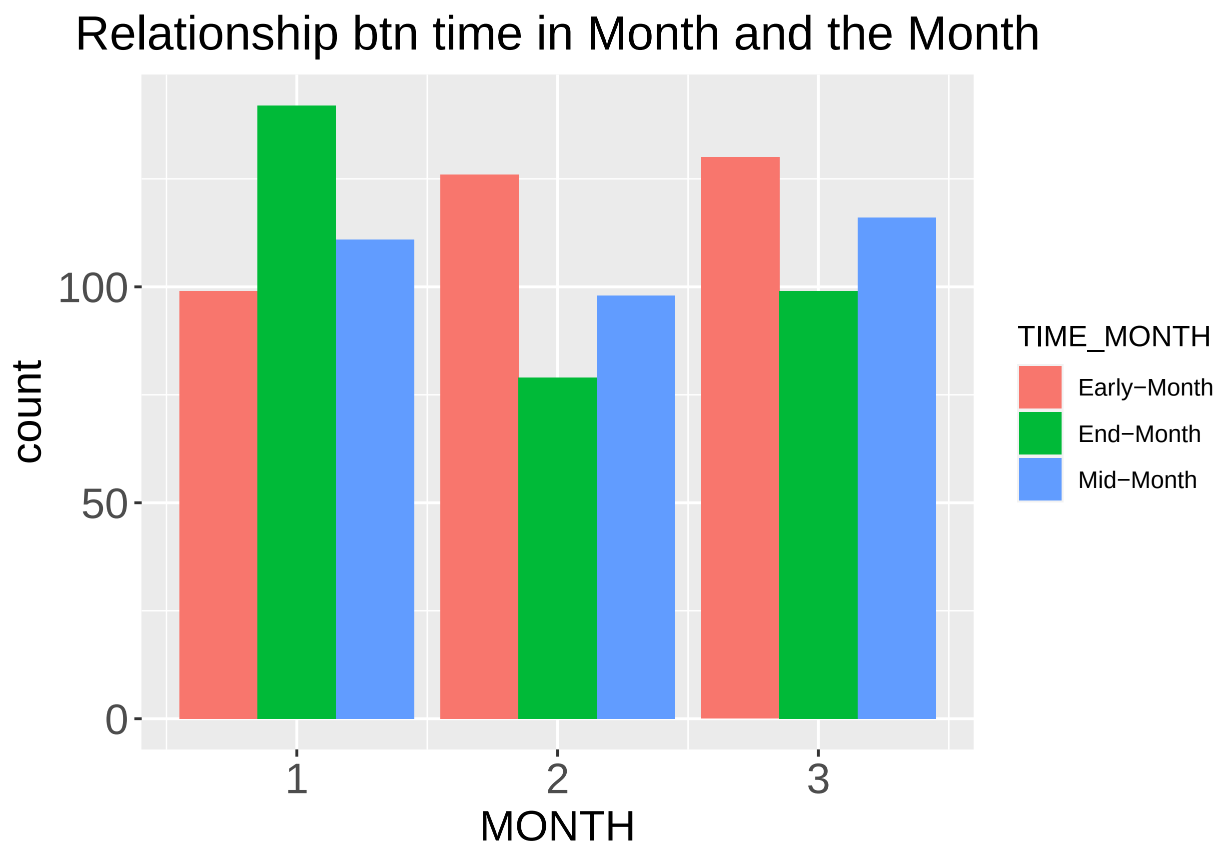
```
#Relationship btn TIme in Month & Payments
options(repr.plot.width = 20, repr.plot.height = 20)
ggplot(df, aes(x = TIME_MONTH, fill = PAYMENT)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship btn Payments and Day of the Month") +
   theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 16),
          plot.title = element_text(hjust = 0.5, size = 18))
```



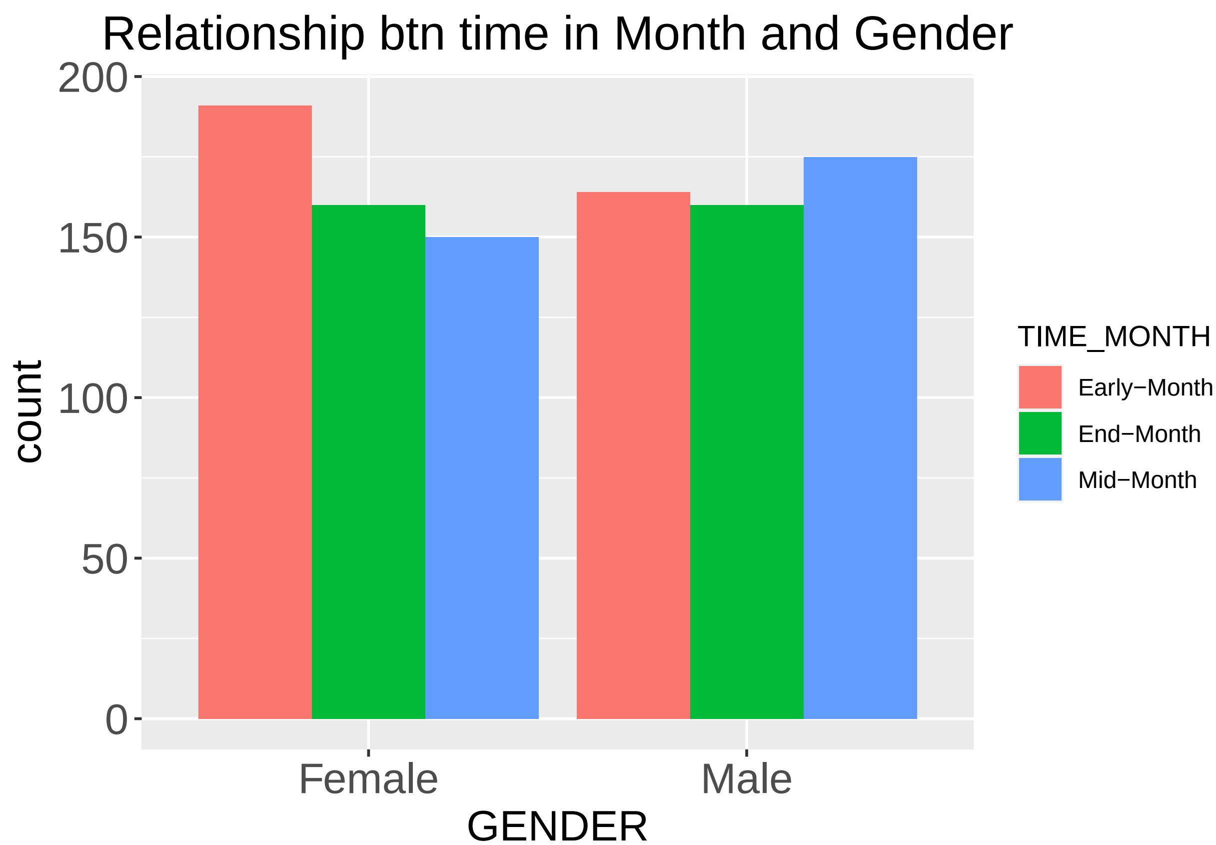

```
#Most of the Payments are made Earlier in the month.
#Across the month, eWallets are used the most earlier in the Month

#Credit Cards are used the least towards the end of the month

#During the Middle of the Month, Credit Cards are the most popular mode of making payments


#When are Different Products bought
options(repr.plot.width = 20, repr.plot.height = 20)
ggplot(df, aes(x = TIME_MONTH, fill = PRODUCT_LINE)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship btn Product Line and Day of the Month") +
   theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 16),
          plot.title = element_text(hjust = 0.5, size = 18))
```



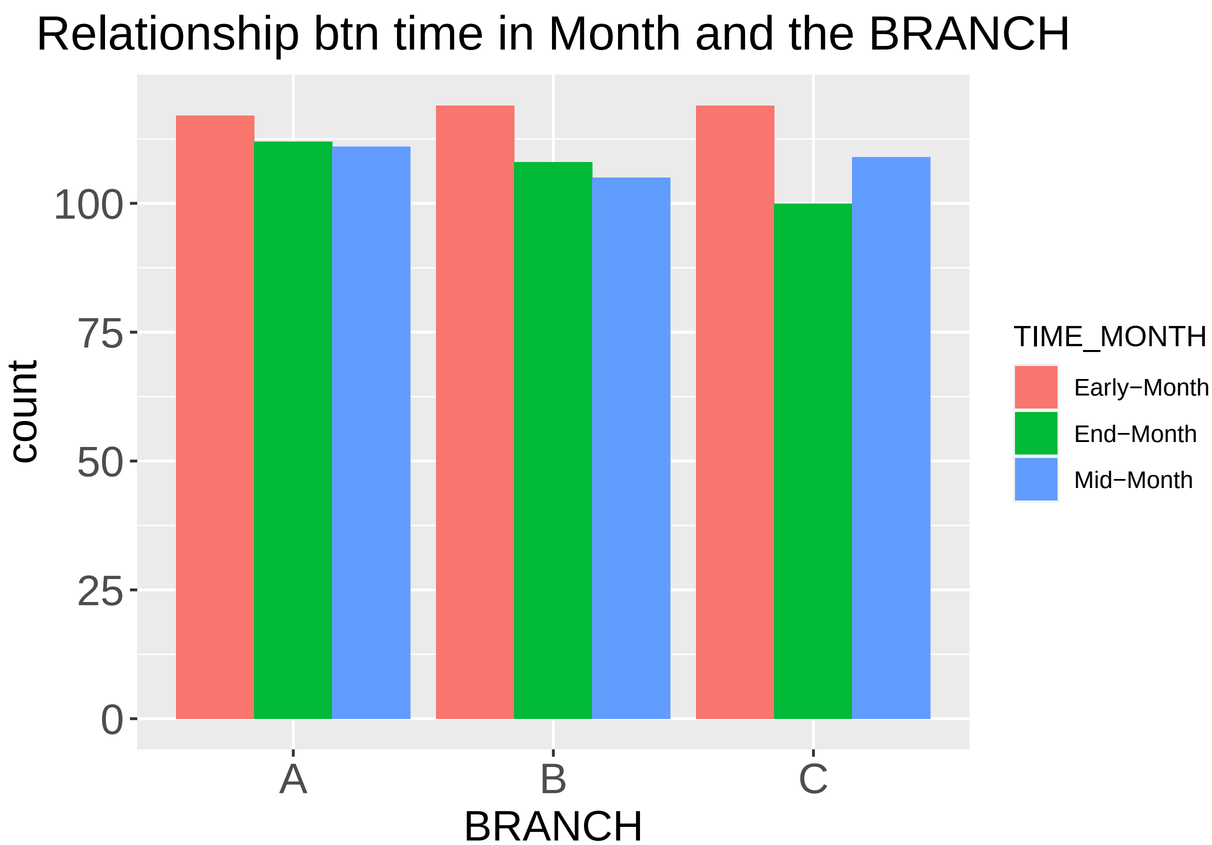

```
#Fashion accessories are bought the most early in the month, followed by Home & Lifestyle Products
# and Food respectively.

#Health and Beauty Products are the least sold products earlier in the month

#It seems most people traveled mid-month as Sports & Travel were the most popular Product Lines.

#Food and Beverages followed closely behind whilst Health and Beauty were the least popular products

#ELectronic Accessories and Fashion accessories were most popular towards the end of the Month whilst
#Home & Lifestyle Products were the least popular



#How the DIfferent seasons in a month compare across the 3 months
options(repr.plot.width = 20, repr.plot.height = 20)
ggplot(df, aes(x = MONTH, fill = TIME_MONTH)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship btn time in Month and the Month") +
   theme(axis.text = element_text(size=16),
         axis.title = element_text(size = 16),
         plot.title = element_text(hjust = 0.5, size = 18))
```



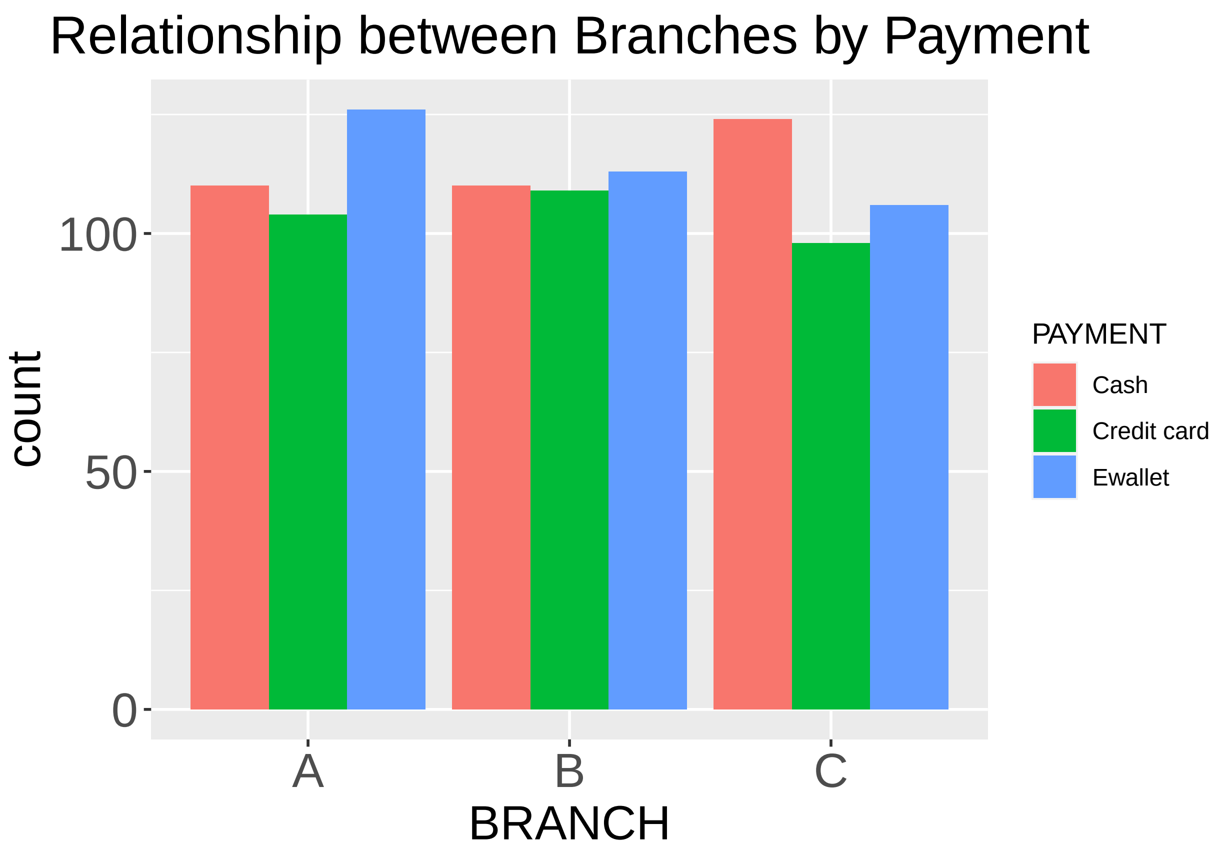

```
#In Month 1, end of the MOnth was the busiest whilst it was the least busy in Months 2 & 3

#Earlier in the month was the busiest in months 2 & 3



options(repr.plot.width = 20, repr.plot.height = 20)
ggplot(df, aes(x = GENDER, fill = TIME_MONTH)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship btn time in Month and Gender") +
   theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 16),
          plot.title = element_text(hjust = 0.5, size = 18))
```



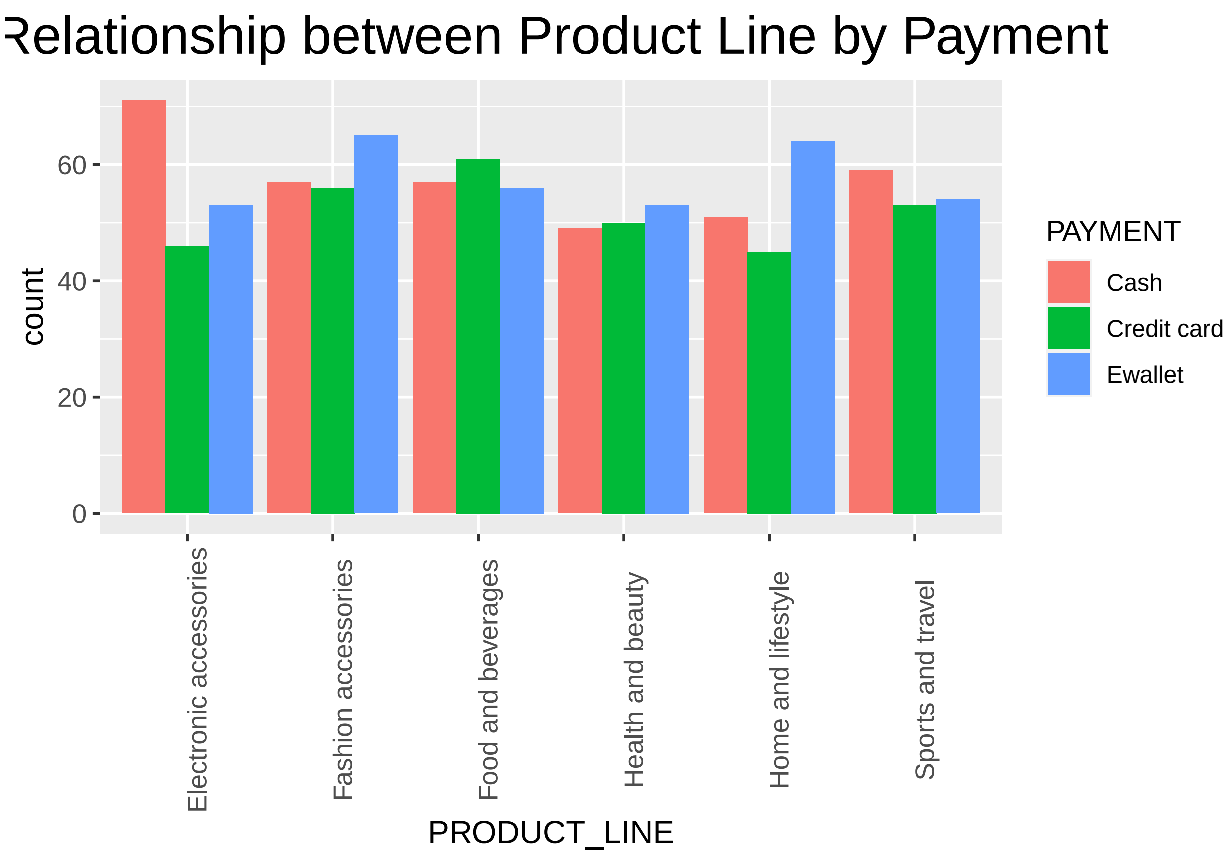

```
#Most of the Women shop earlier in the MOnth
#Although more men shopped in the middle of the month,
#they are more likely to shop at any other time in the Month than earlier in the month compared to wome


#Time of the Month and Branches
options(repr.plot.width = 20, repr.plot.height = 20)
ggplot(df, aes(x = BRANCH, fill = TIME_MONTH)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship btn time in Month and the BRANCH") +
   theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 16),
          plot.title = element_text(hjust = 0.5, size = 18))
```



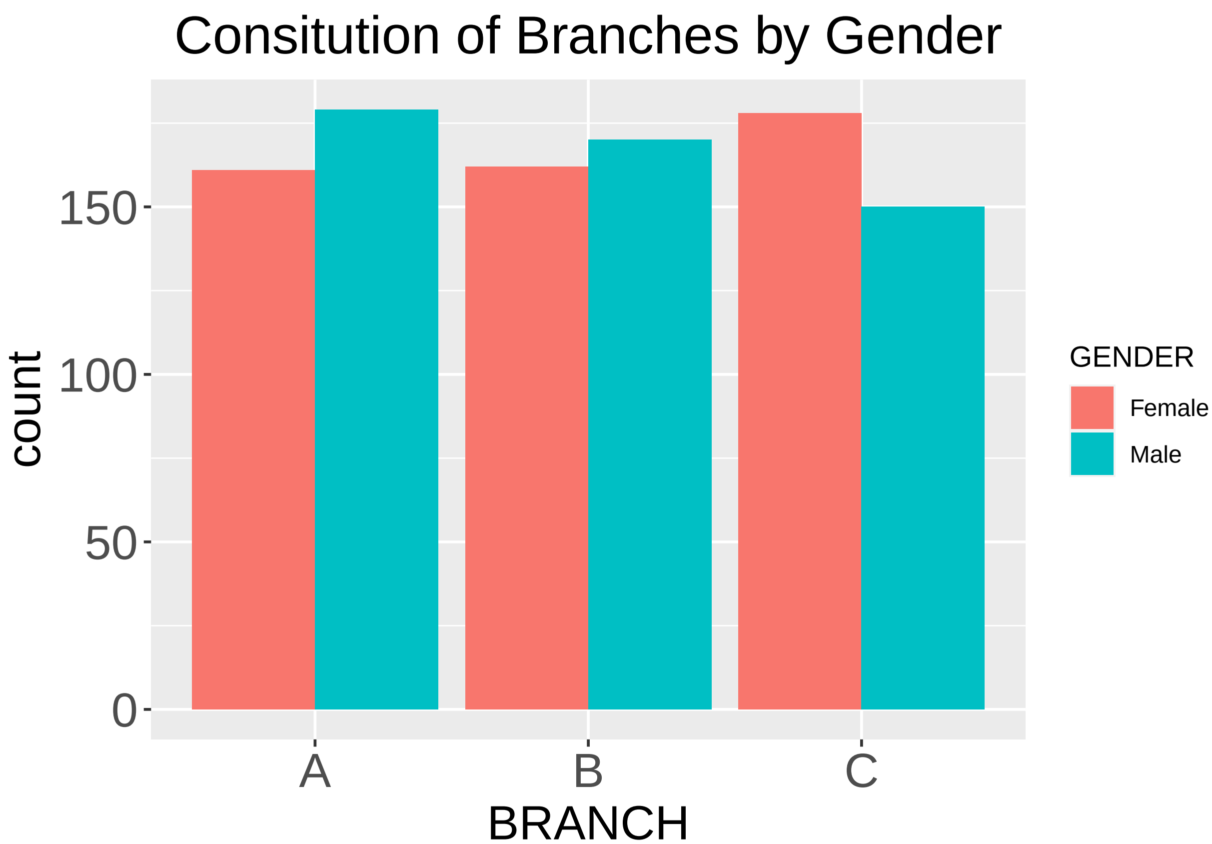

```
#All the Branches are busiest earlier in the Month
#Branches B & C are relatively busier than Branch A

#For Branch C, end month is the less busier period

#FOr branch A, the swings in busier and less busy periods is not as large as it in branch C


# Relationship between Branches & Payment
ggplot(df, aes(x = BRANCH, fill = PAYMENT)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship between Branches by Payment") +
   theme(axis.text = element_text(size=18),
          axis.title = element_text(size = 18),
          plot.title = element_text(hjust = 0.5, size = 20))
```



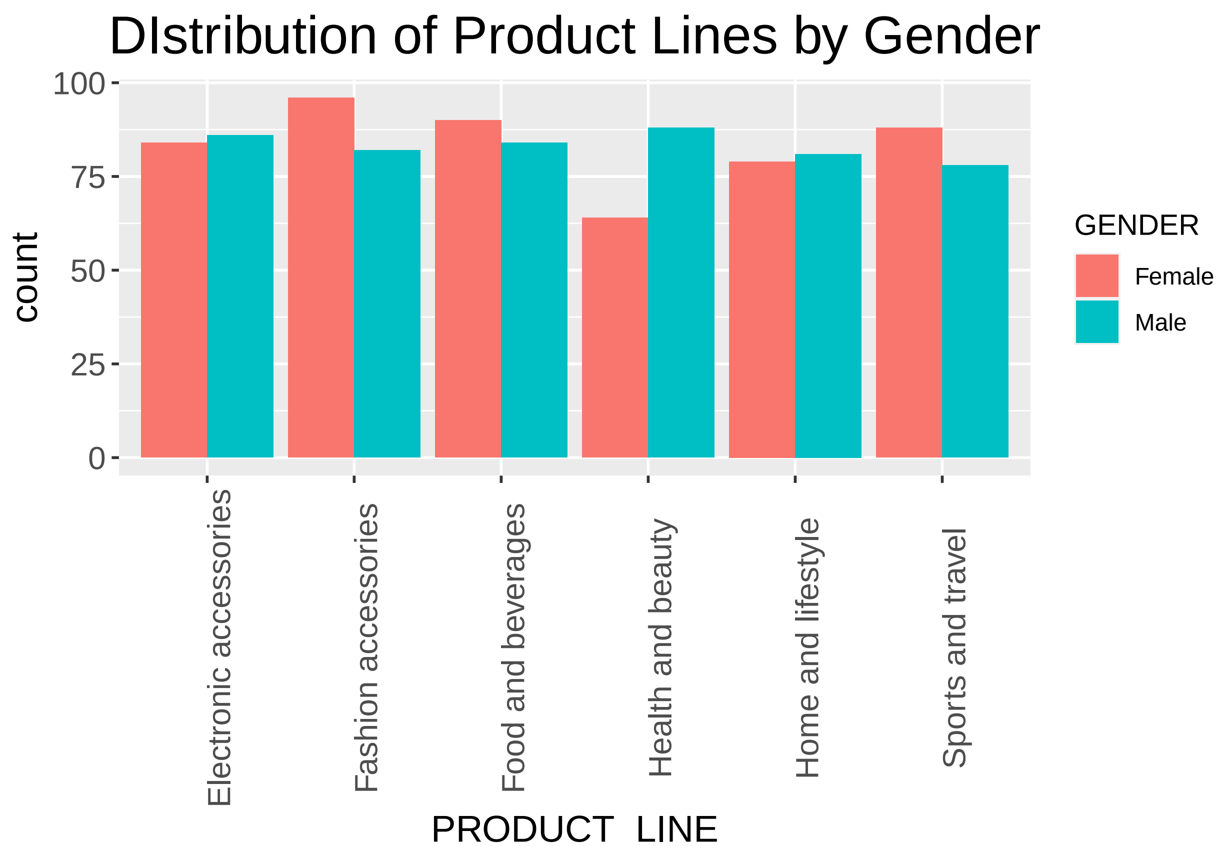

```
#E-Wallets are generally the most popular payment method across the branches

#Credit Cards were the least popular payment method across the branches

# Cash was the most preferred payment method of payment in Branch C

#The most transactions in a Branch were in Branch A and E Wallets were used the most


# Relationship between Product Line & Payment
ggplot(df, aes(x = PRODUCT_LINE, fill = PAYMENT)) +
  geom_bar(position= "dodge") +
   theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Relationship between Product Line by Payment") +
   theme(axis.text = element_text(size=10),
          axis.title = element_text(size = 12),
          plot.title = element_text(hjust = 0.5, size = 20))
```



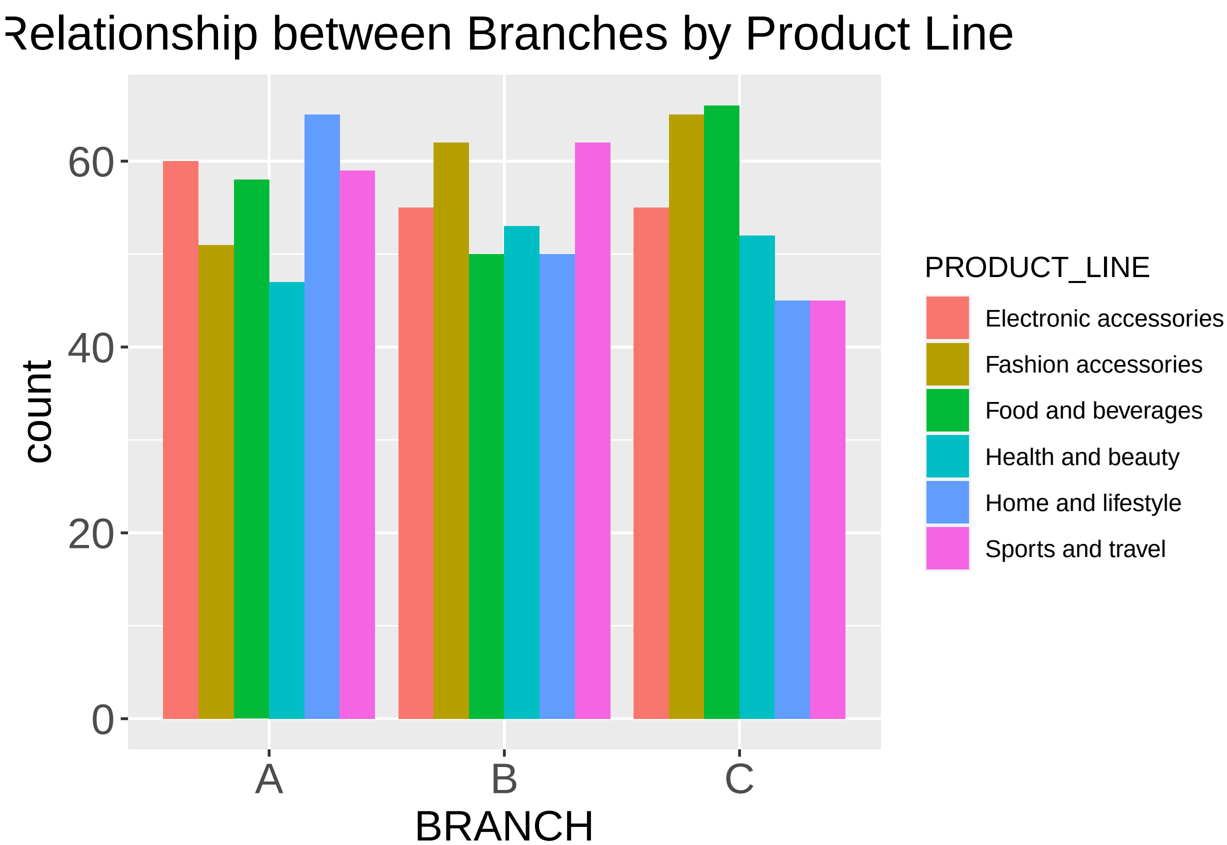

```
#Most of the ELectronic accessories purchases were paid for by Cash
#Fashion Accessories, Health & Beauty and Home & Lifestyle were paid for by eWallets for most of the ti


# Relationship between Branches & Gender
ggplot(df, aes(x = BRANCH, fill = GENDER)) +
  geom_bar(position= "dodge") +
  ggtitle("Consitution of Branches by Gender") +
   theme(axis.text = element_text(size=18),
          axis.title = element_text(size = 18),
          plot.title = element_text(hjust = 0.5, size = 20))
```



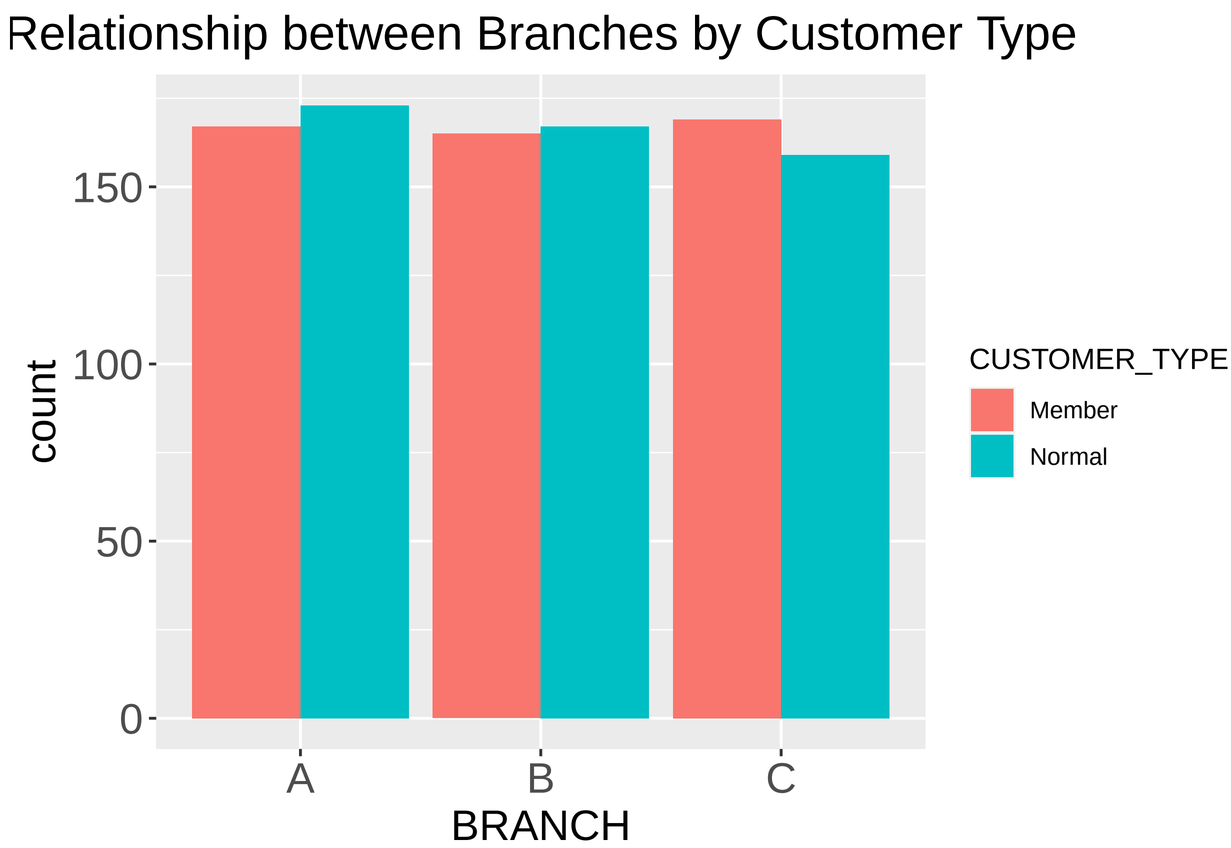

```
#More Males than females visited branches A & B

#More women visited branch C


#Distribution of Product Line by Gender
ggplot(df, aes(x = PRODUCT_LINE, fill = GENDER)) +
  geom_bar(position= "dodge") +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("DIstribution of Product Lines by Gender") +  theme(axis.text = element_text(size=12),
          axis.title = element_text(size = 14),
          plot.title = element_text(hjust = 0.5, size = 20))
```



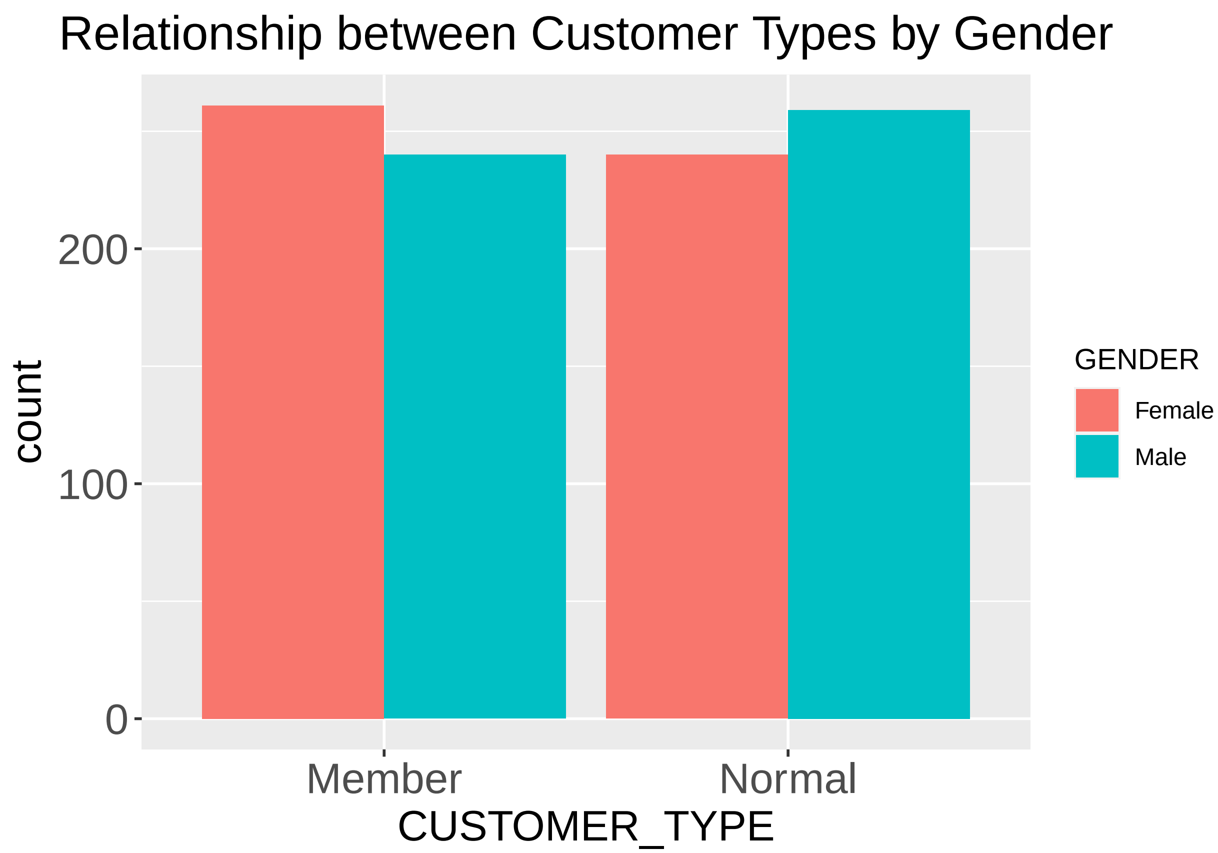

```
#Majority of the men ordered Electronic accessories, Health and beauty and Home & Lifestyle products

#Majority of the women ordered Fashion Accessories, Food & beverages and Sports and travel products

#The most ordered product was Fashion and Accessories

#Food and Beverages were paid for mostly by Credit Cards


# Relationship between Branches & Gender
ggplot(df, aes(x = BRANCH, fill = PRODUCT_LINE)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship between Branches by Product Line") +
   theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 16),
          plot.title = element_text(hjust = 0.5, size = 18))
```



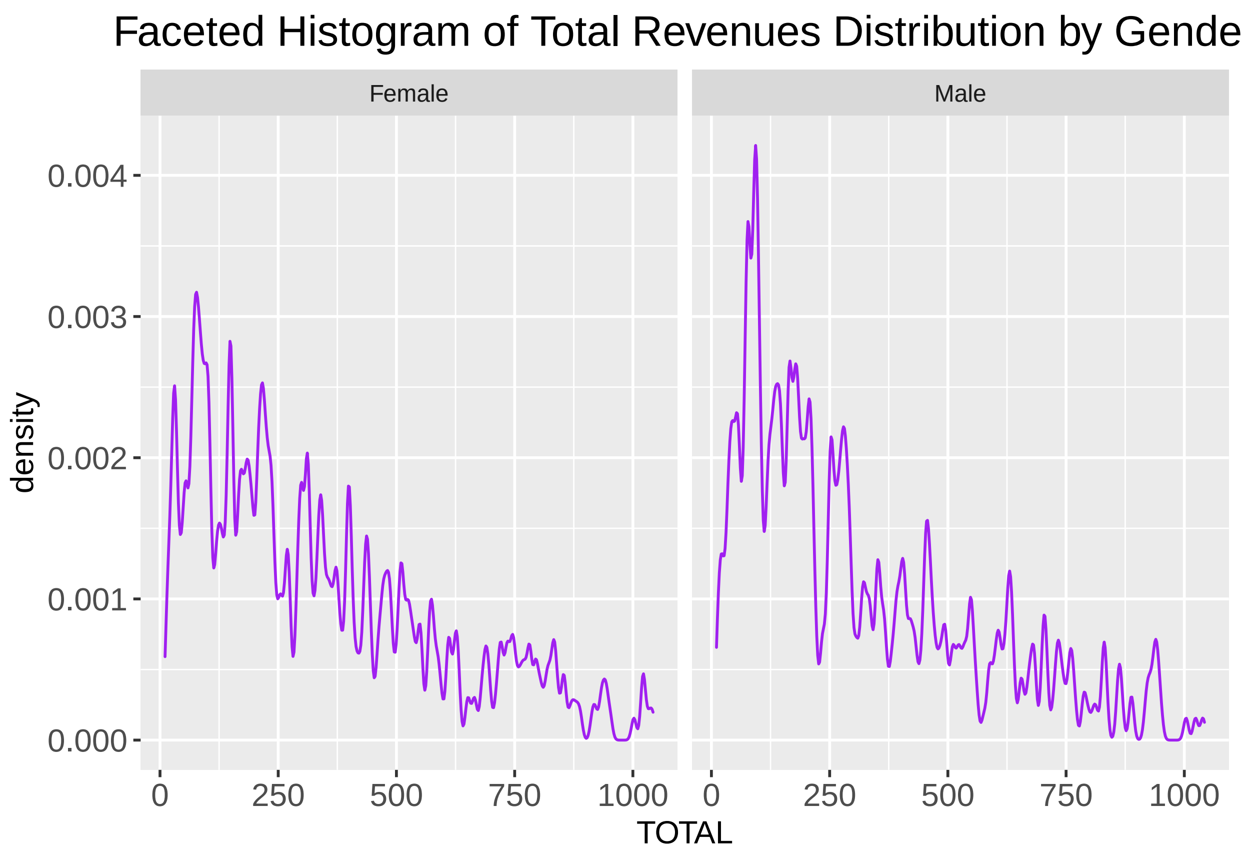

```
#Food & Beverages and Fashion accessories were the most popular products in Branch C whilst
# Home & Lifestyle and Sports & travel were the least popular product lines

#In Branch C, Fashion Accessories and Sports & travel were the most popular product lines whilst
#Food & beverages and Home & Lifestyle were the least popular

#In branch A, Home and Lifestyle products were the most popular. Health and Beauty was the least
#popular product line in branch A


# Relationship between Branches & Customer Type
ggplot(df, aes(x = BRANCH, fill = CUSTOMER_TYPE)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship between Branches by Customer Type") +
   theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 16),
          plot.title = element_text(hjust = 0.5, size = 18))
```



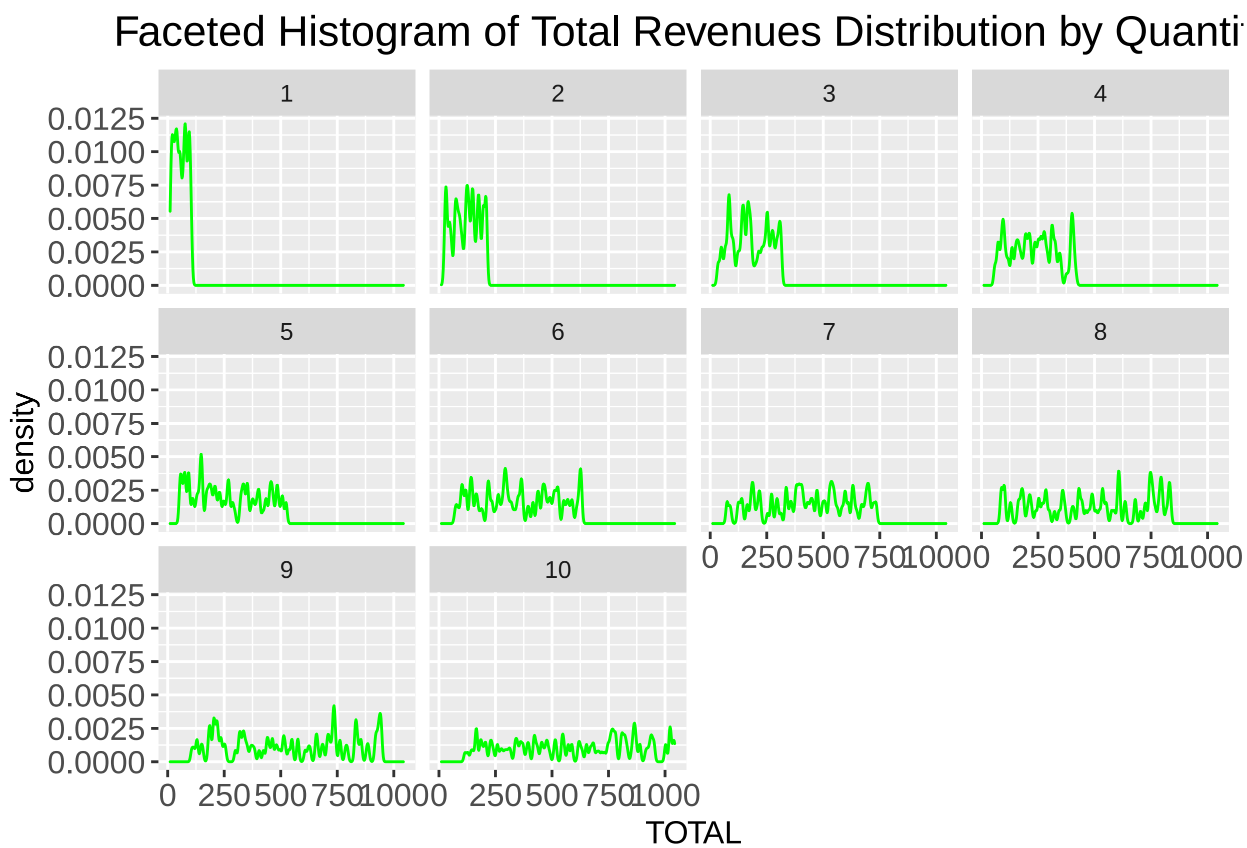

```
#Most of the Customers in branches A & B were non-members
# Branch C had the majority of customers as branch members


# Relationship between Customer Type by Gender
ggplot(df, aes(x = CUSTOMER_TYPE, fill = GENDER)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship between Customer Types by Gender") +
   theme(axis.text = element_text(size=16),
         axis.title = element_text(size = 16),
         plot.title = element_text(hjust = 0.5, size = 18))
```



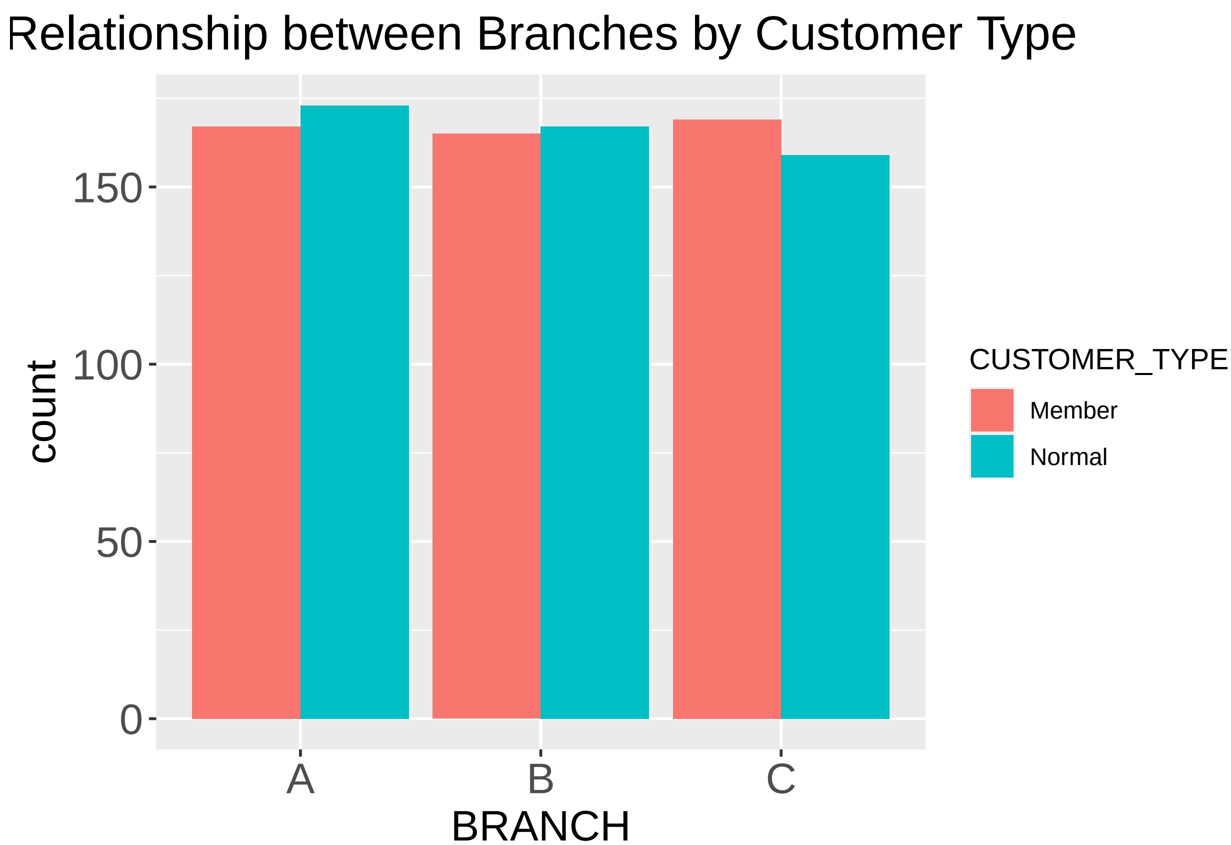

```
#Females were more likely to have Customer membership than the males


#Using Faceted Histograms, we investigate the distribution of Total along
#Gender
ggplot(df, aes(x= TOTAL)) +

  geom_density(bw = 5, color = "purple") +
facet_wrap(~GENDER) +
ggtitle("Faceted Histogram of Total Revenues Distribution by Gender") +  theme(axis.text = element_text
          axis.title = element_text(size = 12),
          plot.title = element_text(hjust = 0.5, size = 16))
```



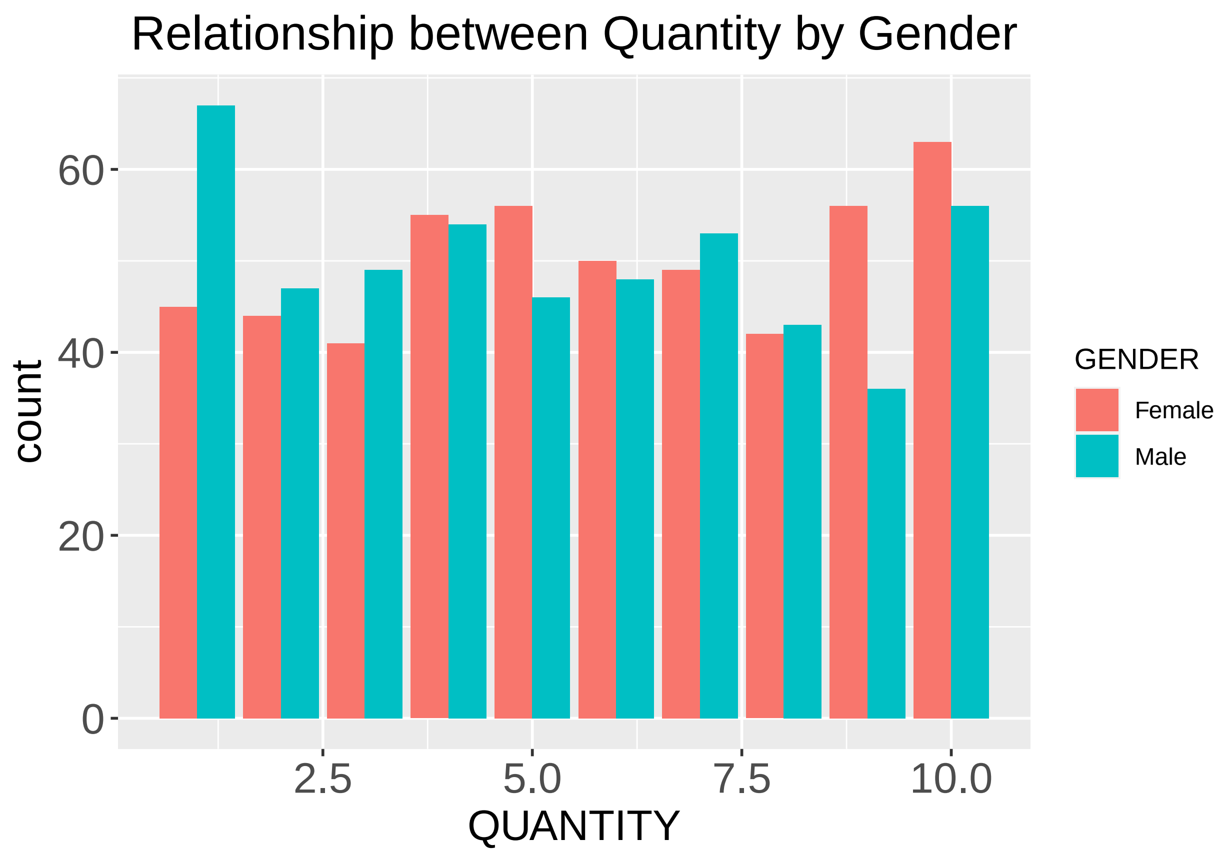

```
#Faceted Plot of Total Revenues by QUANTITY
ggplot(df, aes(x= TOTAL)) +
  geom_density(bw = 5, color = "green") +
facet_wrap(~QUANTITY) +
ggtitle("Faceted Histogram of Total Revenues Distribution by Quantity") +  theme(axis.text = element_te
          axis.title = element_text(size = 12),
          plot.title = element_text(hjust = 0.5, size = 16))
```




```
# Relationship between Branches & Customer Type
ggplot(df, aes(x = BRANCH, fill = CUSTOMER_TYPE)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship between Branches by Customer Type") +
   theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 16),
          plot.title = element_text(hjust = 0.5, size = 18))
```



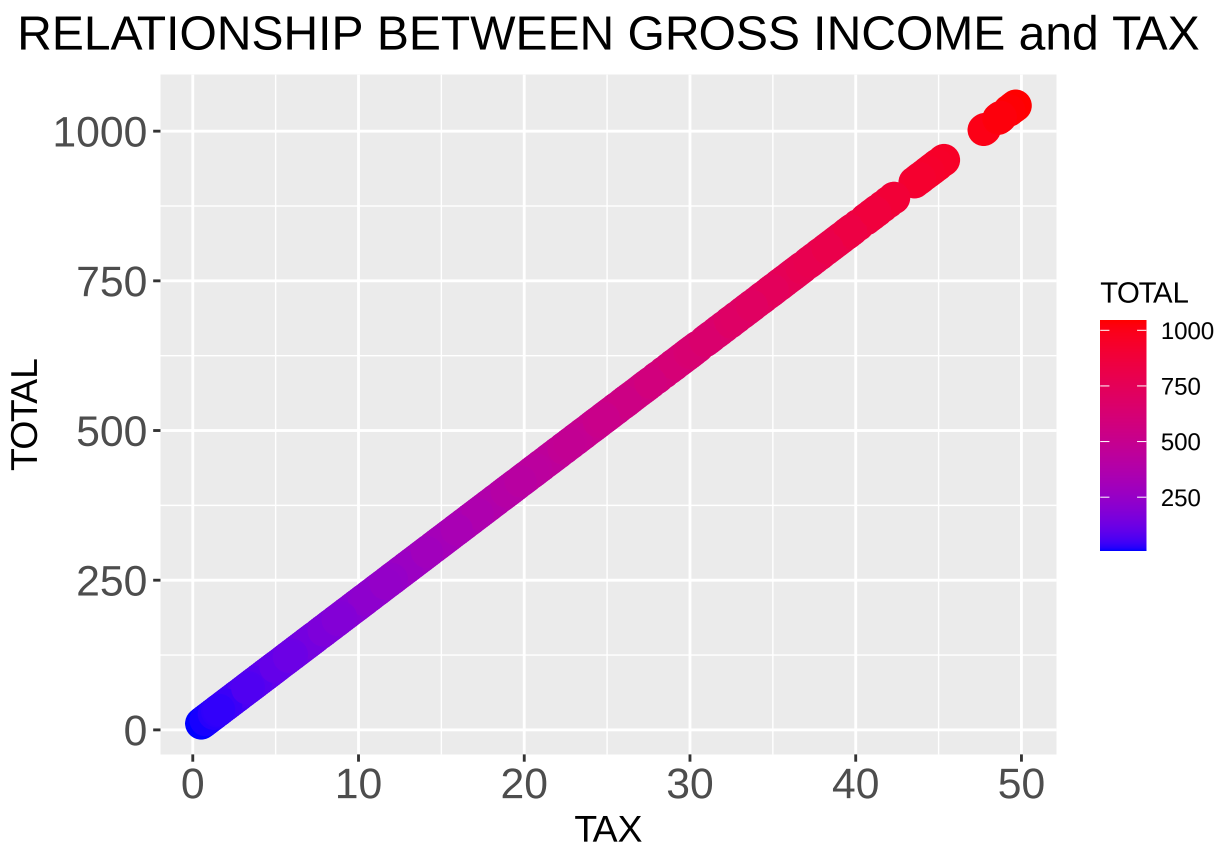

```r
# Relationship between QUANTITY & Customer Type
ggplot(df, aes(x = QUANTITY, fill = GENDER)) +
  geom_bar(position= "dodge") +
  ggtitle("Relationship between Quantity by Gender") +
   theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 16),
          plot.title = element_text(hjust = 0.5, size = 18))
```



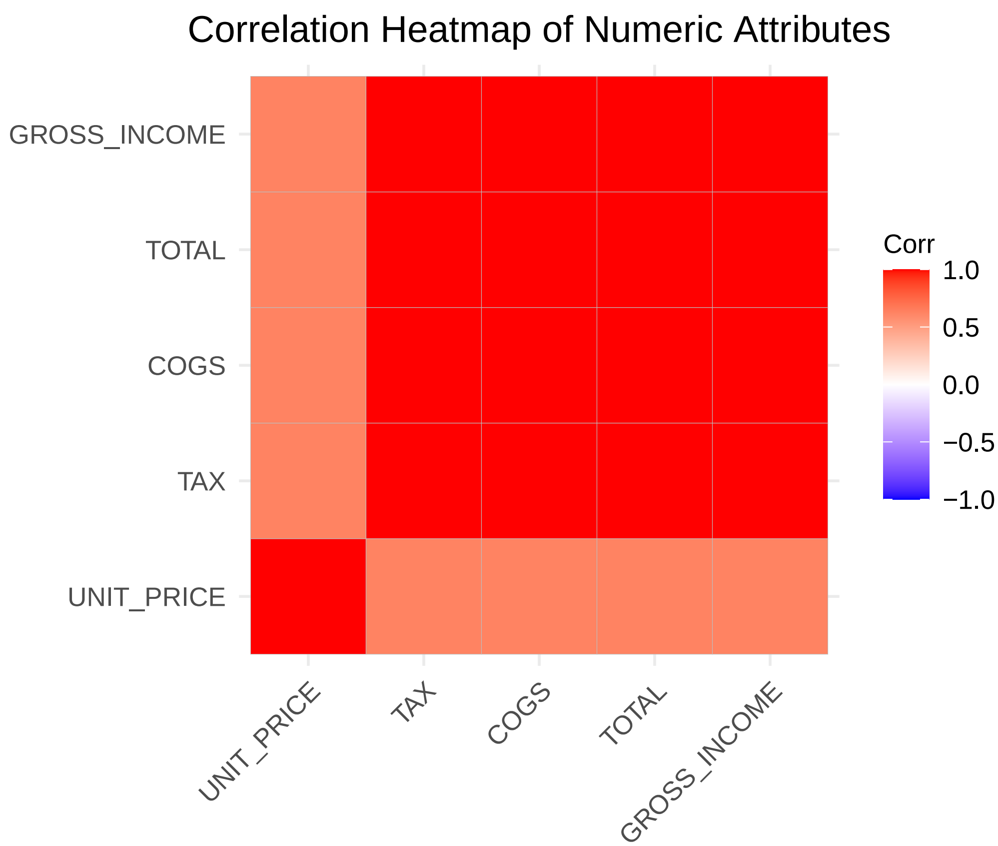

```
#The least Quantities (1) were mostly bought by men whilst the
#highest Quantities were bought by women
```

Multivariate Analysis

```
ggplot(df, aes(x=TAX, y= TOTAL)) +

geom_point(aes(color=TOTAL), size=5) + scale_color_gradient(low='blue', high = 'red') +

ggtitle("RELATIONSHIP BETWEEN GROSS INCOME and TAX") +  theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 14),
          plot.title = element_text(hjust = 0.5, size = 18))
```




```r
#Plotting the Correlation Heatmap of the Numeric Attributes
options(repr.plot.width = 15, repr.plot.height = 10)
ggcorrplot(cor(num.cols), tl.cex =10) +
  labs(title = "Correlation Heatmap of Numeric Attributes") +
  theme(axis.title = element_text(size = 14),
        plot.title = element_text(hjust = 0.5, size = 14),
        legend.title = element_text(size = 10),
        legend.text = element_text(size = 10))
```




```
#The attributes in the dataset have a very high correlation
```

7. IMPLEMENTING THE SOLUTION

a) Feature Selection

In the preceding section, we've seen how highly correlated our attributes are.
Should we proceed to building models for further analysis, our model will be
less robust given the high levels of multicollinearity.
As such we will need to carefully select the most relevant attributes by reducing
the dimensions in the dataset.
To achieve this goal, we will employ several unsupervised learning Feature Selection algorithms

1) Filter Method

Works by applying a metric to assign a score to each attribute in the dataset.
A score-based ranking approach is used to determine the most relevant features
Implementing Technique

```
str(df)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  15 variables:
##  $ BRANCH       : Factor w/ 3 levels "A","B","C": 1 3 1 1 1 3 1 3 1 2 ...
##  $ CUSTOMER_TYPE: Factor w/ 2 levels "Member","Normal": 1 2 2 1 2 2 1 2 1 1 ...
##  $ GENDER       : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 2 1 1 1 1 ...
##  $ PRODUCT_LINE : Factor w/ 6 levels "Electronic accessories",..: 4 1 5 4 6 1 1 5 4 3 ...
##  $ UNIT_PRICE   : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ QUANTITY     : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ TAX          : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ PAYMENT      : Factor w/ 3 levels "Cash","Credit card",..: 3 1 2 3 3 3 3 3 2 2 ...
##  $ COGS         : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ GROSS_INCOME : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ RATING       : int  52 57 35 45 14 2 19 41 33 20 ...
##  $ TOTAL        : num  549 80.2 340.5 489 634.4 ...
##  $ MONTH        : int  1 3 3 1 2 3 2 2 1 2 ...
##  $ DAY          : int  5 8 3 27 8 25 25 24 10 20 ...
##  $ TIME_MONTH   : chr  "Early-Month" "Early-Month" "Early-Month" "End-Month" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
#Dataset has 15 Attributes

#1. Encoding the Categorical Variables using ifelse() function

df$TIME_MONTH <- ifelse(df$TIME_MONTH == "Early-Month", 0,
                        ifelse(df$TIME_MONTH== "End-Month", 1,2))

df$PRODUCT_LINE <- ifelse(df$PRODUCT_LINE == "Electronic accessories", 0,
            ifelse(df$PRODUCT_LINE == "Fashion accessories", 1,
            ifelse(df$PRODUCT_LINE == "Food and beverages", 2,
            ifelse(df$PRODUCT_LINE == "Health and beauty", 3,
            ifelse(df$PRODUCT_LINE == "Home and lifestyle", 4, 5)))))


df$CUSTOMER_TYPE <-  ifelse(df$CUSTOMER_TYPE == "Normal", 0, 1)
df$GENDER <- ifelse(df$GENDER == "Male", 0, 1)
df$PAYMENT <- ifelse(df$PAYMENT == "Cash", 0,
                  ifelse(df$PAYMENT == "Credit card", 1,2))

df$BRANCH <- ifelse(df$BRANCH == "A", 0,
ifelse(df$BRANCH == "B", 1, 2))


#2. Defining the Regressors by dropping the TOTAL Column

#We also drop the DAY column as it represents same data as TIME of Month


df.ind <- select(df, -c(TOTAL, DAY))


#Confirming TOTAL has been dropped from df.ind
head(df.ind)
```

```
##    BRANCH CUSTOMER_TYPE GENDER PRODUCT_LINE UNIT_PRICE QUANTITY     TAX PAYMENT
## 1:      0             1      1            3      74.69        7 26.1415       2
```

```
## 2:      2            0      1            0      15.28        5  3.8200       0
## 3:      0            0      0            4      46.33        7 16.2155       1
## 4:      0            1      0            3      58.22        8 23.2880       2
## 5:      0            0      0            5      86.31        7 30.2085       2
## 6:      2            0      0            0      85.39        7 29.8865       2
##      COGS GROSS_INCOME RATING MONTH TIME_MONTH
## 1: 522.83      26.1415     52     1          0
## 2:  76.40       3.8200     57     3          0
## 3: 324.31      16.2155     35     3          0
## 4: 465.76      23.2880     45     1          1
## 5: 604.17      30.2085     14     2          0
## 6: 597.73      29.8865      2     3          1
```

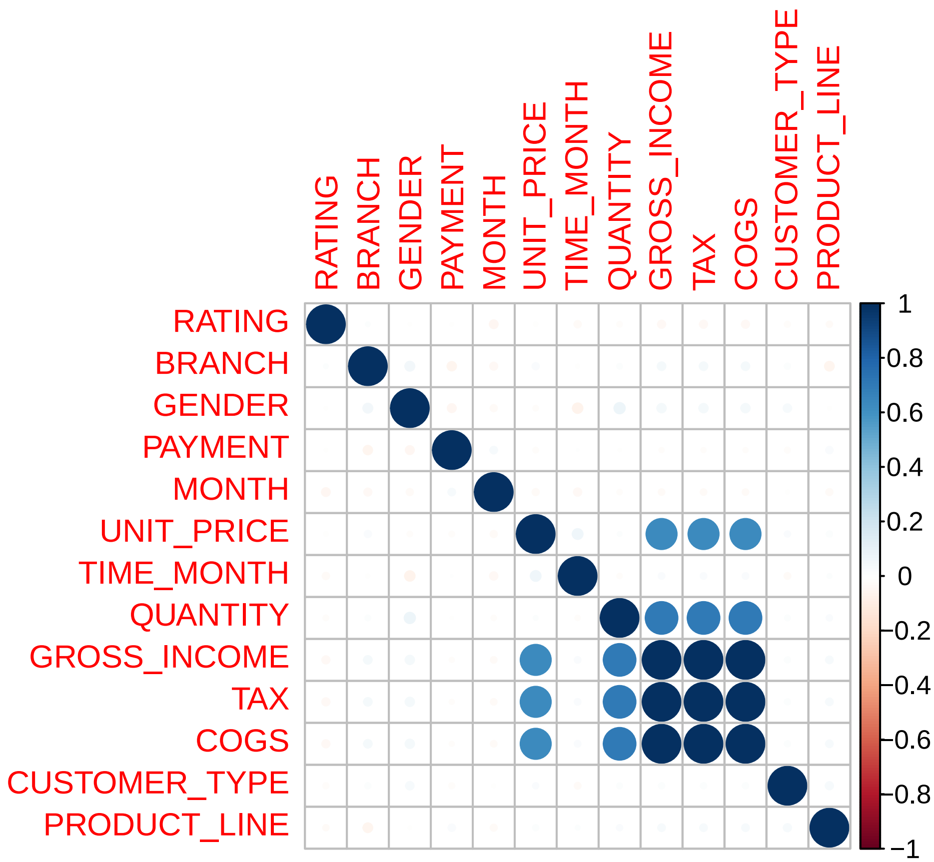
```
#3. Inspecting for Correlation in the attributes

correlationMatrix<- cor(df.ind)

# find variables that are highly correlated
max_corr <- findCorrelation(correlationMatrix, cutoff=0.75)

#Indices of the highly correlated variables
max_corr
```

```
## [1] 7 9
```

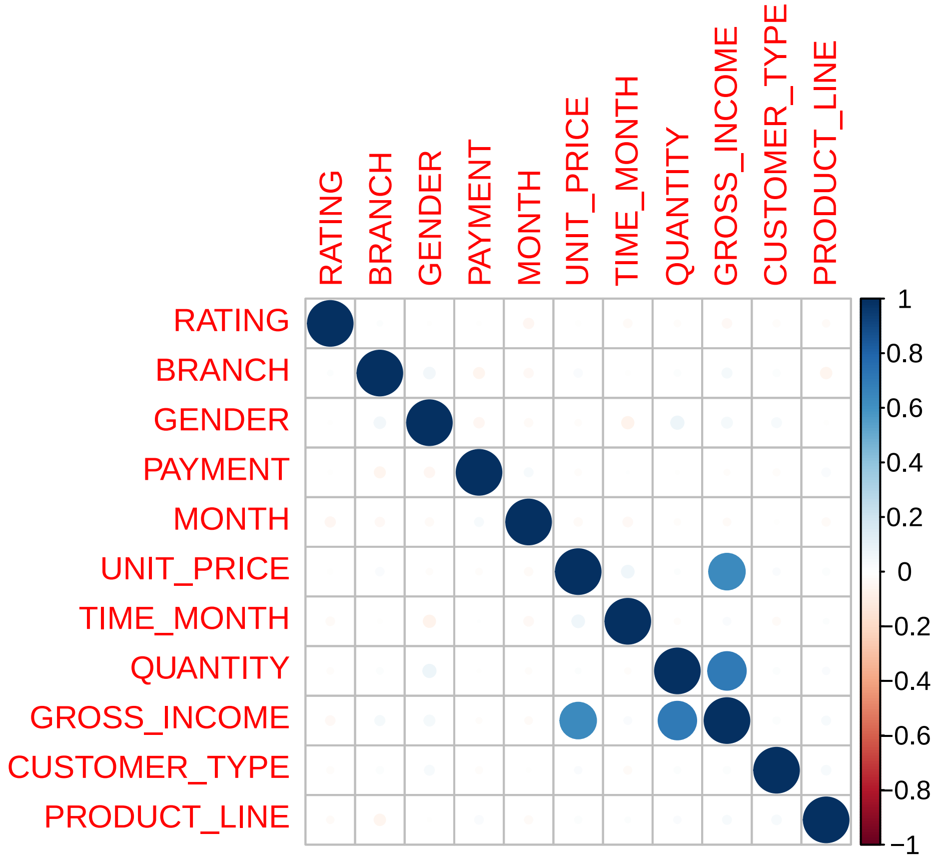
```
#TAX and COGS are highly correlated. We will drop them

df_new <- subset(df.ind, select =-c(max_corr))


str(df_new)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  11 variables:
##  $ BRANCH       : num  0 2 0 0 0 2 0 2 0 1 ...
##  $ CUSTOMER_TYPE: num  1 0 0 1 0 0 1 0 1 1 ...
##  $ GENDER       : num  1 1 0 0 0 0 1 1 1 1 ...
##  $ PRODUCT_LINE : num  3 0 4 3 5 0 0 4 3 2 ...
##  $ UNIT_PRICE   : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ QUANTITY     : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ PAYMENT      : num  2 0 1 2 2 2 2 2 1 1 ...
##  $ GROSS_INCOME : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ RATING       : int  52 57 35 45 14 2 19 41 33 20 ...
##  $ MONTH        : int  1 3 3 1 2 3 2 2 1 2 ...
##  $ TIME_MONTH   : num  0 0 0 1 0 1 1 1 0 2 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
#4. Graphical Comparison
#Comparing the correlation Matrices before and after feature selection
options(repr.plot.height = 15, repr.plot.width = 12)

#par(mfrow = c(1, 2))
corrplot(correlationMatrix, order = "hclust")
```




```
corrplot(cor(df_new), order = "hclust")
```




```
names(df_new)
```

```
## [1] "BRANCH"        "CUSTOMER_TYPE" "GENDER"        "PRODUCT_LINE"
## [5] "UNIT_PRICE"    "QUANTITY"      "PAYMENT"       "GROSS_INCOME"
## [9] "RATING"        "MONTH"         "TIME_MONTH"
```

```
#Branch, Customer Type, Gender, Product Line, Quantity, Payment, Gross Income
#Rating, MOnth & Time of Month are the best variables to include in the model development phase







#RANKING of Features
#Dropping Columns that need to be dropped as established above

#str(df)

#df.rank <- select(df, -DAY)

#Selecting Features using the FSelector Library
#df.rank <- linear.correlation(TOTAL~., df.rank )
```

Wrapper Method

This technique leverages the Clustvarsel package for its implementation.

It will implement a variable selection methodology for model-based clustering to

derive the optimal susbet of variables in a dataset

Implementation

```
#Implementing the Sequential Greedy Search
out <- clustvarsel(df.ind, G=1:9)

out
```

```
## ------------------------------------------------------
## Variable selection for Gaussian model-based clustering
## Stepwise (forward/backward) greedy search
## ------------------------------------------------------
##
##  Variable proposed Type of step   BICclust Model G    BICdiff Decision
##           QUANTITY          Add  -4308.761     E 9   687.4466 Accepted
##              MONTH          Add  -5152.765   VEV 7  1646.6487 Accepted
##                TAX          Add -13700.307   VVE 6 -1457.9036 Rejected
##              MONTH       Remove  -4192.156     E 9  1530.0439 Rejected
##
## Selected subset: QUANTITY, MONTH
```

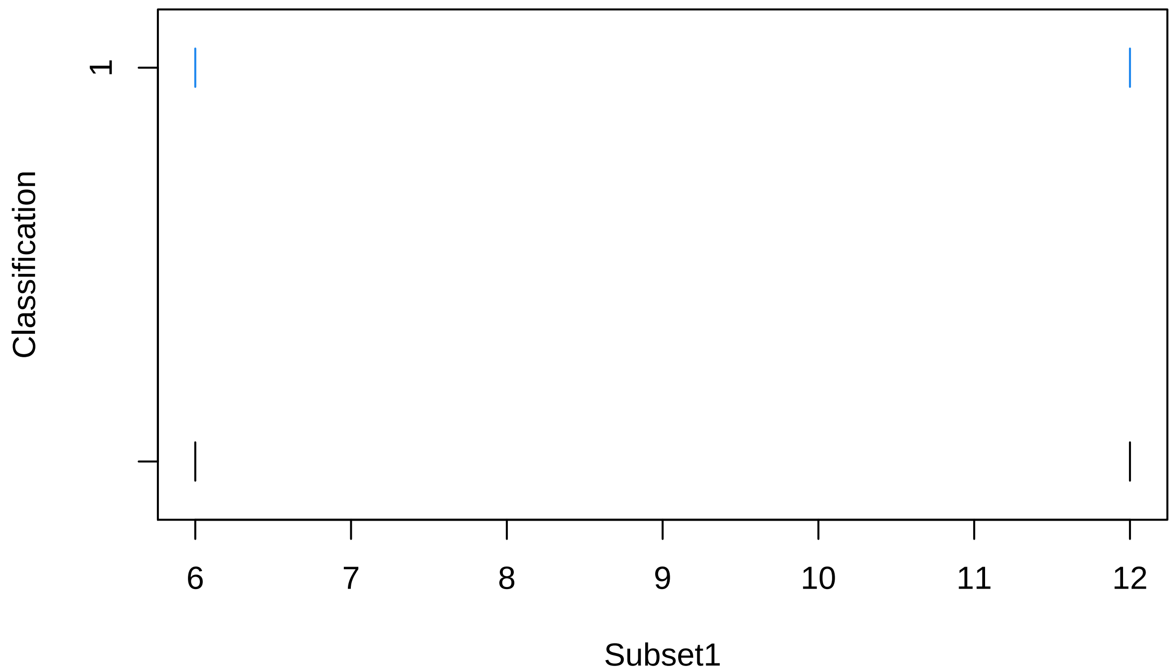
The selection algorithm indicates that the subset we use for the clustering model is composed of Quantity and Month and that other variables should be rejected.

Having identified the variables that we use, we proceed to build the clustering model:

```
Subset1 = df[,out$subset]
mod = Mclust(Subset1, G = 1:9)
summary(mod)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust X (univariate normal) model with 1 component:
##
##  log-likelihood n df      BIC      ICL
##       -5.035102 2  2 -11.4565 -11.4565
##
## Clustering table:
## 1
## 2
```

```
options(repr.plot.width = 15, repr.plot.height = 15)
plot(mod,c("classification"))
```



PRINCIPAL COMPONENT ANALYSIS (PCA)

PCA is one of the most popular dimensionality reduction techniques.

The goal of PCA is to identify and detect correlation between variables. Presence of

strong correlation between variables means the dimension of the dataset could be reduced

It works by first determining the direction of maximum variance in high dimensionality data

which is then projected to a small dimensional subspace while retaining most of the data

```
str(df)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  15 variables:
##  $ BRANCH       : num  0 2 0 0 0 2 0 2 0 1 ...
##  $ CUSTOMER_TYPE: num  1 0 0 1 0 0 1 0 1 1 ...
##  $ GENDER       : num  1 1 0 0 0 0 1 1 1 1 ...
##  $ PRODUCT_LINE : num  3 0 4 3 5 0 0 4 3 2 ...
##  $ UNIT_PRICE   : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ QUANTITY     : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ TAX          : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ PAYMENT      : num  2 0 1 2 2 2 2 2 1 1 ...
##  $ COGS         : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ GROSS_INCOME : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ RATING       : int  52 57 35 45 14 2 19 41 33 20 ...
##  $ TOTAL        : num  549 80.2 340.5 489 634.4 ...
##  $ MONTH        : int  1 3 3 1 2 3 2 2 1 2 ...
```

```
##  $ DAY          : int  5 8 3 27 8 25 25 24 10 20 ...
##  $ TIME_MONTH   : num  0 0 0 1 0 1 1 1 0 2 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

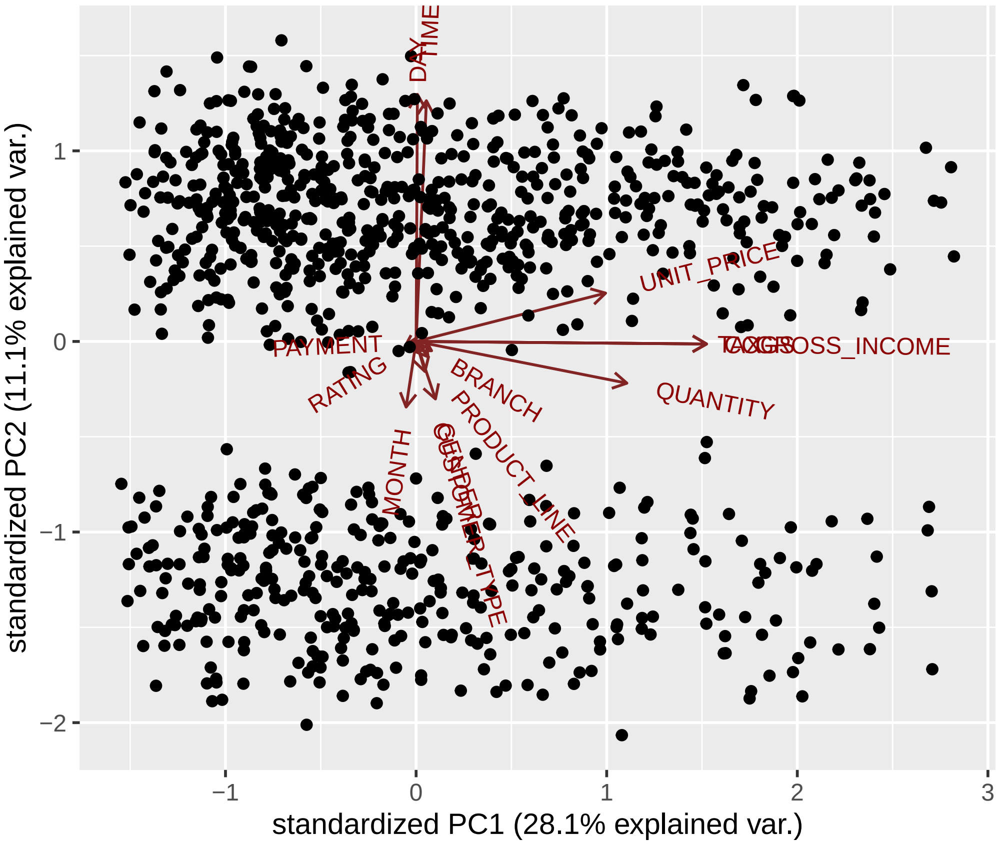
```
#Dropping the TOTAL column
dr.pca <- select(df, -TOTAL)

#1. Implementing Dimensionality Reduction with PCA entails the use of prcomp()

#We need to Scale the data to ensure the correct distances are between datapoints

df.pca <- prcomp(dr.pca, center = TRUE, scale = TRUE)

summary(df.pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.9841 1.2462 1.07035 1.02821 1.01447 1.00366 0.97726
## Proportion of Variance 0.2812 0.1109 0.08183 0.07552 0.07351 0.07195 0.06822
## Cumulative Proportion  0.2812 0.3921 0.47394 0.54946 0.62297 0.69492 0.76314
##                            PC8     PC9    PC10    PC11    PC12      PC13
## Standard deviation     0.96976 0.94925 0.94769 0.70152 0.29034 2.156e-16
## Proportion of Variance 0.06717 0.06436 0.06415 0.03515 0.00602 0.000e+00
## Cumulative Proportion  0.83031 0.89468 0.95883 0.99398 1.00000 1.000e+00
##                            PC14
## Standard deviation     1.08e-16
## Proportion of Variance 0.00e+00
## Cumulative Proportion  1.00e+00
```

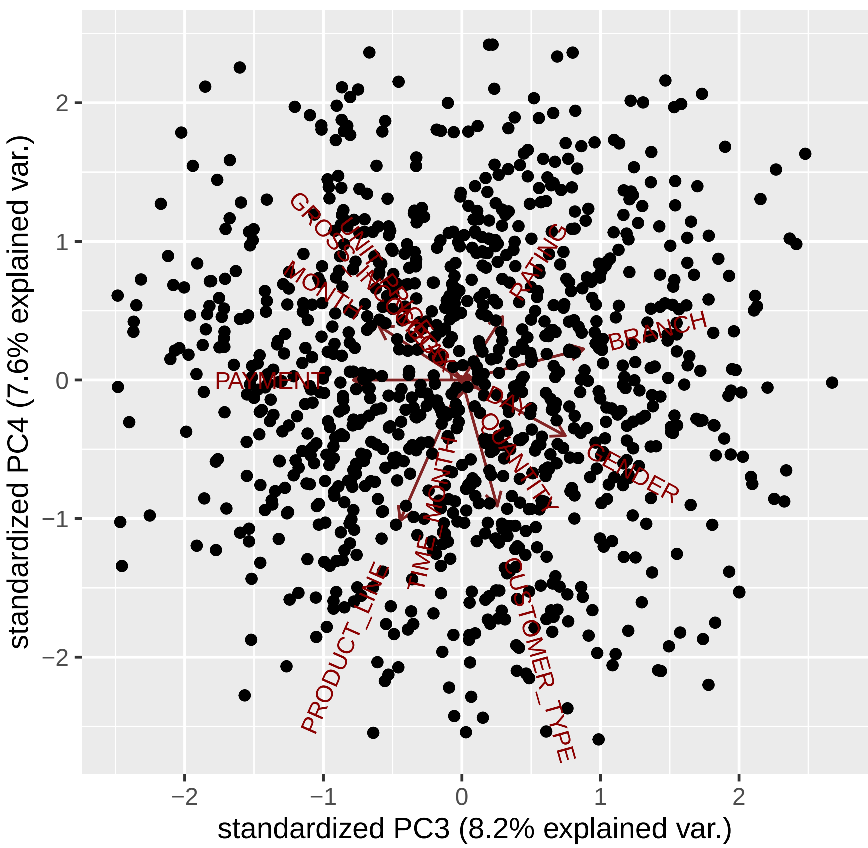
```
#From the summary of components, PC1 explains about 28% of the data
#The first 4 components explain about 55% of the data
```

Visualizing the Results

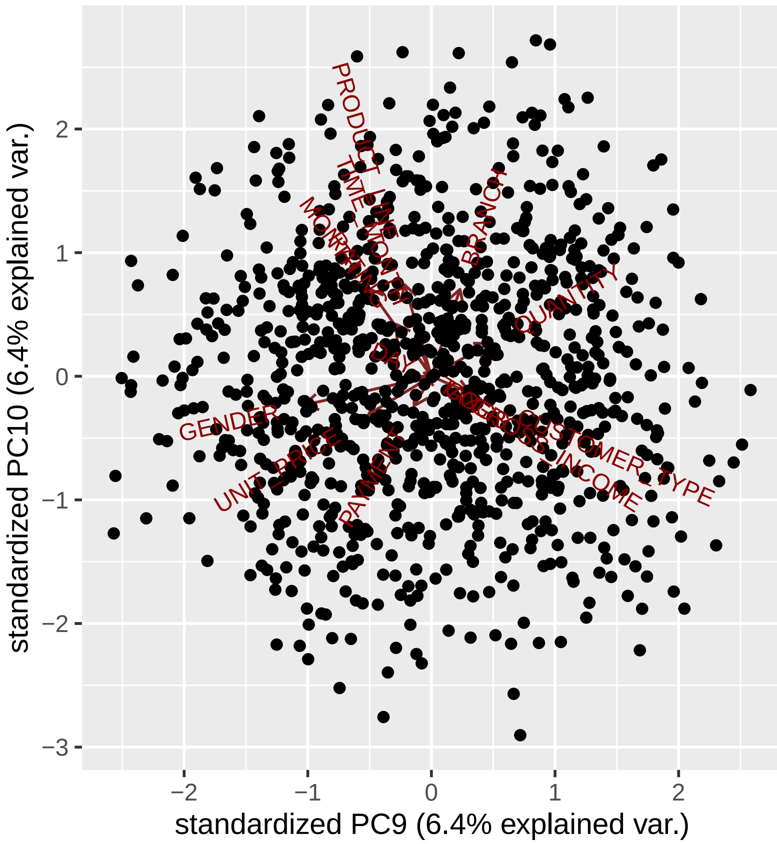
```
#Visualizing the First Principal Component
options(repr.plot.height = 20, repr.plot.width = 18)

ggbiplot(df.pca)
```



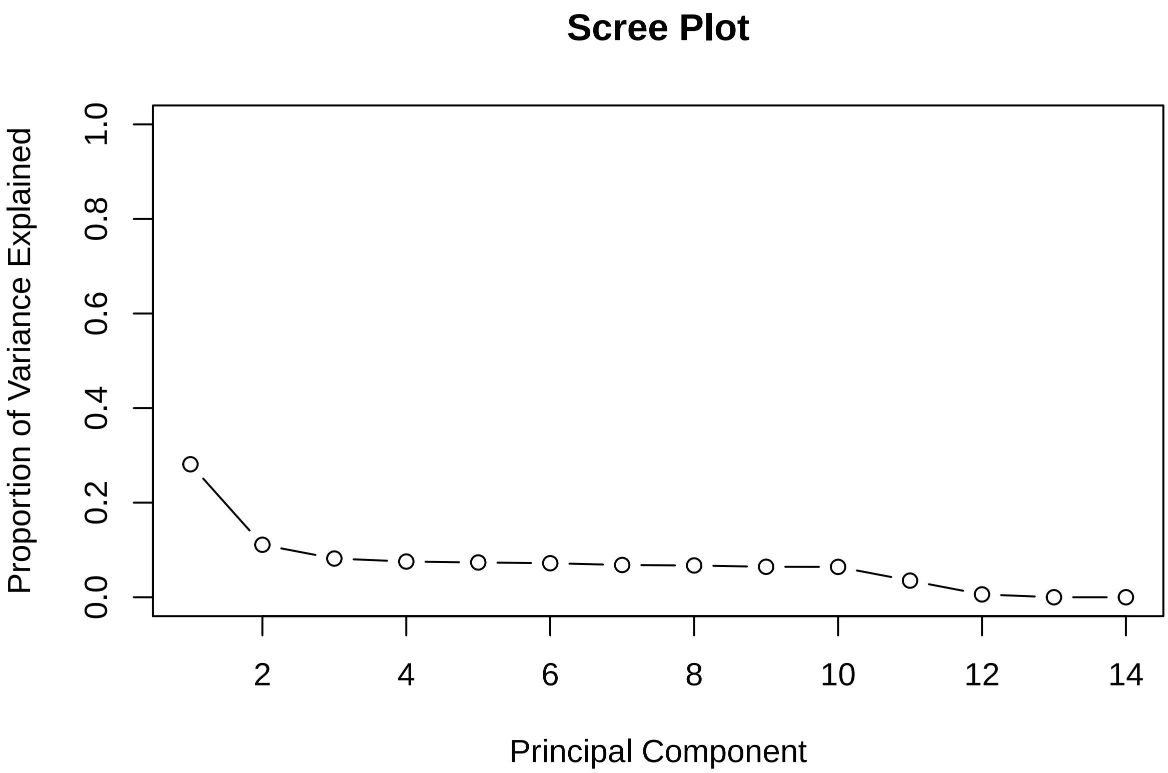

```
#Visualizing Other PCs (3)
options(repr.plot.height = 20, repr.plot.width = 18)

ggbiplot(df.pca, choices = c(3, 4))
```



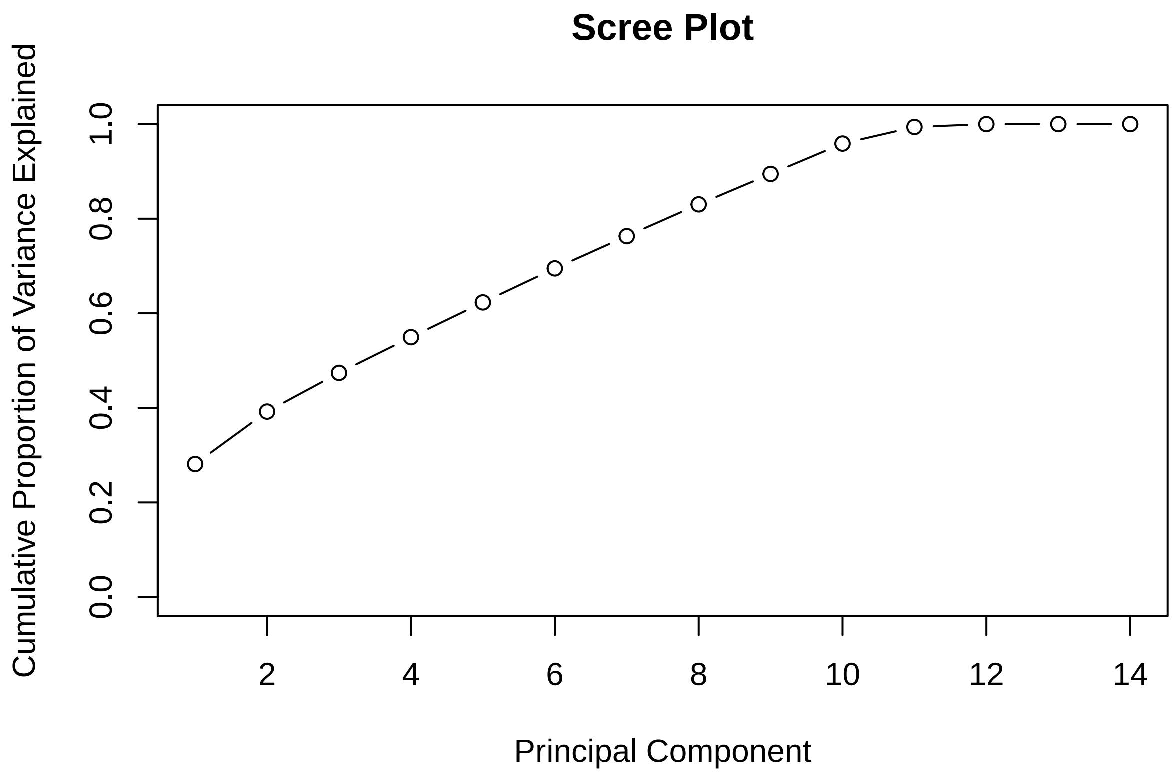

```r
#Visualizing Pc 9
options(repr.plot.height = 20, repr.plot.width = 18)

ggbiplot(df.pca, choices = c(9, 10))
```




```r
#Getting proportion of variance for a scree plot
scree.var <- df.pca$sdev^2
p.var <- scree.var/ sum(scree.var)

#Variance Explained for each Principal Component


plot(p.var, xlab="Principal Component", ylab = "Proportion of Variance Explained",main = "Scree Plot", 
```




```
plot(cumsum(p.var), xlab="Principal Component", ylab = "Cumulative Proportion of Variance Explained",ma
```

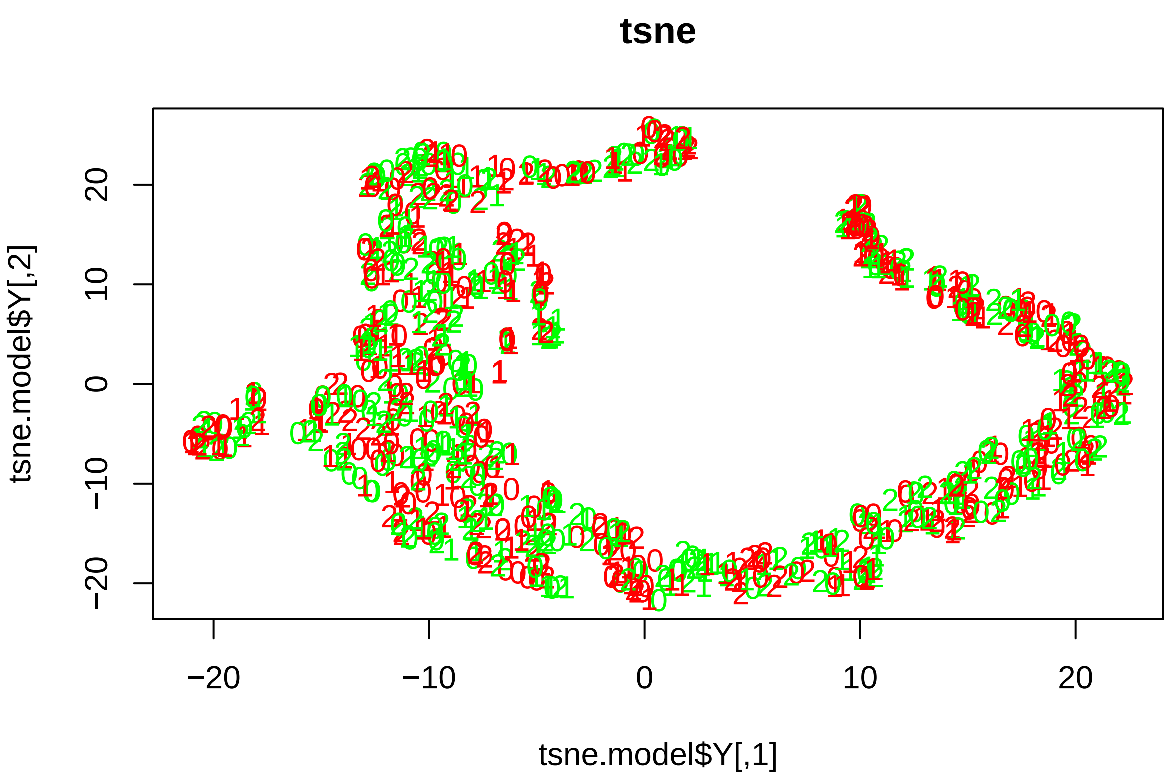


t-Distributed Stochastic Neighbor Embedding(t-SNE)

```
tsne.model <- Rtsne(dr.pca, dims = 2, perplexity = 30, verbose=TRUE, max_iter = 500)
```

```
## Performing PCA
## Read the 1000 x 14 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.19 seconds (sparsity = 0.103902)!
## Learning embedding...
## Iteration 50: error is 61.947611 (50 iterations in 0.15 seconds)
## Iteration 100: error is 54.938019 (50 iterations in 0.10 seconds)
## Iteration 150: error is 54.209012 (50 iterations in 0.13 seconds)
## Iteration 200: error is 53.984300 (50 iterations in 0.13 seconds)
## Iteration 250: error is 53.872141 (50 iterations in 0.13 seconds)
## Iteration 300: error is 0.794425 (50 iterations in 0.09 seconds)
## Iteration 350: error is 0.627833 (50 iterations in 0.11 seconds)
## Iteration 400: error is 0.585119 (50 iterations in 0.12 seconds)
## Iteration 450: error is 0.563899 (50 iterations in 0.13 seconds)
## Iteration 500: error is 0.552464 (50 iterations in 0.11 seconds)
## Fitting performed in 1.20 seconds.
```

```
# Plotting using Branch as factors
colors = rainbow(length(unique(dr.pca$BRANCH)))
names(colors) = unique(dr.pca$BRANCH)

# plotting our graph
options(repr.plot.width = 15, repr.plot.height = 15)
plot(tsne.model$Y, t = 'n', main = 'tsne')
text(tsne.model$Y, labels = dr.pca$BRANCH,
col = colors[dr.pca$BRANCH])
```



COnclusion

Although the PCA model performs better than the tSNE model, PC1 does not do a good job of summarizing most of the variables.

Even the top 4 Principal Components do not provide anything close to 75% explainability of the data which would have been the irreducible minimum

PART 3

ASSOCIATION RULES

Association Rules is an unsupervised learning technique (No Target Variable) used to discover patterns in big data that is usually unstructured

1. Loading the Dataset

```r
# Loading the dataset using a special read in function as the data is very
#unstructured
path <-"http://bit.ly/SupermarketDatasetII"
assoc <-read.transactions(path, sep = ",")
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

```r
#Checking the first 10 transactions
inspect(assoc[1:10])
```

```
##      items
## [1]  {almonds,
##       antioxydant juice,
##       avocado,
##       cottage cheese,
##       energy drink,
##       frozen smoothie,
##       green grapes,
##       green tea,
##       honey,
##       low fat yogurt,
##       mineral water,
##       olive oil,
##       salad,
##       salmon,
##       shrimp,
##       spinach,
##       tomato juice,
##       vegetables mix,
##       whole weat flour,
##       yams}
## [2]  {burgers,
##       eggs,
##       meatballs}
## [3]  {chutney}
## [4]  {avocado,
##       turkey}
## [5]  {energy bar,
##       green tea,
##       milk,
##       mineral water,
##       whole wheat rice}
## [6]  {low fat yogurt}
## [7]  {french fries,
##       whole wheat pasta}
## [8]  {light cream,
##       shallot,
##       soup}
## [9]  {frozen vegetables,
##       green tea,
##       spaghetti}
## [10] {french fries}
```

```r
# previewing the column names
colnames(assoc)
```

```
##   [1] "almonds"              "antioxydant juice"    "asparagus"
##   [4] "avocado"              "babies food"          "bacon"
##   [7] "barbecue sauce"       "black tea"            "blueberries"
##  [10] "body spray"           "bramble"              "brownies"
##  [13] "bug spray"            "burger sauce"         "burgers"
##  [16] "butter"               "cake"                 "candy bars"
##  [19] "carrots"              "cauliflower"          "cereals"
##  [22] "champagne"            "chicken"              "chili"
##  [25] "chocolate"            "chocolate bread"      "chutney"
##  [28] "cider"                "clothes accessories"  "cookies"
##  [31] "cooking oil"          "corn"                 "cottage cheese"
##  [34] "cream"                "dessert wine"         "eggplant"
##  [37] "eggs"                 "energy bar"           "energy drink"
##  [40] "escalope"             "extra dark chocolate" "flax seed"
##  [43] "french fries"         "french wine"          "fresh bread"
##  [46] "fresh tuna"           "fromage blanc"        "frozen smoothie"
##  [49] "frozen vegetables"    "gluten free bar"      "grated cheese"
##  [52] "green beans"          "green grapes"         "green tea"
##  [55] "ground beef"          "gums"                 "ham"
##  [58] "hand protein bar"     "herb & pepper"        "honey"
##  [61] "hot dogs"             "ketchup"              "light cream"
##  [64] "light mayo"           "low fat yogurt"       "magazines"
##  [67] "mashed potato"        "mayonnaise"           "meatballs"
##  [70] "melons"               "milk"                 "mineral water"
##  [73] "mint"                 "mint green tea"       "muffins"
##  [76] "mushroom cream sauce" "napkins"              "nonfat milk"
##  [79] "oatmeal"              "oil"                  "olive oil"
##  [82] "pancakes"             "parmesan cheese"      "pasta"
##  [85] "pepper"               "pet food"             "pickles"
##  [88] "protein bar"          "red wine"             "rice"
##  [91] "salad"                "salmon"               "salt"
##  [94] "sandwich"             "shallot"              "shampoo"
##  [97] "shrimp"               "soda"                 "soup"
## [100] "spaghetti"            "sparkling water"      "spinach"
## [103] "strawberries"         "strong cheese"        "tea"
## [106] "tomato juice"         "tomato sauce"         "tomatoes"
## [109] "toothpaste"           "turkey"               "vegetables mix"
## [112] "water spray"          "white wine"           "whole weat flour"
## [115] "whole wheat pasta"    "whole wheat rice"     "yams"
## [118] "yogurt cake"          "zucchini"
```

```r
#Summary of the data
summary(assoc)
```

```
## transactions as itemMatrix in sparse format with
##  7501 rows (elements/itemsets/transactions) and
##  119 columns (items) and a density of 0.03288973
##
## most frequent items:
## mineral water          eggs     spaghetti  french fries     chocolate
```

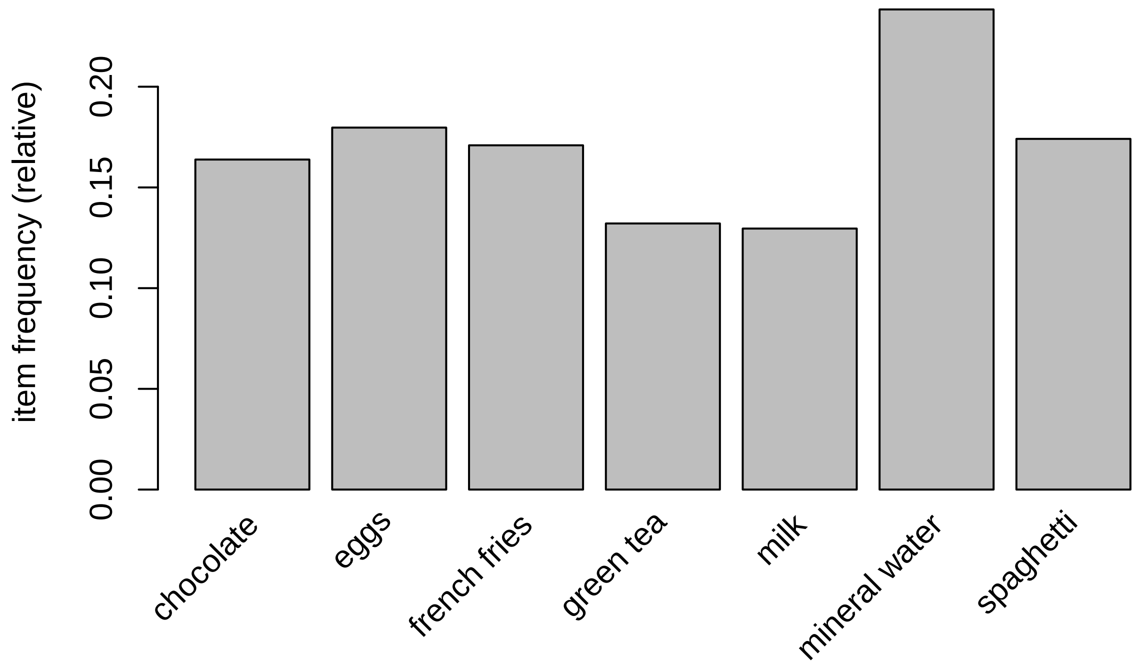
```
##          1788          1348          1306          1282          1229
##       (Other)
##         22405
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1754 1358 1044  816  667  493  391  324  259  139  102   67   40   22   17    4
##   18   19   20
##    1    2    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.914   5.000  20.000
##
## includes extended item information - examples:
##              labels
## 1           almonds
## 2 antioxydant juice
## 3         asparagus
```

```
#Our data has 7501 rows and 119 columns
#The Total number of non-empty cells in the Sparse itemMatrix 0.033

#About (7501 * 119 * 0.033) 29,500 items were purchased in all the transaction

#Mineral Water, Eggs, Spaghetti, French fries and Chocolate were the most purchased
#items in the transaction respectively

#Generally, most people buy few items for instance 1754 transactions were for 1 item

#The biggest transaction was for 20 items and only happened once

#The median number of items bought was 3 whilst the mean was 3.914

#Roughly 50% of the transactions had between 2 & 5 items purchased
```

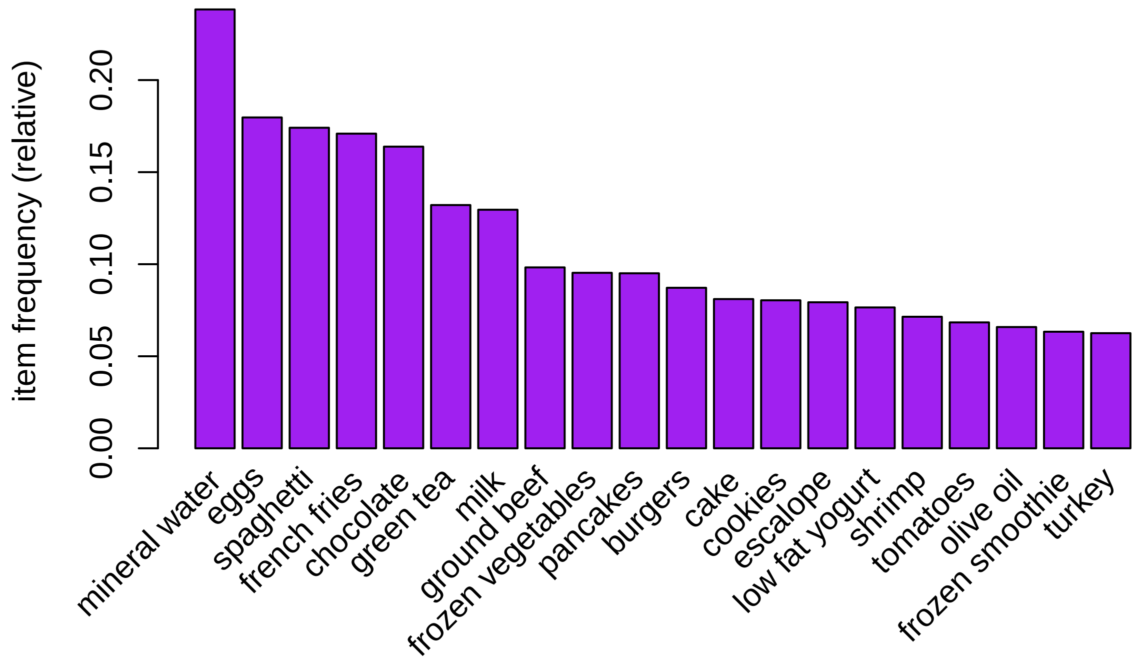
```
#Support

#High Support means high frequency

#Exploring the Support in the first column leveraging itemFrequency() from arules
itemFrequency(assoc[, 1])
```

```
##    almonds
## 0.02039728
```

```
#items with frquency of at least 10%
itemFrequencyPlot(assoc, support = 0.10)
```




```r
# Plot of the top 20 items in the transaction
itemFrequencyPlot(assoc, topN = 20, col = 'purple')
```




```
#Least Popular Items
least <- itemFrequency(assoc, type = "relative")
head(sort(least), n = 20)
```

```
##      water spray          napkins            cream          bramble
##     0.0003999467     0.0006665778     0.0009332089     0.0018664178
##              tea          chutney   mashed potato  chocolate bread
##     0.0038661512     0.0041327823     0.0041327823     0.0042660979
##     dessert wine          ketchup          oatmeal      babies food
##     0.0043994134     0.0043994134     0.0043994134     0.0045327290
##         sandwich        asparagus      cauliflower             corn
##     0.0045327290     0.0047993601     0.0047993601     0.0047993601
##            salad          shampoo hand protein bar  mint green tea
##     0.0049326756     0.0049326756     0.0051993068     0.0055992534
```

```
#Water Spray, Napkins, Cream and Bramble were among the least popular items
#?itemFrequencyPlot
```

Building the Model

```
# Creating model based on association rules using the apriori function from arules
# We use Min Support as 0.001 and confidence as 0.8
rules <- apriori (assoc, parameter = list(supp = 0.001, conf = 0.8))
```

```
## Apriori
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 7
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [116 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.00s].
## writing ... [74 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rules)
```

```
## set of 74 rules
##
## rule length distribution (lhs + rhs):sizes
##  3  4  5  6
## 15 42 16  1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   4.000   4.000   4.041   4.000   6.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001067   Min.   :0.8000   Min.   :0.001067   Min.   : 3.356
##  1st Qu.:0.001067   1st Qu.:0.8000   1st Qu.:0.001333   1st Qu.: 3.432
##  Median :0.001133   Median :0.8333   Median :0.001333   Median : 3.795
##  Mean   :0.001256   Mean   :0.8504   Mean   :0.001479   Mean   : 4.823
##  3rd Qu.:0.001333   3rd Qu.:0.8889   3rd Qu.:0.001600   3rd Qu.: 4.877
##  Max.   :0.002533   Max.   :1.0000   Max.   :0.002666   Max.   :12.722
##      count
##  Min.   : 8.000
##  1st Qu.: 8.000
##  Median : 8.500
##  Mean   : 9.419
##  3rd Qu.:10.000
##  Max.   :19.000
##
## mining info:
##   data ntransactions support confidence
##  assoc          7501   0.001        0.8
##                                                                  call
##  apriori(data = assoc, parameter = list(supp = 0.001, conf = 0.8))
```

```r
#We have 74 rules wherein of the 15, 3 of those rules were if one bought product A & B, there were
#more likely to buy product C

#42 of those rules had 4 items, 16 had 5 items and 1 had 6 items


#Inspecting the first two Rules
inspect(rules[1:2])
```

```
##     lhs                           rhs             support     confidence
## [1] {frozen smoothie, spinach} => {mineral water} 0.001066524 0.8888889
## [2] {bacon, pancakes}          => {spaghetti}     0.001733102 0.8125000
##     coverage    lift     count
## [1] 0.001199840 3.729058  8
## [2] 0.002133049 4.666587 13
```

```r
#The customers who bought Frozen Smoothie and Spinach were most likely to buy Mineral water

#The Customers who bacon and pancakes, were most likely to buy Spaghetti


#Inspecting the Rules

inspect(rules[42:43])
```

```
##     lhs                                  rhs             support
## [1] {french fries, herb & pepper, milk} => {mineral water} 0.001199840
## [2] {chocolate, soup, turkey}           => {mineral water} 0.001066524
##     confidence coverage    lift     count
## [1] 0.8181818  0.001466471 3.432428 9
## [2] 0.8888889  0.001199840 3.729058 8
```

```r
#The customers who bought French fries, herb& pepper and milk were more likely to buy Mineral Water
#same as the customers who bought chocolate, soup and turkey

inspect(rules[72:74])
```

```
##     lhs                     rhs                     support confidence    coverage     lift count
## [1] {french fries,
##      milk,
##      pancakes,
##      spaghetti}         => {mineral water}     0.001066524   0.8000000 0.001333156 3.356152     8
## [2] {chocolate,
##      eggs,
##      frozen vegetables,
##      ground beef}       => {mineral water}     0.001466471   0.8461538 0.001733102 3.549776    11
## [3] {chocolate,
##      ground beef,
##      milk,
##      mineral water,
##      spaghetti}         => {frozen vegetables} 0.001066524   0.8888889 0.001199840 9.325253     8
```

```r
#Customers who bought french fries, milk, pancakes & spaghetti were more likely to buy mineral water
# as were customers who bought chocolate, eggs, frozen veg and ground beef

#For customers who bought 6 items i.e Chocolate, ground beef, milk, mineral water and spaghetti, they
#were most likely to buy Frozen Vegetables


#Sorting by highest lift in 6 records

inspect(sort(rules, by ="lift")[1:6])
```

```
##     lhs                           rhs                     support confidence    coverage      lift count
## [1] {eggs,
##      mineral water,
##      pasta}                    => {shrimp}            0.001333156  0.9090909 0.001466471 12.722185    10
## [2] {french fries,
##      mushroom cream sauce,
##      pasta}                    => {escalope}          0.001066524  1.0000000 0.001066524 12.606723     8
## [3] {milk,
##      pasta}                    => {shrimp}            0.001599787  0.8571429 0.001866418 11.995203    12
## [4] {mushroom cream sauce,
##      pasta}                    => {escalope}          0.002532996  0.9500000 0.002666311 11.976387    19
## [5] {chocolate,
##      ground beef,
##      milk,
##      mineral water,
##      spaghetti}                => {frozen vegetables} 0.001066524  0.8888889 0.001199840  9.325253     8
## [6] {herb & pepper,
##      mineral water,
##      rice}                     => {ground beef}       0.001333156  0.9090909 0.001466471  9.252498    10
```

```r
#Highly unlikely that one buys eggs, mineral water, pasta and the next thing they buy is Shrimp
```

```r
#What if we wanted to promote Spaghetti as a product
spaghetti <- subset(rules, subset = rhs %pin% "spaghetti")

# Then order by confidence
spaghetti<-sort(spaghetti, by="confidence", decreasing=TRUE)
spaghetti
```

```
## set of 16 rules
```

```r
#There are a set of 16 Rules

#What products would people be more likely to buy before buying Spaghetti?

inspect(spaghetti)
```

```
##      lhs                         rhs            support confidence    coverage     lift count
## [1]  {light cream,
##       mineral water,
```

```
##       shrimp}             => {spaghetti} 0.001066524  0.8888889 0.001199840 5.105326     8
## [2]  {ground beef,
##       salmon,
##       shrimp}             => {spaghetti} 0.001066524  0.8888889 0.001199840 5.105326     8
## [3]  {burgers,
##       milk,
##       salmon}             => {spaghetti} 0.001066524  0.8888889 0.001199840 5.105326     8
## [4]  {frozen vegetables,
##       ground beef,
##       mineral water,
##       shrimp}             => {spaghetti} 0.001733102  0.8666667 0.001999733 4.977693    13
## [5]  {burgers,
##       frozen vegetables,
##       pancakes}           => {spaghetti} 0.001466471  0.8461538 0.001733102 4.859877    11
## [6]  {frozen vegetables,
##       olive oil,
##       tomatoes}           => {spaghetti} 0.002133049  0.8421053 0.002532996 4.836624    16
## [7]  {green tea,
##       ground beef,
##       tomato sauce}       => {spaghetti} 0.001333156  0.8333333 0.001599787 4.786243    10
## [8]  {frozen vegetables,
##       tomatoes,
##       whole wheat rice}   => {spaghetti} 0.001333156  0.8333333 0.001599787 4.786243    10
## [9]  {chicken,
##       protein bar}        => {spaghetti} 0.001199840  0.8181818 0.001466471 4.699220     9
## [10] {frozen vegetables,
##       ground beef,
##       mineral water,
##       tomatoes}           => {spaghetti} 0.001199840  0.8181818 0.001466471 4.699220     9
## [11] {bacon,
##       pancakes}           => {spaghetti} 0.001733102  0.8125000 0.002133049 4.666587    13
## [12] {milk,
##       mineral water,
##       parmesan cheese}    => {spaghetti} 0.001066524  0.8000000 0.001333156 4.594793     8
## [13] {cooking oil,
##       mineral water,
##       red wine}           => {spaghetti} 0.001066524  0.8000000 0.001333156 4.594793     8
## [14] {avocado,
##       burgers,
##       milk}               => {spaghetti} 0.001066524  0.8000000 0.001333156 4.594793     8
## [15] {frozen vegetables,
##       mineral water,
##       olive oil,
##       tomatoes}           => {spaghetti} 0.001066524  0.8000000 0.001333156 4.594793     8
## [16] {chocolate,
##       french fries,
##       mineral water,
##       olive oil}          => {spaghetti} 0.001066524  0.8000000 0.001333156 4.594793     8
```

Conclusion

Surprising to see some items that intuitively, a potential customer would buy alongside other items but they did not e.g., Milk and Tea

Recommendation

Perhaps more efforts could be invested in displaying associated items in the same supermarket isles to boost sales or discount the least popular items that go well with certain families but only if they are bought together

ANOMALY DETECTION

In this section, we will investigate whether there are any anomalies in the Sales dataset

The objective of this investigation is the determination of whether there were any fraudulent transactions in the sales

Loading the Data

```
sales <- read.csv("http://bit.ly/CarreFourSalesDataset")
```

```
#Checking out the first 6 records
head(sales)
```

```
##        Date    Sales
## 1  1/5/2019 548.9715
## 2  3/8/2019  80.2200
## 3  3/3/2019 340.5255
## 4 1/27/2019 489.0480
## 5  2/8/2019 634.3785
## 6 3/25/2019 627.6165
```

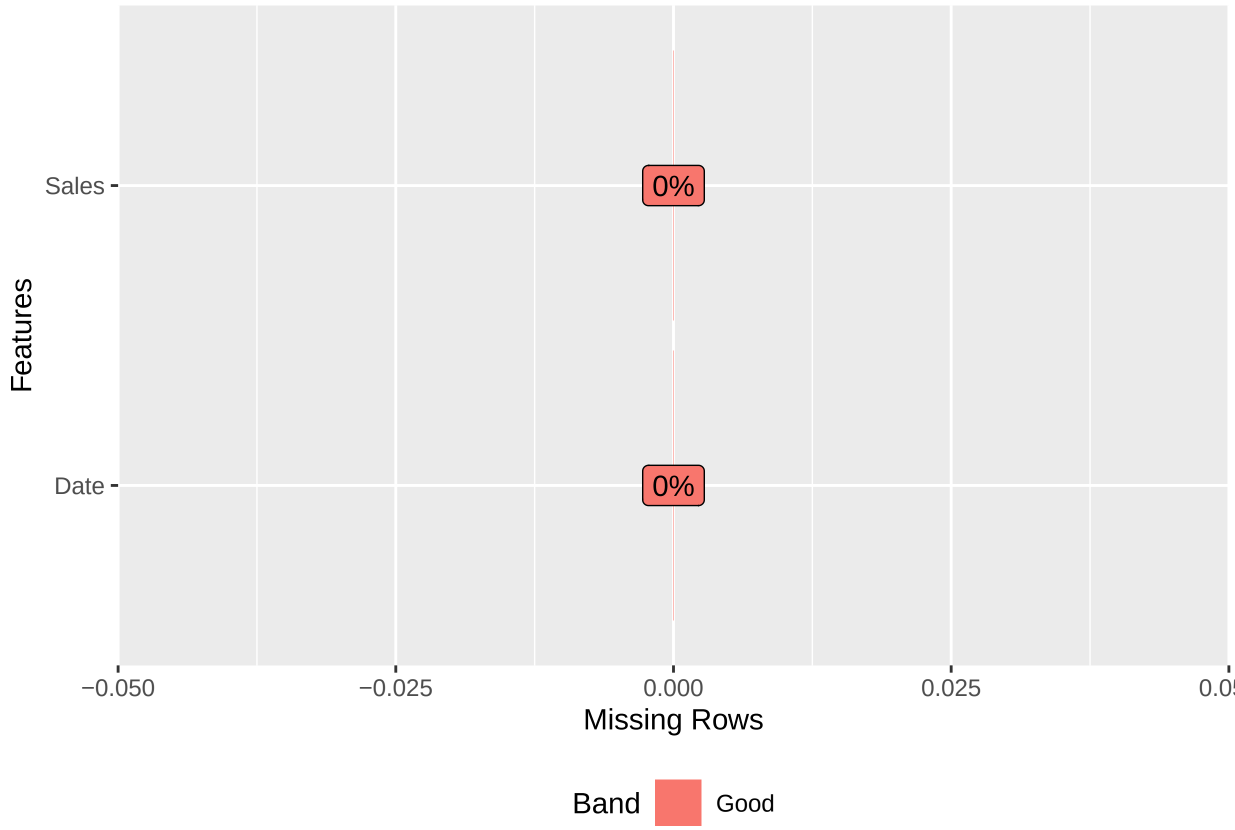
```
#Checking Dimensions of the Data

dim(sales)
```

```
## [1] 1000    2
```

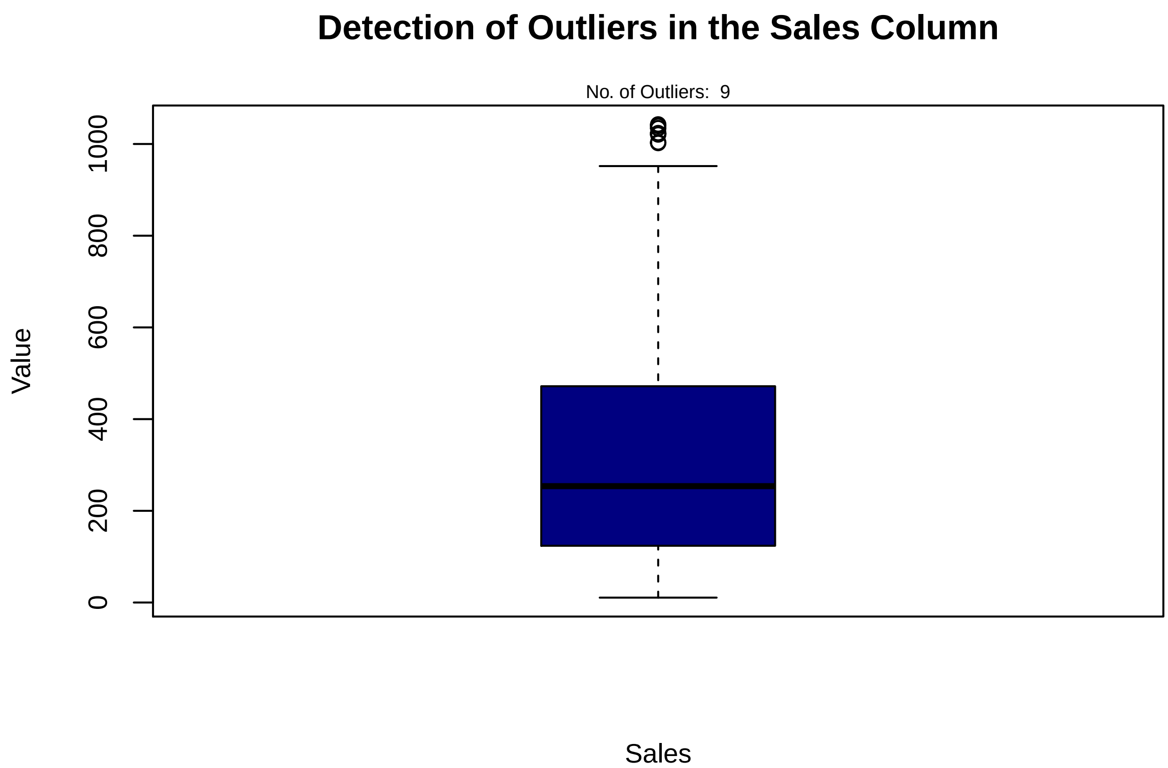
```
#There are 2 Columns and 1000 rows in the dataset


#Checking for any duplicated rows
sum(duplicated(sales))
```

```
## [1] 0
```

```
#There are no duplicates in the data

#Plot and Distribution of Missing values
plot_missing(sales)
```




```
#There are no Missing records in the dataset


#Outlier Detection in the Sales Column using Boxplot

options(repr.plot.width = 12, repr.plot.height =10)

boxplot(sales$Sales, main="Detection of Outliers in the Sales Column", xlab = "Sales", ylab = "Value", 

sales_outlier <- boxplot.stats(sales$Sales)$out
mtext(paste("No. of Outliers: ", paste(length(sales_outlier), collapse=", ")), cex=0.6)
```




```r
#We have outliers in the Sales Column

#We retain them in the data as they represent actual data points


# Getting the right Date format
sales$Date = as.Date(sales$Date, format = "%m/%d/%y")

#Previewing the Date Column
head(unique(sales$Date))
```

```
## [1] "2020-01-05" "2020-03-08" "2020-03-03" "2020-01-27" "2020-02-08"
## [6] "2020-03-25"
```

```r
#The sales data was collected in 2020

#Separating the Date COlumn into Year, Month and Day
sales$Year <- year(ymd(sales$Date))

sales$Month <- month(ymd(sales$Date))

sales$Day <- day(ymd(sales$Date))

#Previewing the data structure

str(sales)
```

```
## 'data.frame':    1000 obs. of  5 variables:
##  $ Date : Date, format: "2020-01-05" "2020-03-08" ...
##  $ Sales: num  549 80.2 340.5 489 634.4 ...
##  $ Year : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ Month: int  1 3 3 1 2 3 2 2 1 2 ...
##  $ Day  : int  5 8 3 27 8 25 25 24 10 20 ...
```

```r
#Dropping the Year Column

sales <- select(sales, -Year)

#Checking for Unique values in the Month variable

unique(sales$Month)
```

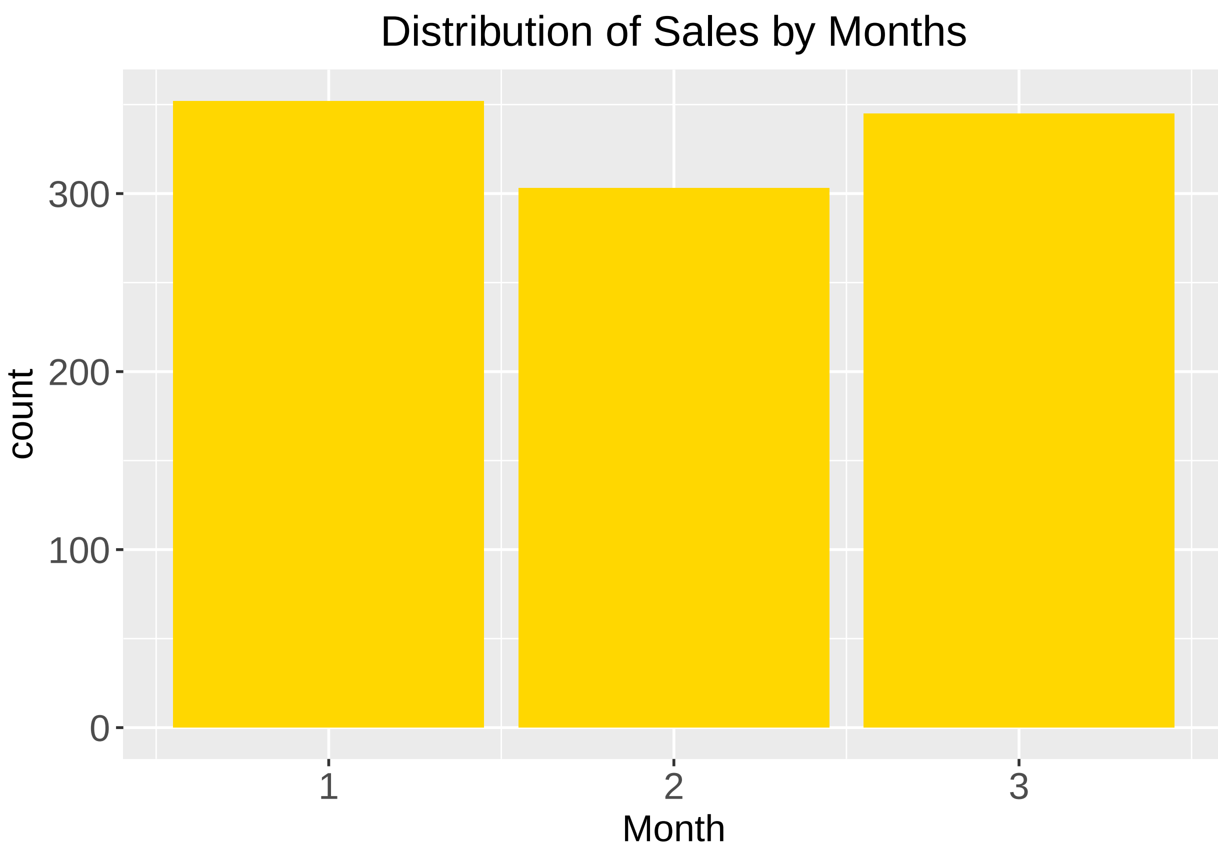
```
## [1] 1 3 2
```

```r
#There are three months in the Month variable i,e., Jan, Feb and March
```

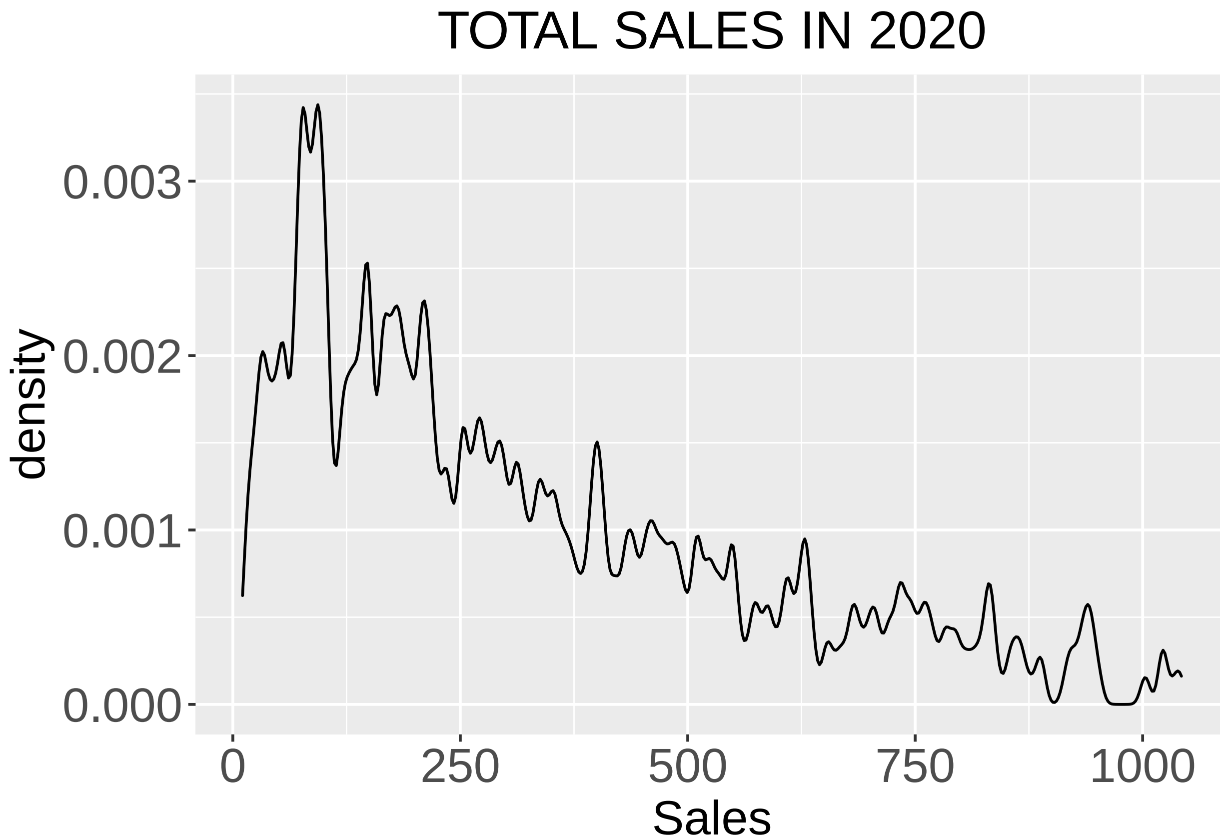
EXPLORATORY DATA ANALYSIS

Univariate Analysis

```r
# DIstribution by Months
ggplot(sales, aes(x = Month)) +
  geom_bar( position= "dodge", fill= 'gold') +

  ggtitle("Distribution of Sales by Months") +
   theme(axis.text = element_text(size=14),
          axis.title = element_text(size = 14),
          plot.title = element_text(hjust = 0.5, size = 16))
```

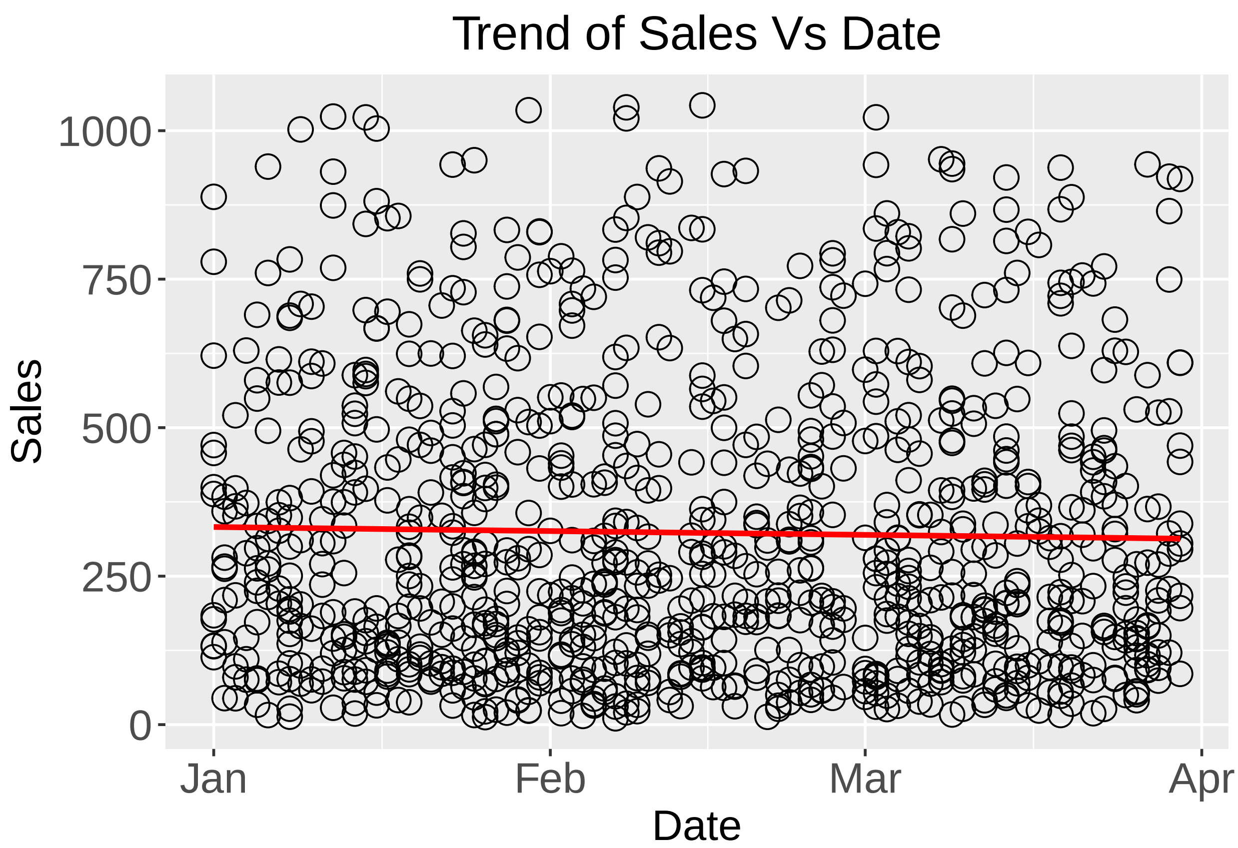



```
#January had more sales than any other months whilst February had the least number of Sales


#Density Plot Distribution of the Total Column
ggplot(sales, aes(x= Sales)) +
geom_density(bw = 5) +
ggtitle("TOTAL SALES IN 2020") +
 theme(axis.text = element_text(size=18),
          axis.title = element_text(size = 18),
          plot.title = element_text(hjust = 0.5, size = 20))
```



Bivariate Analysis

```
options(repr.plot.width = 20, repr.plot.height = 20)
ggplot(sales, aes(x = Date, y=Sales)) +
  geom_point(size = 4, shape=1) +
  geom_smooth(aes(group=1), method= 'lm', formula = y~log(x), se=F, color ='red') +
  ggtitle("Trend of Sales Vs Date") +  theme(axis.text = element_text(size=16),
          axis.title = element_text(size = 16),
          plot.title = element_text(hjust = 0.5, size = 18))
```



Implementing the Solution

Building the Model

```
sales <- sales[order(sales$Date),]

# Dataset as a tibble
tib <- as_tibble(sales)

#Aggregation of Sales by Date
sales.tib <- aggregate(tib["Sales"], by=list(tib$Date), sum)

sales_tibble <- as_tibble(sales.tib)


anom <- sales_tibble %>%
    time_decompose(Sales, merge = TRUE) %>%
    anomalize(remainder, alpha = 0.25) %>%
    time_recompose()
```

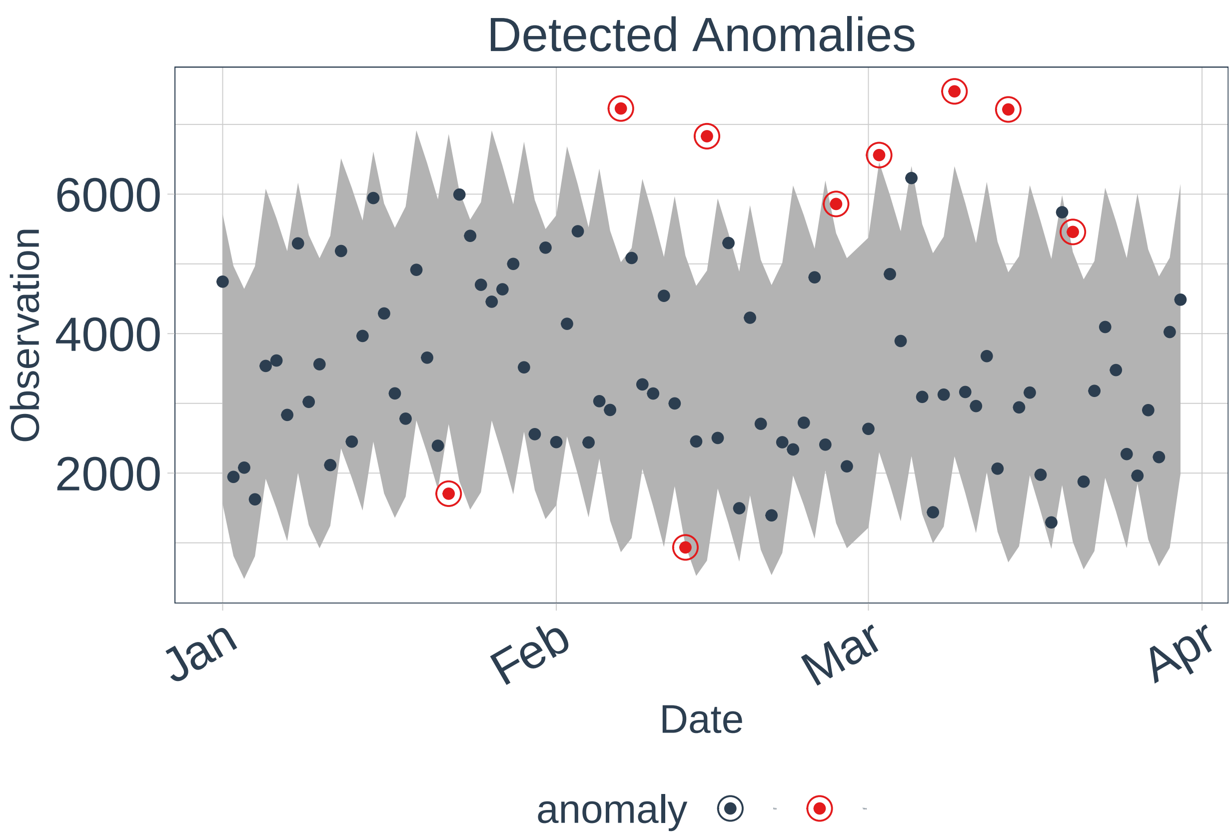
```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Group.1
```

```
## frequency = 7 days
```

```
## trend = 30 days
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

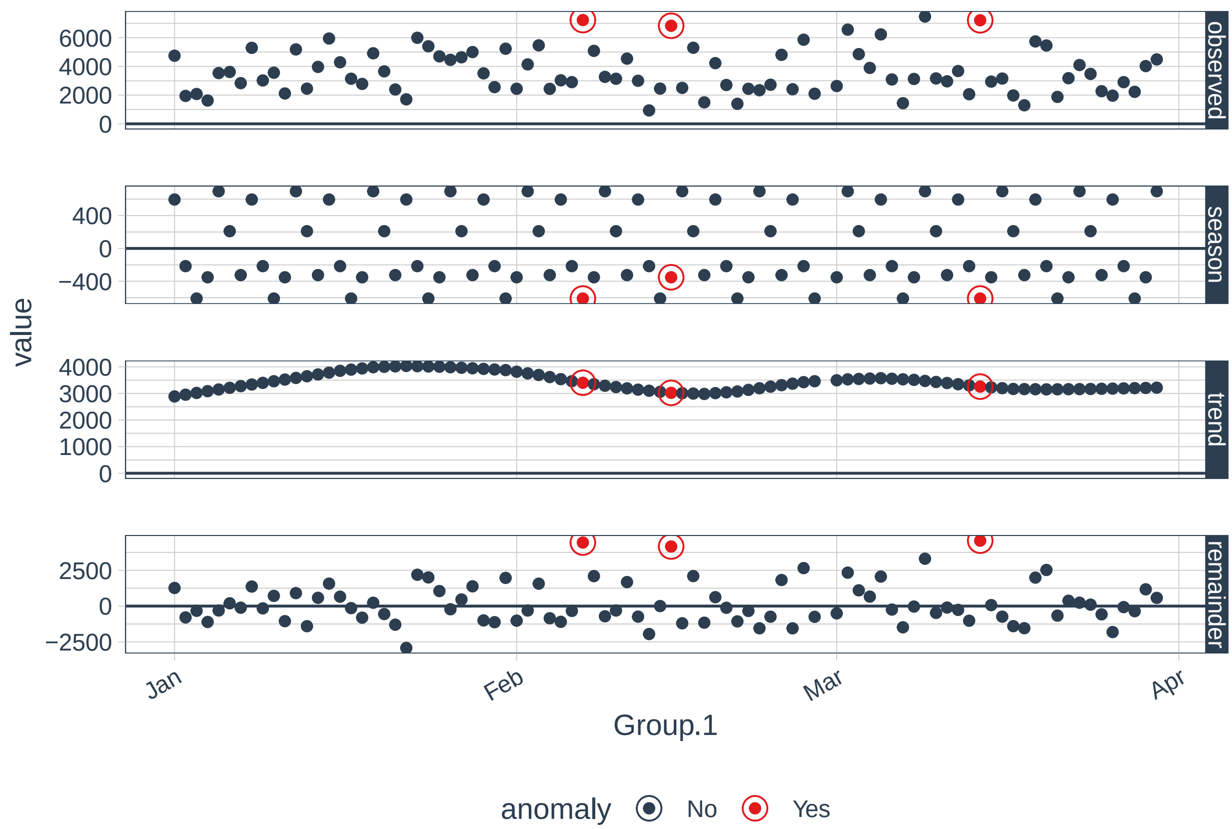
```
# Plotting the anomalies

options(repr.plot.width = 20, repr.plot.height = 20)
anom %>% plot_anomalies(time_recomposed = TRUE) +
    labs(title = "Detected Anomalies", x = "Date", y = "Observation") + theme(axis.text = element_text(s
          axis.title = element_text(size = 15),
          plot.title = element_text(hjust = 0.5, size = 18),
          legend.title = element_text(size=15),
          legend.text = element_text(size=1))
```




```
anomaly.detect <- sales_tibble %>%
time_decompose(Sales, method = "stl", frequency = "auto", trend = "auto") %>%
anomalize(remainder, method = "gesd", alpha = 0.05, max_anoms = 0.2) %>%
plot_anomaly_decomposition()
```

```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Group.1

## frequency = 7 days

## trend = 30 days
```

```
options(repr.plot.width = 15, repr.plot.height = 12)
anomaly.detect
```




```
#Extracting the Potentially Fraudulent Data Points
sales_tibble %>%
   time_decompose(Sales, merge = TRUE) %>%
    anomalize(remainder, alpha = 0.25) %>%
    time_recompose() %>%
    filter(anomaly == 'Yes')
```

```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Group.1

## frequency = 7 days

## trend = 30 days

## # A time tibble: 9 x 11
## # Index: Group.1
##   Group.1     Sales observed season trend remainder remainder_l1 remainder_l2
##   <date>      <dbl>    <dbl>  <dbl> <dbl>     <dbl>        <dbl>        <dbl>
## 1 2020-01-22 1705.     1705.   596. 4032.    -2923.       -1925.        2234.
## 2 2020-02-07 7228.     7228.  -610. 3402.     4437.       -1925.        2234.
## 3 2020-02-13  934.      934.  -216. 3101.    -1951.       -1925.        2234.
```

```
## 4 2020-02-15 6831.    6831.  -351. 3022.     4161.       -1925.        2234.
## 5 2020-02-27 5859.    5859.  -216. 3423.     2652.       -1925.        2234.
## 6 2020-03-02 6560.    6560.   696. 3529.     2335.       -1925.        2234.
## 7 2020-03-09 7474.    7474.   696. 3470.     3308.       -1925.        2234.
## 8 2020-03-14 7215.    7215.  -610. 3256.     4569.       -1925.        2234.
## 9 2020-03-20 5458.    5458.  -216. 3152.     2522.       -1925.        2234.
## # ... with 3 more variables: anomaly <chr>, recomposed_l1 <dbl>,
## #   recomposed_l2 <dbl>
```

Follow Up Questions

1. Did we have the right data? Yes, we did.
2. Did we ask the right questions? Yes, we did

Conclusion

Our Model has identified the potentially fraudulent transaction

Recommendation

The relevant teams can take it forward by investigating further those transactions