Week 14 IP

Ted Askoye

9/23/2020

1. Business Understanding

1 a.) Defining the Question

You are a Data analyst at Carrefour Kenya and are currently undertaking a project that will inform the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax). Hence making an analysis to Customer Data from a the supermarket and implement dimensionality reduction.

2. Defining the Metrics of Success

The success of this analysis will occur when we analysis the customer data to understand it fully and later implementing the appropriate dimensionality reduction techniques.

3. Context

Dimensionality reduction is the process of reducing the number of random variables under review, by getting a set of principal variables. It can be divided into feature selection and feature extraction and is important for the visualization of features while it also helps deal with multicollinearity of the features.

4. Experimental Design

We will define the question, the metric of success, context and experimental design taken. This will be followed by reading and exploring the dataset and its appropriateness of the available data to answer the given question. This will be followed by cleaning the data off outliers, anomalies and null values from missing data, perfom an exploratory data analysis after which we will Implement feature extraction and feature selection,record our observations and provide a conclusion and reccomendation.

5. Data Relevance

Our data is very relevant to our research question.

6. Loading relevant Libraries and Reading the Data

```
# Importing the required packages

library("data.table")
library("plyr")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

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

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------------------------------------------- tid

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v stringr 1.4.0
## v tidyr   1.1.2     v forcats 0.5.0
## v readr   1.3.1

## -- Conflicts ------------------------------------------------------------------------------ tidyverse_
## x dplyr::arrange()   masks plyr::arrange()
## x dplyr::between()   masks data.table::between()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
```

```
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x dplyr::mutate()    masks plyr::mutate()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
## x purrr::transpose() masks data.table::transpose()
```

```
library("tidyr")
library("lubridate")
```

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

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

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

```
library("ggcorrplot")
library("ggplot2")
library("corrplot")
```

```
## corrplot 0.84 loaded
```

```
library("moments")
library('xtable')
library('countrycode')
library('class')
library("rpart")
library("rpart.plot")
library("mlbench")
library('e1071')
```

```
##
## Attaching package: 'e1071'
```

```
## The following objects are masked from 'package:moments':
##
##     kurtosis, moment, skewness
```

```
library('rpart')
library('caret')
```

```
## Loading required package: lattice
```

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

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

```r
library('ranger')
library('kernlab')
```

```
##
## Attaching package: 'kernlab'

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

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
library('ggbiplot')
```

```
## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:kernlab':
##
##     alpha

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

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

## Loading required package: grid
```

```r
library('ISLR')
library('devtools')
```

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

```r
# Loading the Dataset

cf_df <- read.csv(url("http://bit.ly/CarreFourDataset"))
```

Previewing the data

```r
# Previewing The First Seven records in the Dataset

head(cf_df, n=7)
```

```
##    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
## 7 355-53-5943      A        Member Female Electronic accessories      68.84
##   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
## 7        6 20.6520 2/25/2019 14:36     Ewallet 413.04                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
## 7      20.6520    5.8 433.6920
```

```r
# Previewing The Last Seven records in the Dataset

tail(cf_df, n=7)
```

```
##       Invoice.ID Branch Customer.type Gender           Product.line Unit.price
## 994  690-01-6631      B        Normal   Male    Fashion accessories      17.49
## 995  652-49-6720      C        Member Female Electronic accessories      60.95
## 996  233-67-5758      C        Normal   Male      Health and beauty      40.35
## 997  303-96-2227      B        Normal Female     Home and lifestyle      97.38
## 998  727-02-1313      A        Member   Male     Food and beverages      31.84
## 999  347-56-2442      A        Normal   Male     Home and lifestyle      65.82
## 1000 849-09-3807      A        Member Female    Fashion accessories      88.34
##      Quantity     Tax      Date  Time Payment   cogs gross.margin.percentage
## 994        10  8.7450 2/22/2019 18:35 Ewallet 174.90                4.761905
## 995         1  3.0475 2/18/2019 11:40 Ewallet  60.95                4.761905
```

```
## 996         1  2.0175 1/29/2019 13:46 Ewallet  40.35                4.761905
## 997        10 48.6900  3/2/2019 17:16 Ewallet 973.80                4.761905
## 998         1  1.5920  2/9/2019 13:22    Cash  31.84                4.761905
## 999         1  3.2910 2/22/2019 15:33    Cash  65.82                4.761905
## 1000        7 30.9190 2/18/2019 13:28    Cash 618.38                4.761905
##      gross.income Rating     Total
## 994        8.7450    6.6  183.6450
## 995        3.0475    5.9   63.9975
## 996        2.0175    6.2   42.3675
## 997       48.6900    4.4 1022.4900
## 998        1.5920    7.7   33.4320
## 999        3.2910    4.1   69.1110
## 1000      30.9190    6.6  649.2990
```

```r
# Checking the Data Dimensions

dim(cf_df)
```

```
## [1] 1000   16
```

The dataset has 1000 records and 10 columns

```r
# Checking the Structure of the Dataset

str(cf_df)
```

```
## '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" "H
##  $ 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 ...
```

```r
# Checking The Data present in each column

glimpse(cf_df)
```

```
## Rows: 1,000
## Columns: 16
```

```
## $ Invoice.ID              <chr> "750-67-8428", "226-31-3081", "631-41-3108"...
## $ Branch                  <chr> "A", "C", "A", "A", "A", "C", "A", "C", "A"...
## $ Customer.type           <chr> "Member", "Normal", "Normal", "Member", "No...
## $ Gender                  <chr> "Female", "Female", "Male", "Male", "Male",...
## $ Product.line            <chr> "Health and beauty", "Electronic accessorie...
## $ Unit.price              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ Quantity                <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ Tax                     <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ Date                    <chr> "1/5/2019", "3/8/2019", "3/3/2019", "1/27/2...
## $ Time                    <chr> "13:08", "10:29", "13:23", "20:33", "10:37"...
## $ Payment                 <chr> "Ewallet", "Cash", "Credit card", "Ewallet"...
## $ cogs                    <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.income            <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ Rating                  <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ Total                   <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634....
```

7. Data Preparation

Uniformity

```
# Check column names

colnames(cf_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"
```

```
# Renaming column names

names(cf_df)[1]<- 'Invoice_ID'
names(cf_df)[3]<- 'Customer_type'
names(cf_df)[5]<- 'Product_line'
names(cf_df)[6]<- 'Unit_price'
names(cf_df)[14]<- 'Gross_income'

# Checking whether the column names have been changed

colnames(cf_df)
```

We'll rename the column names for Uniformity purposes

```
##  [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 for the length of unique values in each column

lapply(cf_df, function (x) {length(unique(x))})
```

```
## $Invoice_ID
## [1] 1000
##
## $Branch
## [1] 3
##
## $Customer_type
## [1] 2
##
## $Gender
## [1] 2
##
## $Product_line
## [1] 6
##
## $Unit_price
## [1] 943
##
## $Quantity
## [1] 10
##
## $Tax
## [1] 990
##
## $Date
## [1] 89
##
## $Time
## [1] 506
##
## $Payment
## [1] 3
##
## $cogs
## [1] 990
##
## $gross.margin.percentage
## [1] 1
##
## $Gross_income
## [1] 990
```

```
##
## $Rating
## [1] 61
##
## $Total
## [1] 990
```

```r
# Cheking if Tax and gross columns are duplicated

unique(cf_df$Tax == cf_df$Gross_income)
```

```
## [1] TRUE
```

```r
# Drop the Gross Margin percentage and Tax (Tax and Gross income are duplicated ) column

cf_df <- cf_df[, -8]
cf_df <- cf_df[, -12]

dim(cf_df)
```

Gross income percentage' has one unique variable making it redundant in our analysis.

```
## [1] 1000   14
```

Completeness

```r
# Checking for missing values

colSums(is.na(cf_df))
```

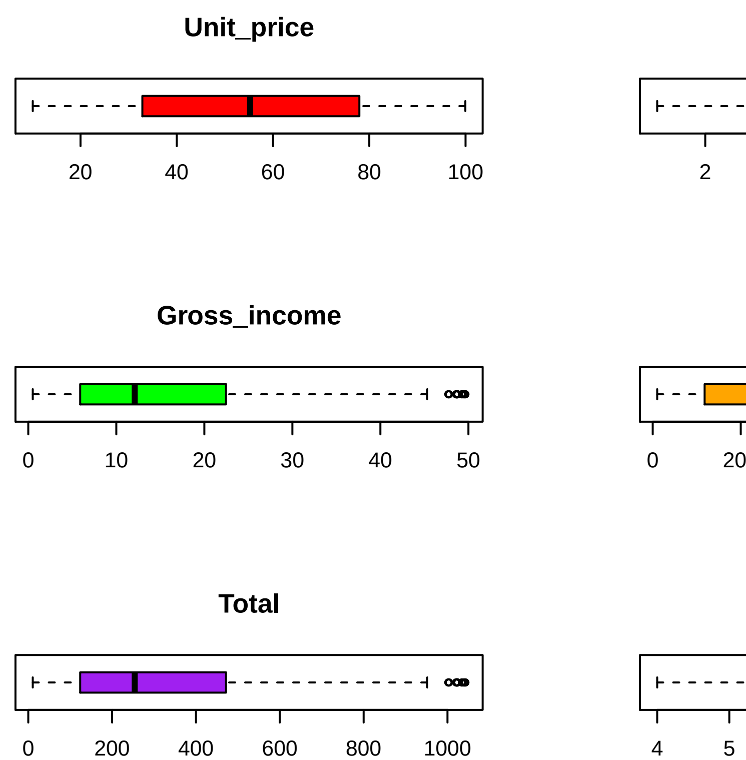
```
##    Invoice_ID        Branch Customer_type        Gender  Product_line
##             0             0             0             0             0
##    Unit_price      Quantity          Date          Time       Payment
##             0             0             0             0             0
##          cogs  Gross_income        Rating         Total
##             0             0             0             0
```

```r
# Checking for duplicate values

duplicates <- cf_df[duplicated(cf_df),]
duplicates
```

```
##  [1] Invoice_ID    Branch        Customer_type Gender        Product_line
##  [6] Unit_price    Quantity      Date          Time          Payment
## [11] cogs          Gross_income  Rating        Total
## <0 rows> (or 0-length row.names)
```

Outlier Detection

```r
# Plotting boxplots for all the numerical variables

par(mfrow=c(3,2))
boxplot((cf_df$`Unit_price`), horizontal = TRUE, col = 'red', main = "Unit_price")
boxplot((cf_df$`Quantity`), horizontal = TRUE, col = 'blue', main = "Quantity")
boxplot((cf_df$`Gross`), horizontal = TRUE, col = 'green', main = "Gross_income")
boxplot((cf_df$`cogs`), horizontal = TRUE, col = 'orange', main = "cogs")
boxplot((cf_df$`Total`), horizontal = TRUE, col = 'purple', main = "Total")
boxplot((cf_df$`Rating`), horizontal = TRUE, col = 'skyblue', main = "Rating")
```



 # Checking for anomalies in our numerical variables
We will not remove the outliers as they may have vital information

8. Exploratory Data Analysis

Univariate Analysis

```r
# Checking the statistical summary of the data

summary(cf_df)
```

```
##   Invoice_ID           Branch          Customer_type         Gender
##  Length:1000        Length:1000        Length:1000        Length:1000
##  Class :character   Class :character   Class :character   Class :character
```

```
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  Product_line        Unit_price      Quantity        Date
##  Length:1000        Min.   :10.08   Min.   : 1.00   Length:1000
##  Class :character   1st Qu.:32.88   1st Qu.: 3.00   Class :character
##  Mode  :character   Median :55.23   Median : 5.00   Mode  :character
##                     Mean   :55.67   Mean   : 5.51
##                     3rd Qu.:77.94   3rd Qu.: 8.00
##                     Max.   :99.96   Max.   :10.00
##      Time             Payment              cogs         Gross_income
##  Length:1000        Length:1000        Min.   : 10.17   Min.   : 0.5085
##  Class :character   Class :character   1st Qu.:118.50   1st Qu.: 5.9249
##  Mode  :character   Mode  :character   Median :241.76   Median :12.0880
##                                        Mean   :307.59   Mean   :15.3794
##                                        3rd Qu.:448.90   3rd Qu.:22.4453
##                                        Max.   :993.00   Max.   :49.6500
##      Rating          Total
##  Min.   : 4.000   Min.   :  10.68
##  1st Qu.: 5.500   1st Qu.: 124.42
##  Median : 7.000   Median : 253.85
##  Mean   : 6.973   Mean   : 322.97
##  3rd Qu.: 8.500   3rd Qu.: 471.35
##  Max.   :10.000   Max.   :1042.65
```

Measures of Central Tendancy and Dispersion - Summary

Central Tendancy - Mode, Mean and Median

```r
# First, a function for mode will be created since R does not have a built in function.

getmode <- function(v) {
    uniqv <- unique(v)
    uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

```r
# Unit Price
mode.up <- getmode(cf_df$Unit_price)
mode.up
```

```
## [1] 83.77
```

```r
mean(cf_df$Unit_price)
```

```
## [1] 55.67213
```

```r
median(cf_df$Unit_price)
```

```
## [1] 55.23
```

```r
# Gross_income

mode.gi <- getmode(cf_df$Gross_income)
mode.gi
```

```
## [1] 39.48
```

```r
mean(cf_df$Gross_income)
```

```
## [1] 15.37937
```

```r
median(cf_df$Gross_income)
```

```
## [1] 12.088
```

```r
# Quantity

mode.quan <- getmode(cf_df$Quantity)
mode.quan
```

```
## [1] 10
```

```r
mean(cf_df$Quantity)
```

```
## [1] 5.51
```

```r
median(cf_df$Quantity)
```

```
## [1] 5
```

```r
# Cogs

mode.cogs <- getmode(cf_df$cogs)
mode.cogs
```

```
## [1] 789.6
```

```r
mean(cf_df$cogs)
```

```
## [1] 307.5874
```

```r
median(cf_df$cogs)
```

```
## [1] 241.76
```

```r
# Total
mode.total <- getmode(cf_df$Total)
mode.total
```

```
## [1] 829.08
```

```r
mean(cf_df$Total)
```

```
## [1] 322.9667
```

```r
median(cf_df$Total)
```

```
## [1] 253.848
```

```r
# Rating
mode.rating <- getmode(cf_df$Rating)
mode.rating
```

```
## [1] 6
```

```r
mean(cf_df$Rating)
```

```
## [1] 6.9727
```

```r
median(cf_df$Rating)
```

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

```r


# Unit price

hist((cf_df$`Unit_price`), col = 'orange', main = "Unit Price")
```

Measure of Dispersion and Histograms - Standard Deviation, Variance, Skewness, Kurtosis and

Range

```
sd.up <- sd(cf_df$Unit_price)
sd.up
```

```
## [1] 26.49463
```

```
var.up <- var(cf_df$Unit_price)
var.up
```

```
## [1] 701.9653
```

```
range.up <- range(cf_df$Unit_price)
range.up
```

```
## [1] 10.08 99.96
```

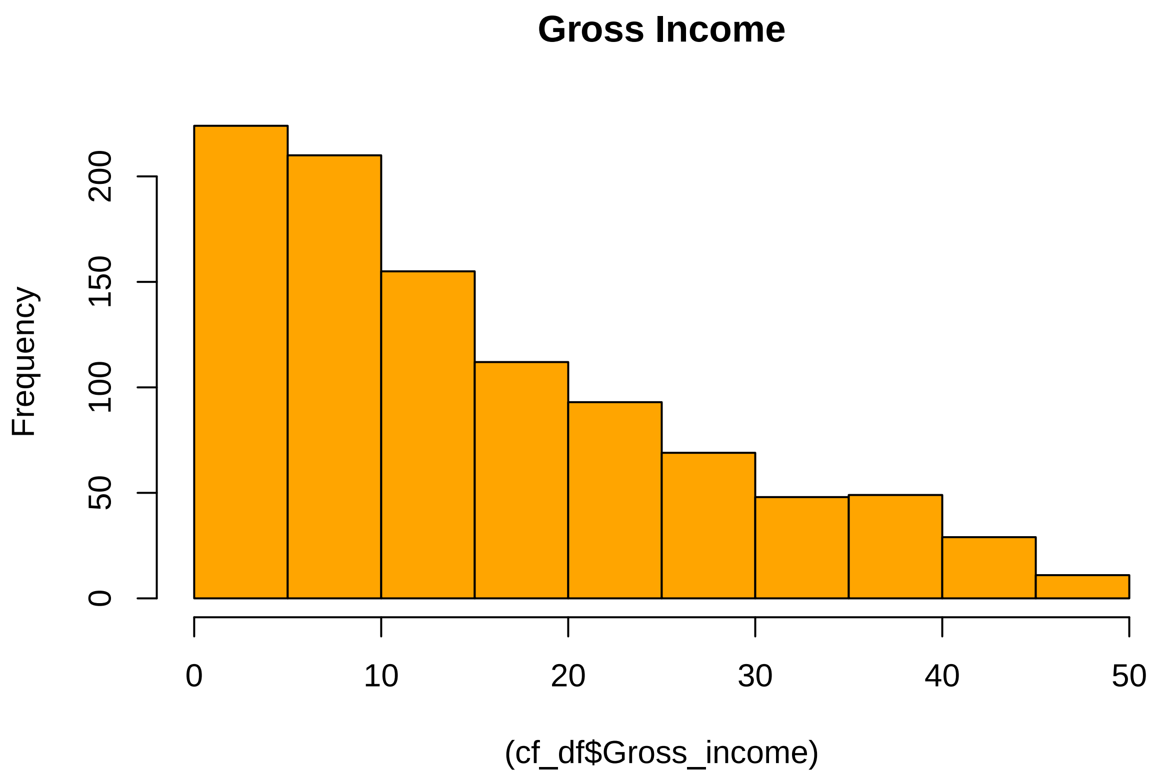
```
skew.up <- skewness(cf_df$Unit_price)
skew.up
```

```
## [1] 0.00705623
```

```
kurt.up <- kurtosis(cf_df$Unit_price)
kurt.up
```

```
## [1] -1.222062
```

```r
# Gross income

hist((cf_df$`Gross_income`), col = 'orange', main = "Gross Income")
```




```r
sd.gi <- sd(cf_df$Gross_income)
sd.gi
```

```
## [1] 11.70883
```

```r
var.gi <- var(cf_df$Gross_income)
var.gi
```

```
## [1] 137.0966
```

```r
range.gi <- range(cf_df$Gross_income)
range.gi
```

```
## [1]  0.5085 49.6500
```

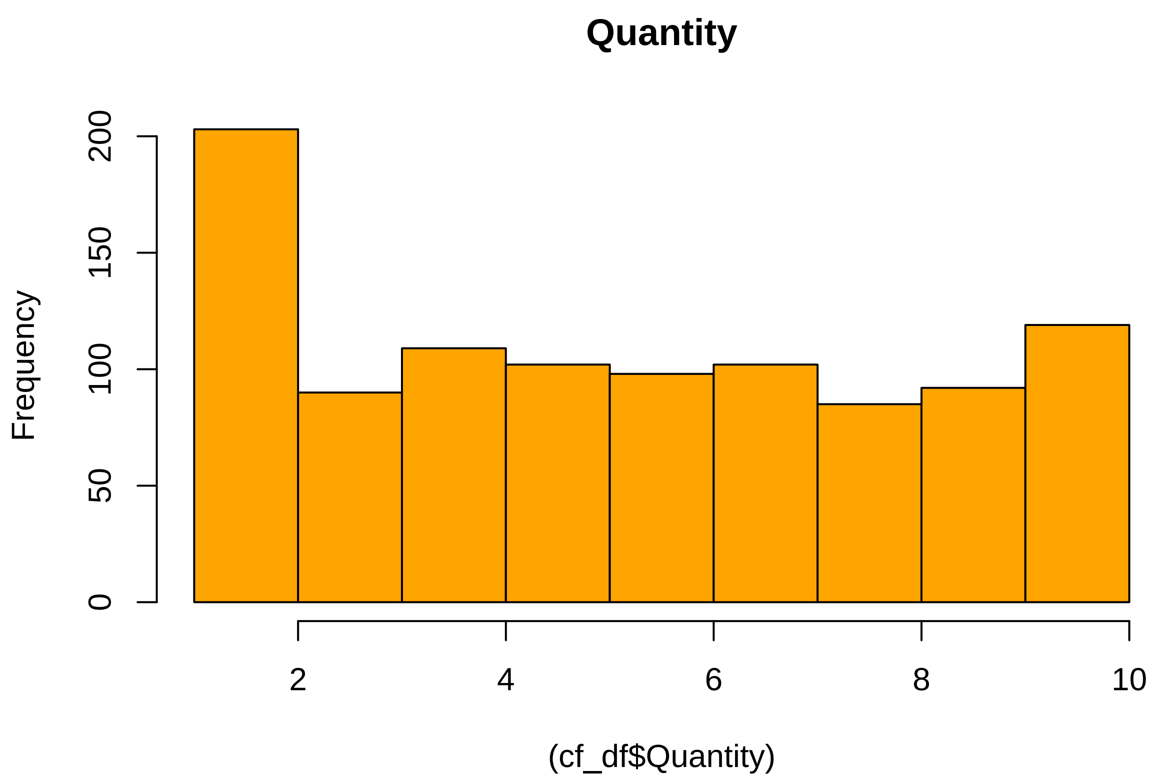
```r
skew.gi <- skewness(cf_df$Gross_income)
skew.gi
```

```
## [1] 0.8898939
```

```r
kurt.gi <- kurtosis(cf_df$Gross_income)
kurt.gi
```

```
## [1] -0.09329206
```

```r
# Quantity


hist((cf_df$`Quantity`), col = 'orange', main = "Quantity")
```




```r
sd.quan <- sd(cf_df$Quantity)
sd.quan
```

```
## [1] 2.923431
```

```r
var.quan <- var(cf_df$Quantity)
var.quan
```

```
## [1] 8.546446
```

```r
range.quan <- range(cf_df$Quantity)
range.quan
```

```
## [1]  1 10
```

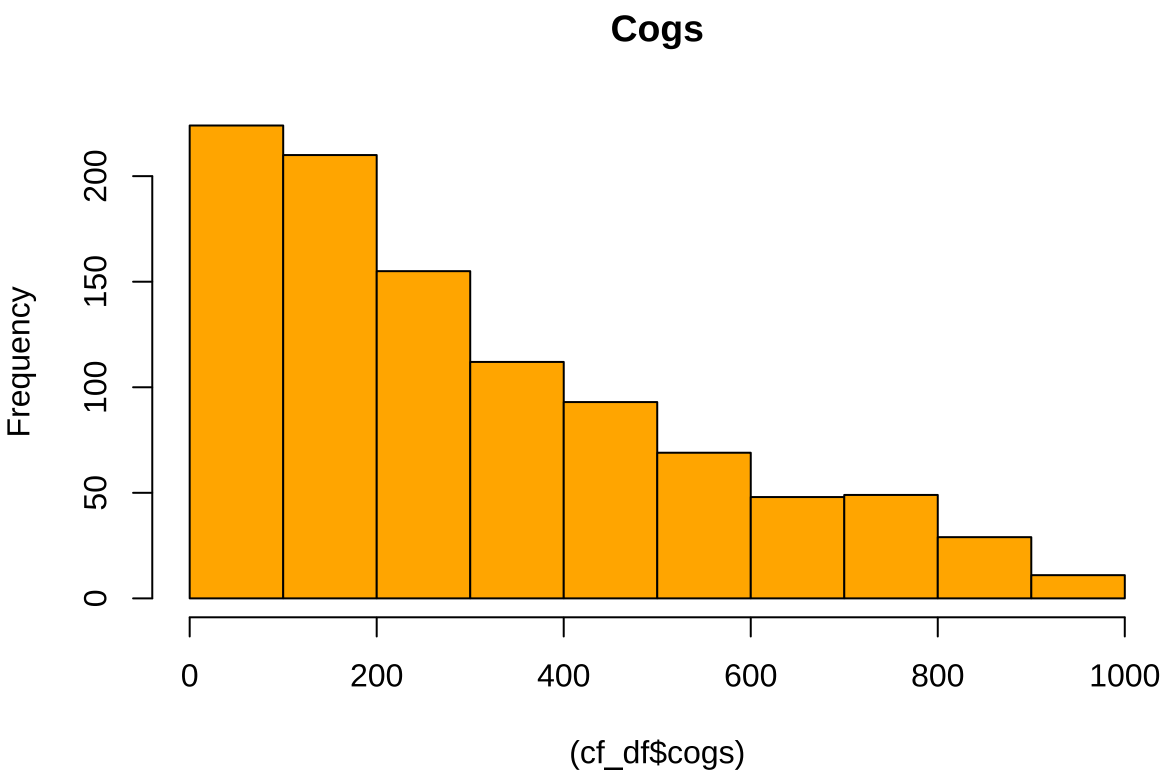
```r
skew.quan <- skewness(cf_df$Quantity)
skew.quan
```

```
## [1] 0.01290225
```

```r
kurt.quan <- kurtosis(cf_df$Quantity)
kurt.quan
```

```
## [1] -1.219039
```

```r
# Cogs

hist((cf_df$`cogs`), col = 'orange', main = "Cogs")
```




```r
sd.cogs <- sd(cf_df$cogs)
sd.cogs
```

```
## [1] 234.1765
```

```r
var.cogs <- var(cf_df$cogs)
var.cogs
```

```
## [1] 54838.64
```

```r
range.cogs <- range(cf_df$cogs)
range.cogs
```

```
## [1]  10.17 993.00
```

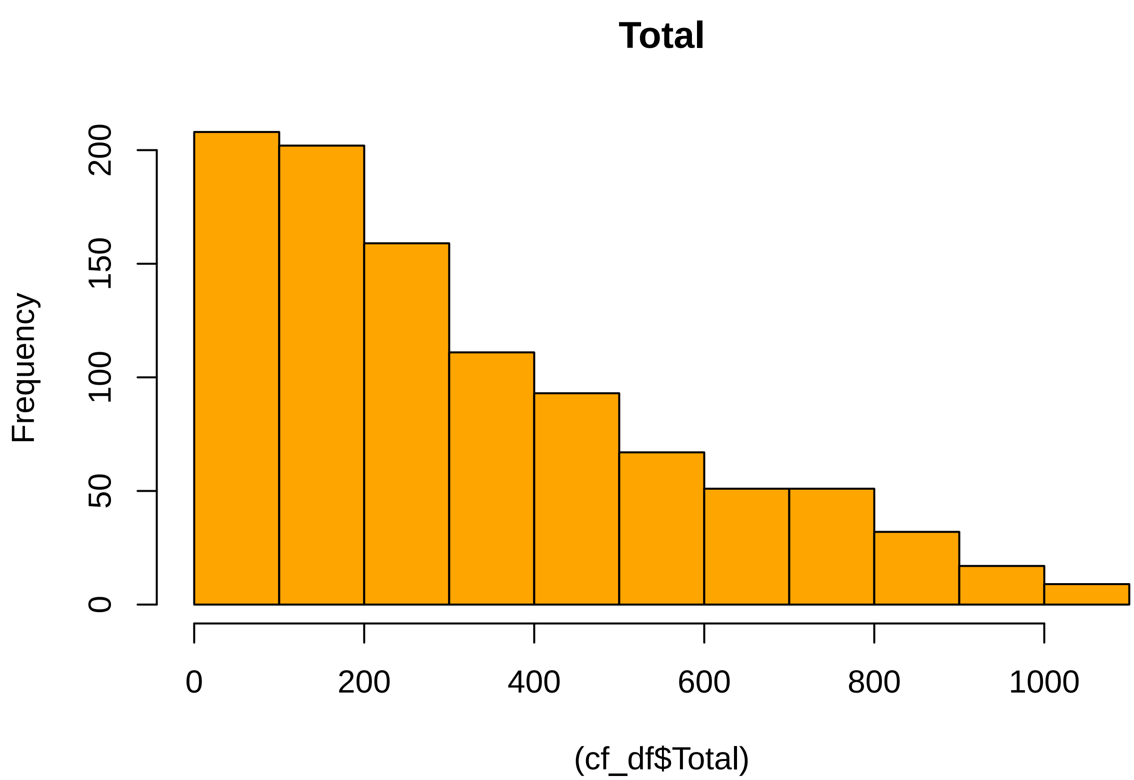
```r
skew.cogs <- skewness(cf_df$cogs)
skew.cogs
```

```
## [1] 0.8898939
```

```r
kurt.cogs <- kurtosis(cf_df$cogs)
kurt.cogs
```

```
## [1] -0.09329206
```

```r
# Total

hist((cf_df$`Total`), col = 'orange', main = "Total")
```




```r
sd.total <- sd(cf_df$Total)
sd.total
```

```
## [1] 245.8853
```

```
var.total <- var(cf_df$Total)
var.total
```

```
## [1] 60459.6
```

```
range.total <- range(cf_df$Total)
range.total
```

```
## [1]   10.6785 1042.6500
```

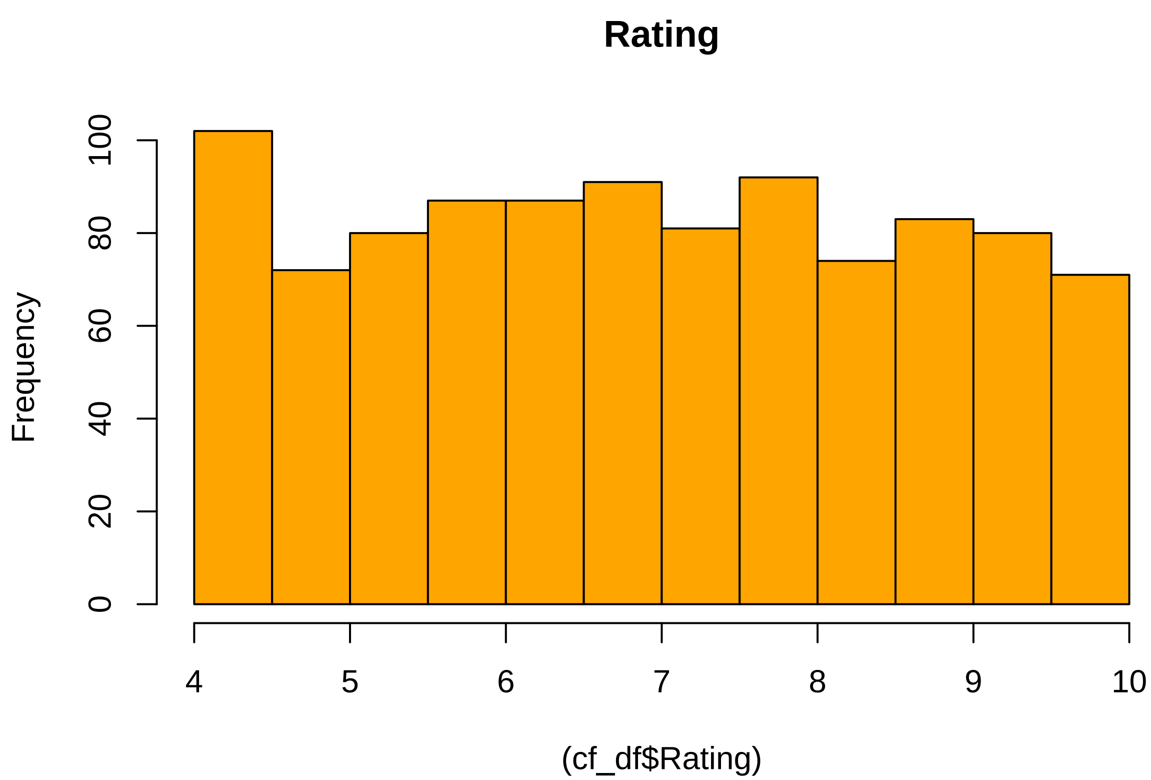
```
skew.total <- skewness(cf_df$Total)
skew.total
```

```
## [1] 0.8898939
```

```
kurt.total <- kurtosis(cf_df$Total)
kurt.total
```

```
## [1] -0.09329206
```

```
# Rating

hist((cf_df$`Rating`), col = 'orange', main = "Rating")
```



```r
sd.r <- sd(cf_df$Rating)
sd.r
```

```
## [1] 1.71858
```

```r
var.r <- var(cf_df$Rating)
var.r
```

```
## [1] 2.953518
```

```r
range.r <- range(cf_df$Rating)
range.r
```

```
## [1]  4 10
```

```r
skew.r <- skewness(cf_df$Rating)
skew.r
```

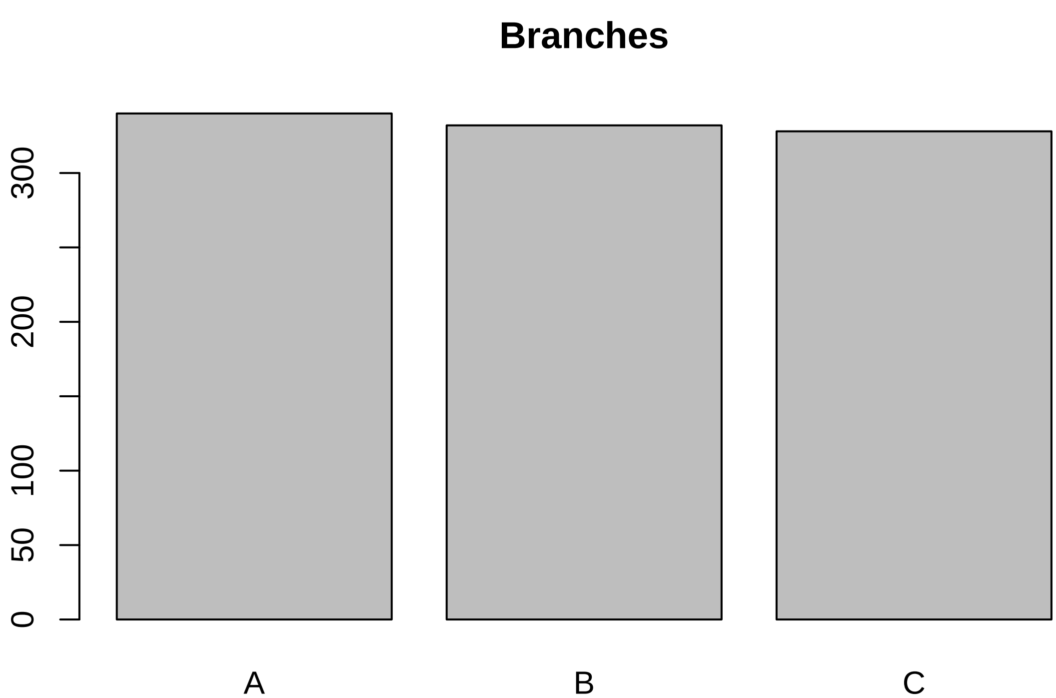
```
## [1] 0.008982638
```

```r
kurt.r <- kurtosis(cf_df$Rating)
kurt.r
```

```
## [1] -1.155525
```

Barplots

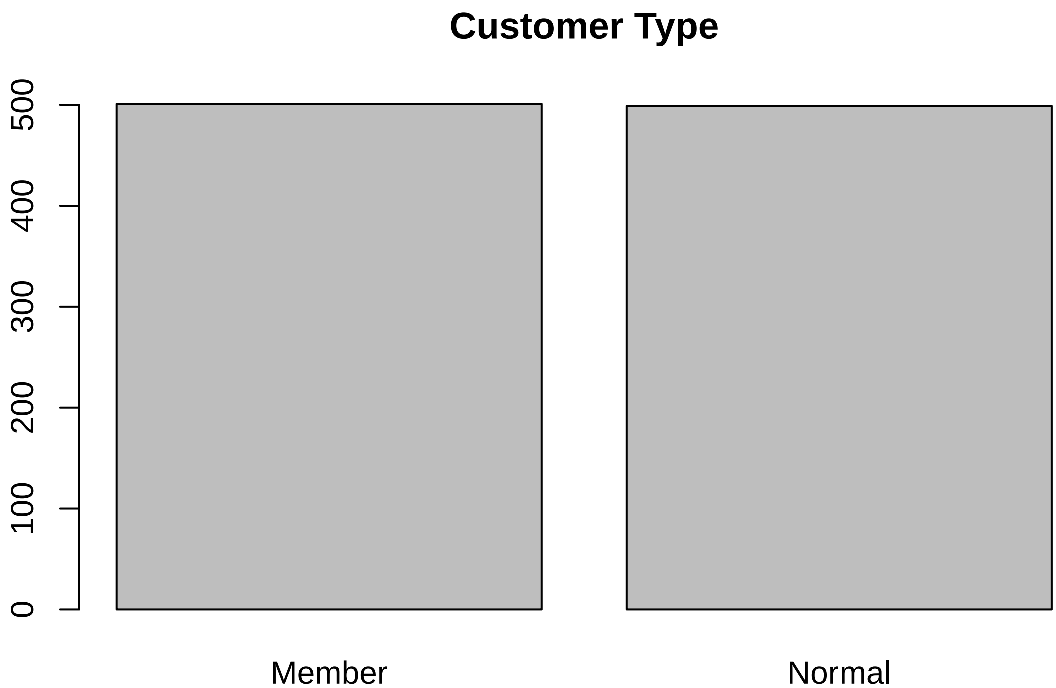
```r
# Branches
barplot(table(cf_df$Branch), main = "Branches")
```



Branch

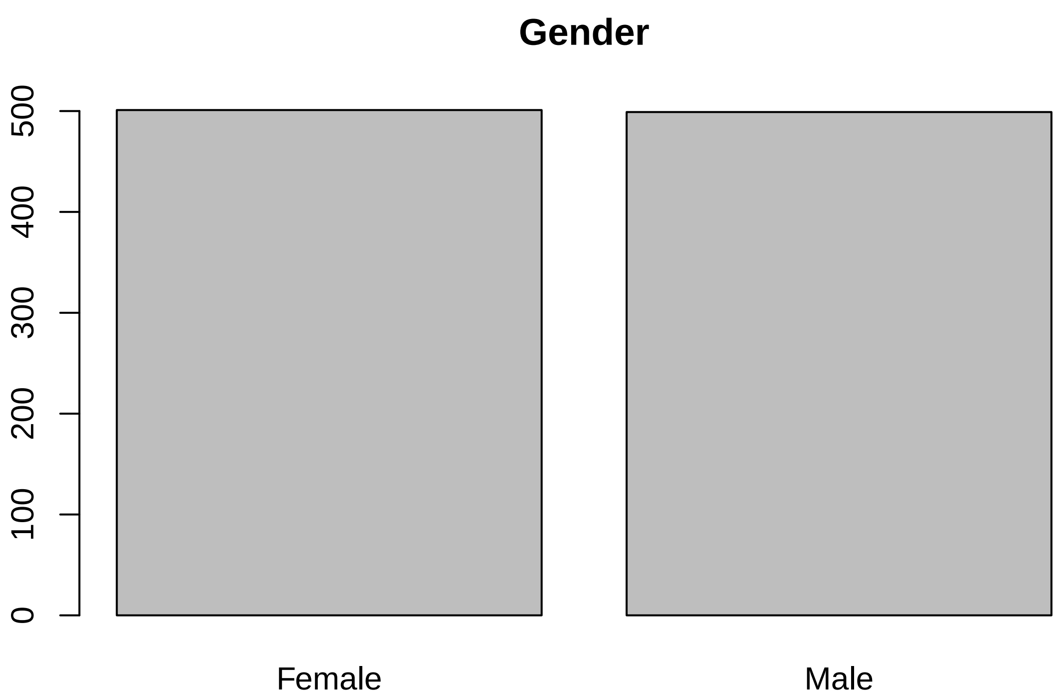
A has the most activity but there is very little difference between them.

```r
# Customer Type
barplot(table(cf_df$Customer_type), main = "Customer Type")
```



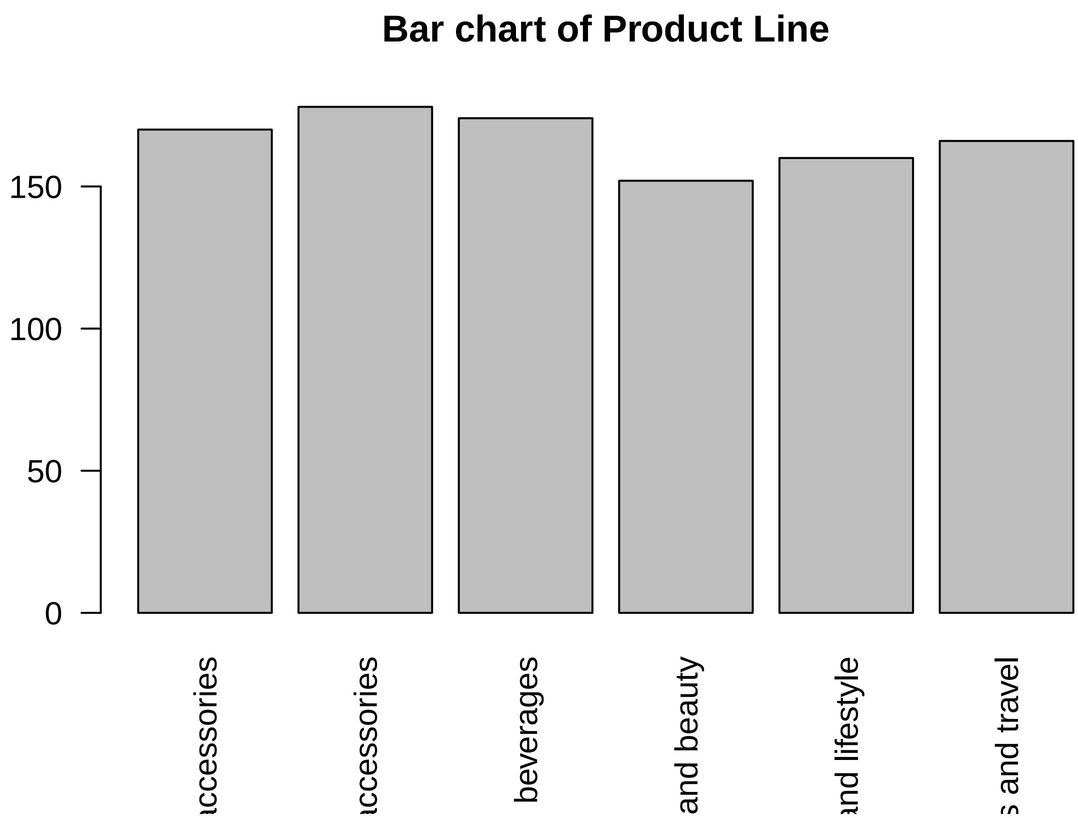
Transactions between the members and non-members are equally matched.

```r
# Gender
barplot(table(cf_df$Gender), main = "Gender")
```



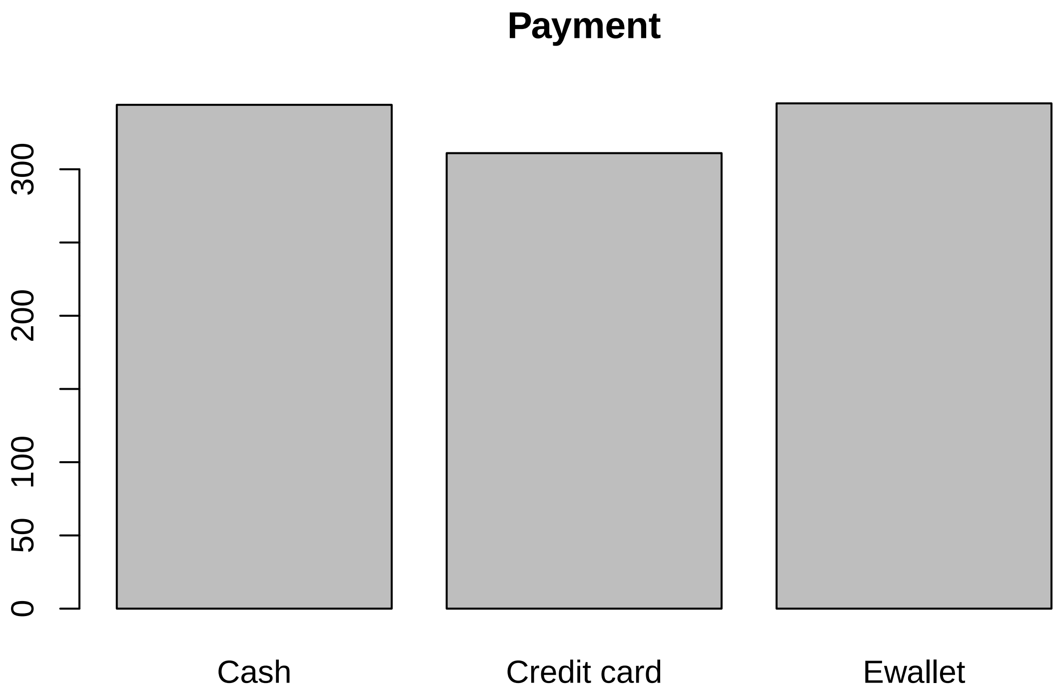
There seems to be very little to no difference in terms of Genders

```r
# Product Line

barplot(table(cf_df$Product_line), main = "Bar chart of Product Line", las=2)
```



It seems Electronics and accessories and food and berverages are the most popular product lines.

```r
# Payment
barplot(table(cf_df$Payment), main = "Payment")
```



Most of the customers transact using their ewallet or using cash.

Bivariate Analysis

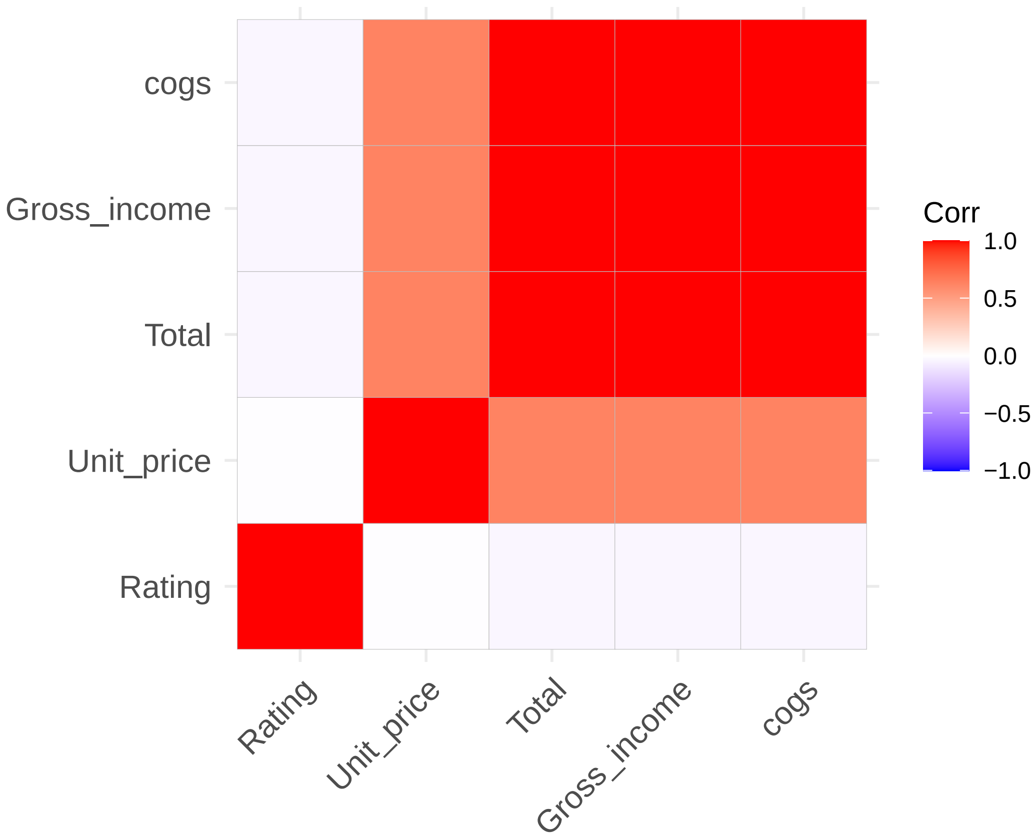
```r
# Creating a new dataframe num with numerical data variables
Unit_price<- cf_df$Unit_price
Gross_income<-cf_df$Gross_income
cogs<-cf_df$cogs
Total<-cf_df$Total
Rating<-cf_df$Rating

num_data <- data.frame(Unit_price, Gross_income, cogs, Total, Rating)
head(num_data)
```

```
##   Unit_price Gross_income   cogs    Total Rating
## 1      74.69      26.1415 522.83 548.9715    9.1
## 2      15.28       3.8200  76.40  80.2200    9.6
## 3      46.33      16.2155 324.31 340.5255    7.4
## 4      58.22      23.2880 465.76 489.0480    8.4
## 5      86.31      30.2085 604.17 634.3785    5.3
## 6      85.39      29.8865 597.73 627.6165    4.1
```

```r
# Correlation
# Correlation is a statistical technique that can show whether and how strongly pairs of variables are

# Calculating the correlation matrix

corr <- cor(num_data)
head(corr)
```

```
##                 Unit_price Gross_income       cogs      Total       Rating
## Unit_price    1.000000000    0.6339621  0.6339621  0.6339621 -0.008777507
## Gross_income  0.633962089    1.0000000  1.0000000  1.0000000 -0.036441705
## cogs          0.633962089    1.0000000  1.0000000  1.0000000 -0.036441705
## Total         0.633962089    1.0000000  1.0000000  1.0000000 -0.036441705
## Rating       -0.008777507   -0.0364417 -0.0364417 -0.0364417  1.000000000
```

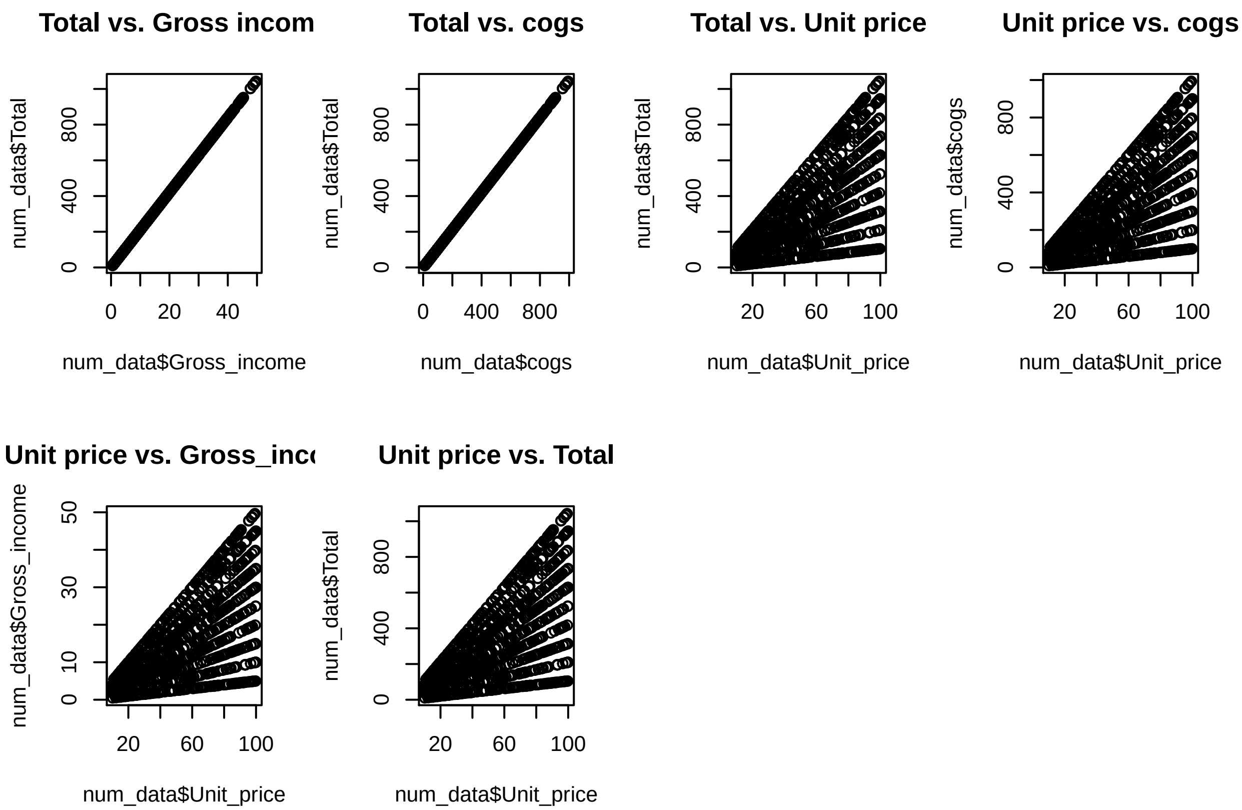
```r
# Plotting the correlation matrix

ggcorrplot(corr,hc.order = TRUE)
```



We observe that most of the variables are perfectly correlated which is problematic in modelling hence the need for feature extraction or feature selection.

Scatterplots

```r
par(mfrow=c(2,4))
plot(num_data$Gross_income,num_data$Total, main="Total vs. Gross income")
plot(num_data$cogs, num_data$Total, main="Total vs. cogs")
plot(num_data$Unit_price, num_data$Total, main="Total vs. Unit price")
plot(num_data$Unit_price,num_data$cogs, main="Unit price vs. cogs")
plot(num_data$Unit_price,num_data$Gross_income, main="Unit price vs. Gross_income")
plot(num_data$Unit_price,num_data$Total, main="Unit price vs. Total")
```



9. Implementing The Solution

Principal Component Analysis

####We'll perform and visualize PCA in the given dataset.

```
# Selecting the numerical data
cf_df_num <- select_if(cf_df,is.numeric)
str(cf_df_num)
```

```
## 'data.frame':    1000 obs. of  6 variables:
##  $ 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 ...
##  $ 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      : 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 ...
```

```
# We then pass the data to the prcomp() and set the center and scale arguments, to be FALSE and TRUE

ef.pca <- prcomp(cf_df_num, center = FALSE, scale. = TRUE)
summary(ef.pca)
```

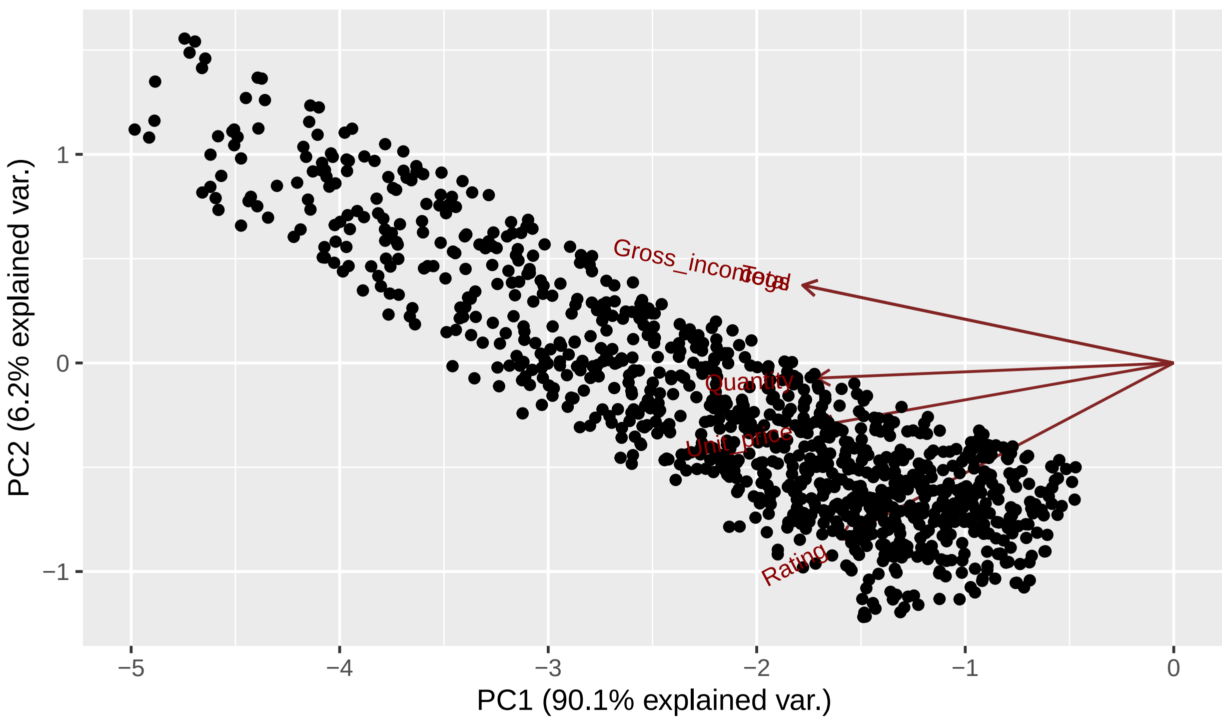
```
## Importance of components:
##                           PC1     PC2     PC3     PC4       PC5       PC6
## Standard deviation     2.3249 0.60849 0.44308 0.16806 3.184e-16 1.283e-16
## Proportion of Variance 0.9009 0.06171 0.03272 0.00471 0.000e+00 0.000e+00
## Cumulative Proportion  0.9009 0.96257 0.99529 1.00000 1.000e+00 1.000e+00
```

```r
# Calling str() to have a look at our PCA object
str(ef.pca)
```

PC1 explains 90% of the total variance, which means that more than three-quarters of the information in the dataset can be encapsulated by just that one Principal Component. PC2 explains 6.1% of the variance, PC3 - 2.2% and PC4- 0.4%

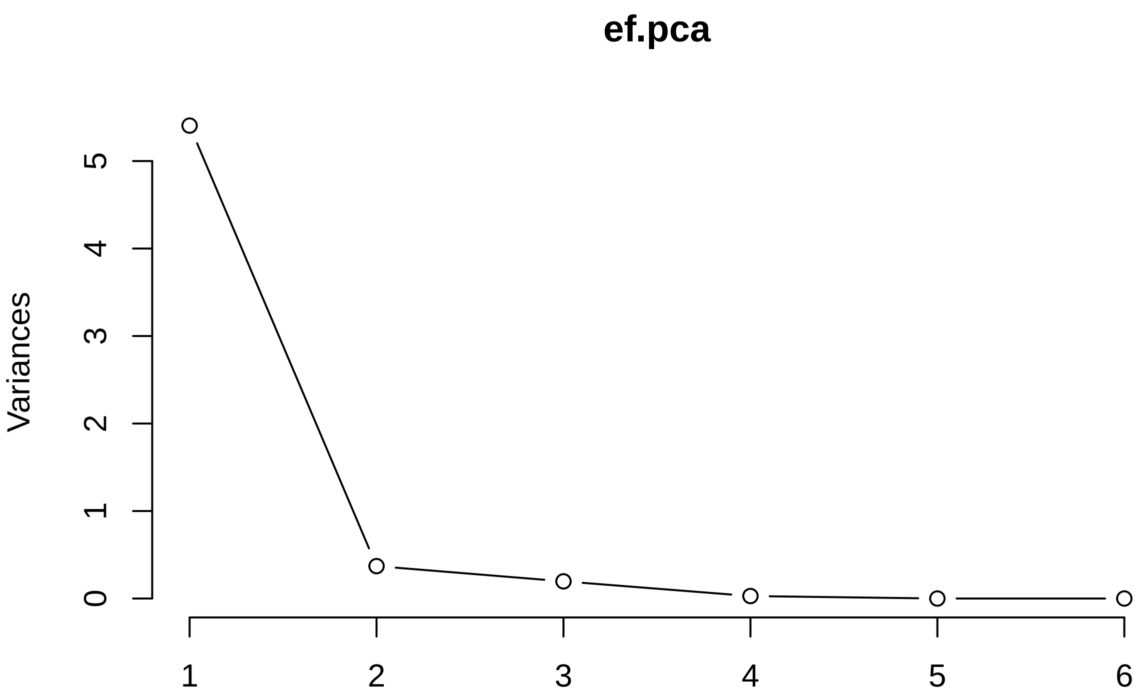
```
## List of 5
##  $ sdev    : num [1:6] 2.32 6.08e-01 4.43e-01 1.68e-01 3.18e-16 ...
##  $ rotation: num [1:6, 1:6] -0.402 -0.405 -0.421 -0.421 -0.379 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:6] "Unit_price" "Quantity" "cogs" "Gross_income" ...
##   .. ..$ : chr [1:6] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : logi FALSE
##  $ scale   : Named num [1:6] 61.68 6.24 386.71 19.34 7.18 ...
##   ..- attr(*, "names")= chr [1:6] "Unit_price" "Quantity" "cogs" "Gross_income" ...
##  $ x       : num [1:1000, 1:6] -3.13 -1.18 -2.2 -2.86 -3.27 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:6] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

```r
# We will now plot our pca.
set.seed(123)
ggbiplot(ef.pca, labels=rownames(ef.pca),ellipse = TRUE,obs.scale=1,var.scale=1)
```



This is not so clear, we'll therefore plot to see the number of components that contribute more to PC1

```r
plot(ef.pca, type="l")
```



6 components contribute more but PC1 contribte the most in this anaysis.

Part 2: Feature Selection

```r
# Calculating the correlation matrix

corrmatrix <- cor(num_data)

# Find attributes that are highly correlated

highcorr <- findCorrelation(corrmatrix, cutoff=0.75)

# Highly correlated attributes

highcorr
```

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

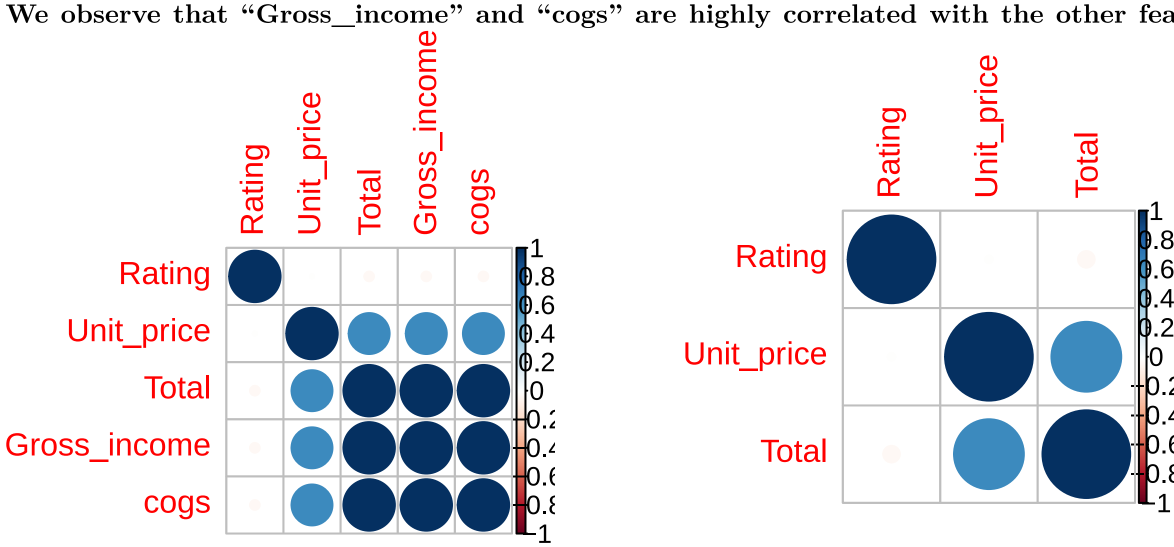
```r
names(num_data[highcorr])
```

```
## [1] "Gross_income" "cogs"
```

```r
# Removing Redundant Features
num_data_clean <- num_data[-highcorr]
```

```r
# Performing our graphical comparison
par(mfrow = c(1, 2))
corrplot(corrmatrix, order = "hclust")
corrplot(cor(num_data_clean), order = "hclust")
```

We observe that "Gross_income" and "cogs" are highly correlated with the other features.



###
Removing highly correlated variables result to less coreelated variables. Hence the selected features are Unit_Price, Total and Rating. There are no more highly correlated variables.

Part 3 : Association Rules

```r
# Installing and reading the necessary packages for the association rules analysis
#install.packages("arules", dependencies = TRUE)
library(arules)
```

This section will require that you create association rules that will allow you to identify relationships between variables in the dataset.

```
## 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:kernlab':
##
##     size
```

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

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

```r
# Reading and previewing the dataset as transcations

sdf <- read.transactions("http://bit.ly/SupermarketDatasetII")
```

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

```r
head(sdf)
```

```
## transactions in sparse format with
##  6 transactions (rows) and
##  5729 items (columns)
```

```r
# Checking the dimensions of the data
dim(sdf)
```

```
## [1] 7501 5729
```

```r
# Displaying the structure of our dataset

str(sdf)
```

The dataset has 7,501 transactions and 5729 columns

```
## Formal class 'transactions' [package "arules"] with 3 slots
##   ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. ..@ i       : int [1:23299] 1087 1614 1705 1732 1993 2101 2105 2358 2444 3463 ...
##   .. .. ..@ p       : int [1:7502] 0 15 16 17 18 24 27 31 33 36 ...
##   .. .. ..@ Dim     : int [1:2] 5729 7501
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : NULL
##   .. .. .. ..$ : NULL
##   .. .. ..@ factors : list()
##   ..@ itemInfo   :'data.frame':  5729 obs. of  1 variable:
##   .. ..$ labels: chr [1:5729] "&" "accessories" "accessories,antioxydant" "accessories,champagne,fre
##   ..@ itemsetInfo:'data.frame':  0 obs. of  0 variables
```

```r
# Verifying the class of the object
class(sdf)
```

```
## [1] "transactions"
## attr(,"package")
## [1] "arules"
```

```r
# Previewing the first few items in the dataset
inspect(sdf[1:5])
```

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

```r
# Previewing the items in the dataset as if it were in a dataframe
items<-as.data.frame(itemLabels(sdf))
colnames(items) <- "Item"
head(items, 10)
```

```
##                                      Item
## 1                                       &
## 2                             accessories
## 3                 accessories,antioxydant
## 4             accessories,champagne,fresh
## 5           accessories,champagne,protein
## 6                   accessories,chocolate
## 7  accessories,chocolate,champagne,frozen
## 8            accessories,chocolate,frozen
## 9               accessories,chocolate,low
## 10      accessories,chocolate,pasta,salt
```

```r
# Getting the summary statistics of the data

summary(sdf)
```

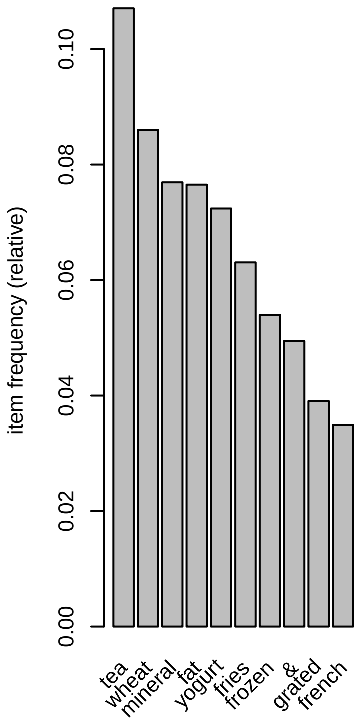
```
## transactions as itemMatrix in sparse format with
##  7501 rows (elements/itemsets/transactions) and
```

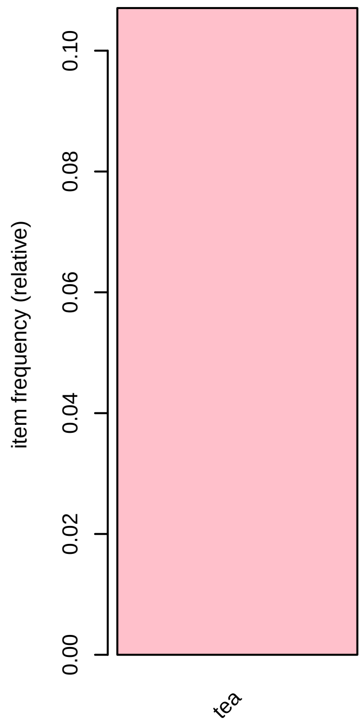
```
##  5729 columns (items) and a density of 0.0005421748
##
## most frequent items:
##     tea   wheat mineral     fat  yogurt (Other)
##     803     645     577     574     543   20157
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   15   16
## 1603 2007 1382  942  651  407  228  151   70   39   13    5    1    1    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.106   4.000  16.000
##
## includes extended item information - examples:
##                    labels
## 1                       &
## 2             accessories
## 3 accessories,antioxydant
```

```r
# Exploring the frequency of some articles

itemFrequency(sdf[, 7:11],type = "absolute")
```

```
## accessories,chocolate,champagne,frozen           accessories,chocolate,frozen
##                                      1                                      1
##              accessories,chocolate,low       accessories,chocolate,pasta,salt
##                                      1                                      1
##       accessories,chocolate,salt,green
##                                      1
```

```r
# Producing a chart of frequencies and filtering to consider only items with a minimum percentage of su
par(mfrow = c(1, 3))
# plot the frequency of items
itemFrequencyPlot(sdf, topN = 10,col="grey")
itemFrequencyPlot(sdf, support = 0.1,col="pink")
```





The top 10 most common items in the transactions dataset are Mineral water, eggs, spaghetti, french fries, chocolate, green tea, milk, ground beef, frozen vegetables and pancakes.

```
# Setting the parameters for our association analysis
rules <- apriori (sdf, 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 ...[5729 item(s), 7501 transaction(s)] done [0.01s].
## sorting and recoding items ... [354 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [271 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules
```

```
## set of 271 rules
```

```r
# Building a apriori model with Min Support as 0.002 and confidence as 0.8.

rules2 <- apriori (sdf,parameter = list(supp = 0.002, conf = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5   0.002      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: 15
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[5729 item(s), 7501 transaction(s)] done [0.01s].
## sorting and recoding items ... [189 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [99 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
rules2
```

```
## set of 99 rules
```

```r
# Building apriori model with Min Support as 0.002 and confidence as 0.6.

rules3 <- apriori (sdf, parameter = list(supp = 0.001, conf = 0.6))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.6    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 ...[5729 item(s), 7501 transaction(s)] done [0.02s].
## sorting and recoding items ... [354 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
```

```
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [319 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
rules3
```

```
## set of 319 rules
```

```r
#Performing an exploration of our model using the summary function
summary(rules)
```

```
## set of 271 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4
## 107 144  20
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.679   3.000   4.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001067   Min.   :0.800   Min.   :0.001067   Min.   :  7.611
##  1st Qu.:0.001200   1st Qu.:0.931   1st Qu.:0.001200   1st Qu.: 11.630
##  Median :0.001600   Median :1.000   Median :0.001600   Median : 13.068
##  Mean   :0.002834   Mean   :0.963   Mean   :0.002973   Mean   : 22.372
##  3rd Qu.:0.002666   3rd Qu.:1.000   3rd Qu.:0.002800   3rd Qu.: 20.218
##  Max.   :0.068391   Max.   :1.000   Max.   :0.076523   Max.   :613.718
##      count
##  Min.   :  8.00
##  1st Qu.:  9.00
##  Median : 12.00
##  Mean   : 21.26
##  3rd Qu.: 20.00
##  Max.   :513.00
##
## mining info:
##  data ntransactions support confidence
##   sdf          7501   0.001        0.8
```

```r
# Observing rules built in our model i.e. first 10 model rules

inspect(rules[1:10])
```

```
##      lhs                               rhs      support     confidence
## [1]  {cookies,low}                  => {yogurt} 0.001066524 1.0
## [2]  {cookies,low}                  => {fat}    0.001066524 1.0
## [3]  {extra}                        => {dark}   0.001066524 1.0
## [4]  {burgers,whole}                => {wheat}  0.001199840 1.0
## [5]  {fries,escalope,pasta,mushroom} => {cream}  0.001066524 1.0
## [6]  {fries,cookies,green}          => {tea}    0.001333156 1.0
## [7]  {shrimp,whole}                 => {wheat}  0.001066524 1.0
```

```
## [8]  {rice,cake}                      => {wheat}  0.001333156 1.0
## [9]  {tomatoes,whole}                  => {wheat}  0.001066524 0.8
## [10] {rice,chocolate}                  => {wheat}  0.001199840 0.9
##      coverage    lift      count
## [1]  0.001066524 13.813996  8
## [2]  0.001066524 13.067944  8
## [3]  0.001066524 83.344444  8
## [4]  0.001199840 11.629457  9
## [5]  0.001066524 47.777070  8
## [6]  0.001333156  9.341220 10
## [7]  0.001066524 11.629457  8
## [8]  0.001333156 11.629457 10
## [9]  0.001333156  9.303566  8
## [10] 0.001333156 10.466512  9
```

```r
# Ordering these rules by a criteria such as the level of confidence then looking at the first five rul
rules<-sort(rules, by="confidence", decreasing=TRUE)
inspect(rules[1:5])
```

```
##     lhs                                rhs      support     confidence
## [1] {cookies,low}                   => {yogurt} 0.001066524 1
## [2] {cookies,low}                   => {fat}    0.001066524 1
## [3] {extra}                         => {dark}   0.001066524 1
## [4] {burgers,whole}                 => {wheat}  0.001199840 1
## [5] {fries,escalope,pasta,mushroom} => {cream}  0.001066524 1
##     coverage    lift     count
## [1] 0.001066524 13.81400 8
## [2] 0.001066524 13.06794 8
## [3] 0.001066524 83.34444 8
## [4] 0.001199840 11.62946 9
## [5] 0.001066524 47.77707 8
```

Part 4 : Anomaly Detection

```r
# Reading and previewing the dataset

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

This is to detect whether there are any anomalies in the given sales dataset.

```
##        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
```

```r
# Checking the dimensions of our dataset

dim(ano)
```

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

```r
# Checking the structure of our dataset

str(ano)
```

```
## 'data.frame':    1000 obs. of  2 variables:
##  $ Date : chr  "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ Sales: num  549 80.2 340.5 489 634.4 ...
```

```r
# Checking the Statistical summary

summary(ano)
```

```
##      Date               Sales
##  Length:1000        Min.   :  10.68
##  Class :character   1st Qu.: 124.42
##  Mode  :character   Median : 253.85
##                     Mean   : 322.97
##                     3rd Qu.: 471.35
##                     Max.   :1042.65
```

```r
# Checking for null values

colSums(is.na(ano))
```

```
##  Date Sales
##     0     0
```

```r
# Changing the date column from character to Date

ano <- transform(ano, Date = format(as.Date(Date, '%m/%d/%Y'), '%Y/%m/%d'))
ano <- transform(ano, Date = as.Date(Date))
sapply(ano, class)
```

```
##      Date     Sales
##    "Date" "numeric"
```

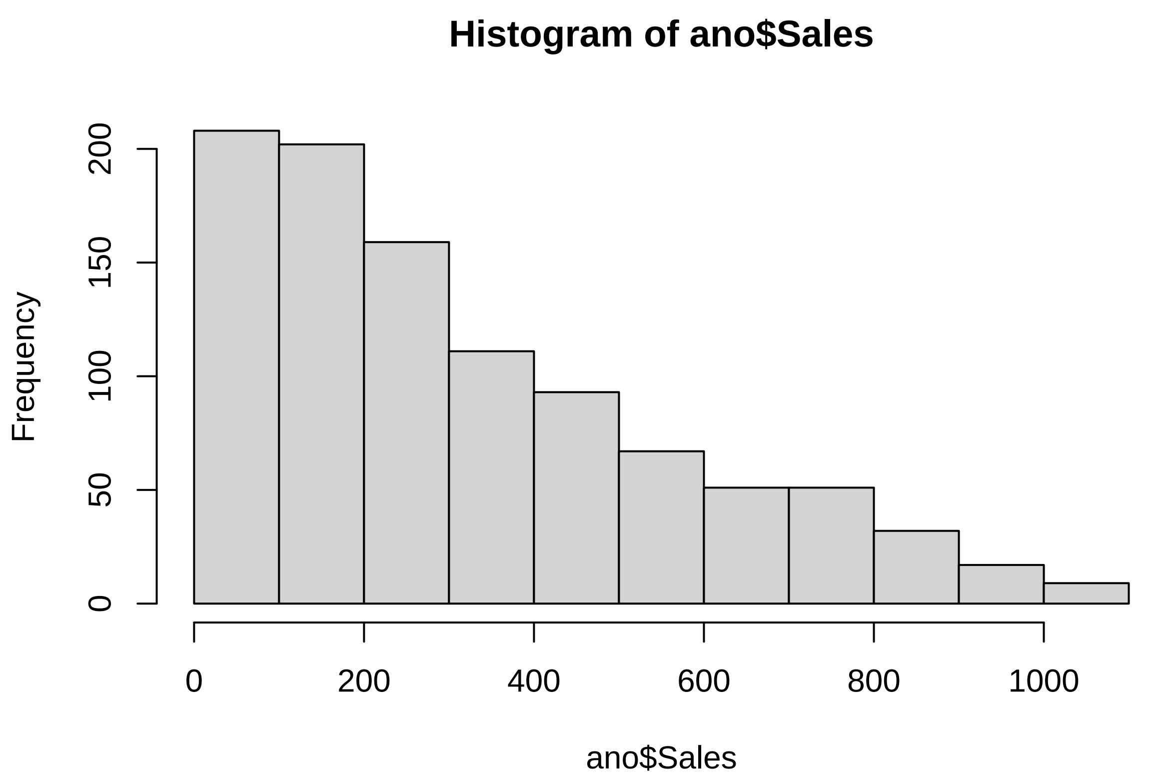
```r
# Grouping the dataset by the Date column

Sales <- ano$Sales
Date <- ano$Date
ano = ano %>% arrange(Date)
head(ano)
```

```
##          Date   Sales
## 1 2019-01-01 457.443
## 2 2019-01-01 399.756
## 3 2019-01-01 470.673
## 4 2019-01-01 388.290
## 5 2019-01-01 132.762
## 6 2019-01-01 132.027
```

```r
# Sales Distribution
# sort dates in ascending order
ano_sales <- ano[order(ano$Date),]
head(ano_sales, 5)
```

```
##          Date   Sales
## 1 2019-01-01 457.443
## 2 2019-01-01 399.756
## 3 2019-01-01 470.673
## 4 2019-01-01 388.290
## 5 2019-01-01 132.762
```

```r
## Plotting Histogram to show sales distribution

hist(ano$Sales)
```




```r
# Transactions Count
# We'll group the transactions per day then tally them

ano_count <- ano %>% group_by(Date) %>% tally()
```

```r
colnames(ano_count) <- c('Date', 'Count')
head(ano_count)
```

```
## # A tibble: 6 x 2
##   Date       Count
##   <date>     <int>
## 1 2019-01-01    12
## 2 2019-01-02     8
## 3 2019-01-03     8
## 4 2019-01-04     6
## 5 2019-01-05    12
## 6 2019-01-06     9
```

```r
## Visualizing Transaction count .
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 </>
```

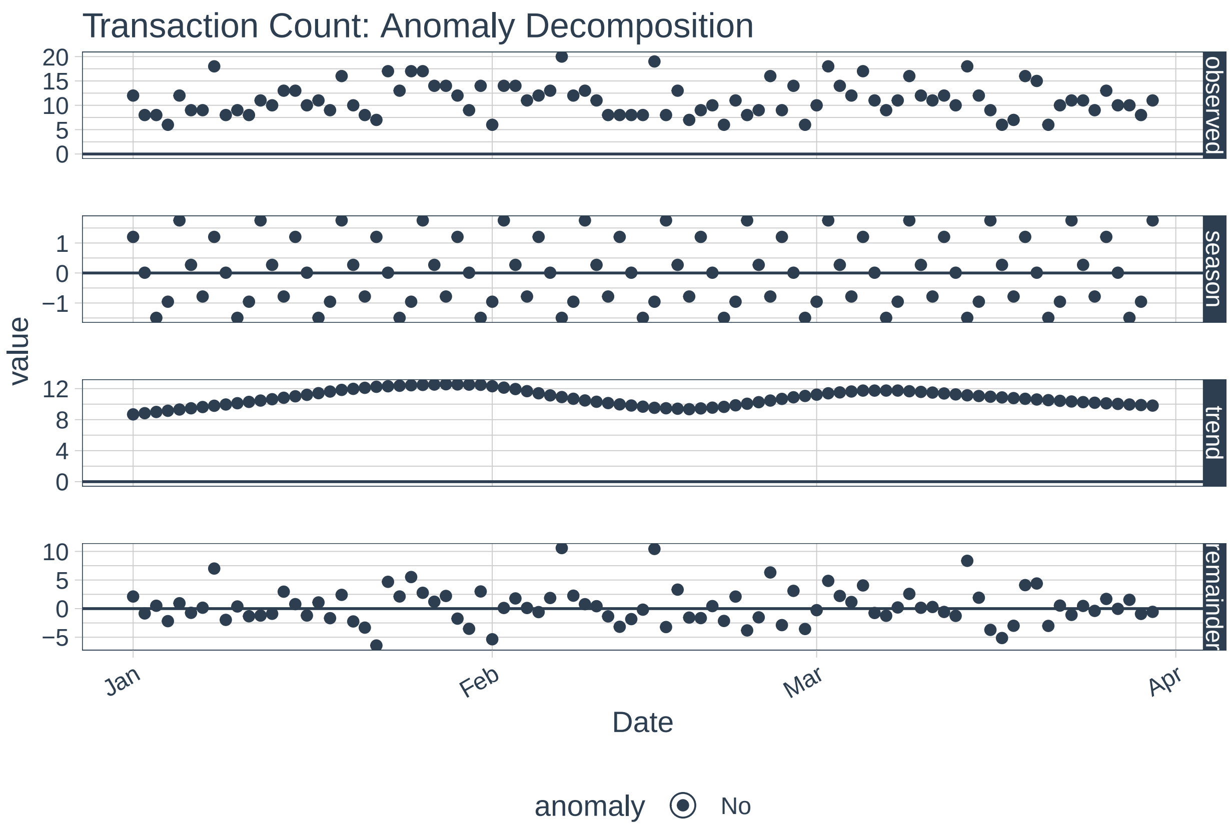
```r
ano_count %>%
    time_decompose(Count) %>%
    anomalize(remainder) %>%
    plot_anomaly_decomposition() +
    ggtitle("Transaction Count: Anomaly Decomposition")
```

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

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

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

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

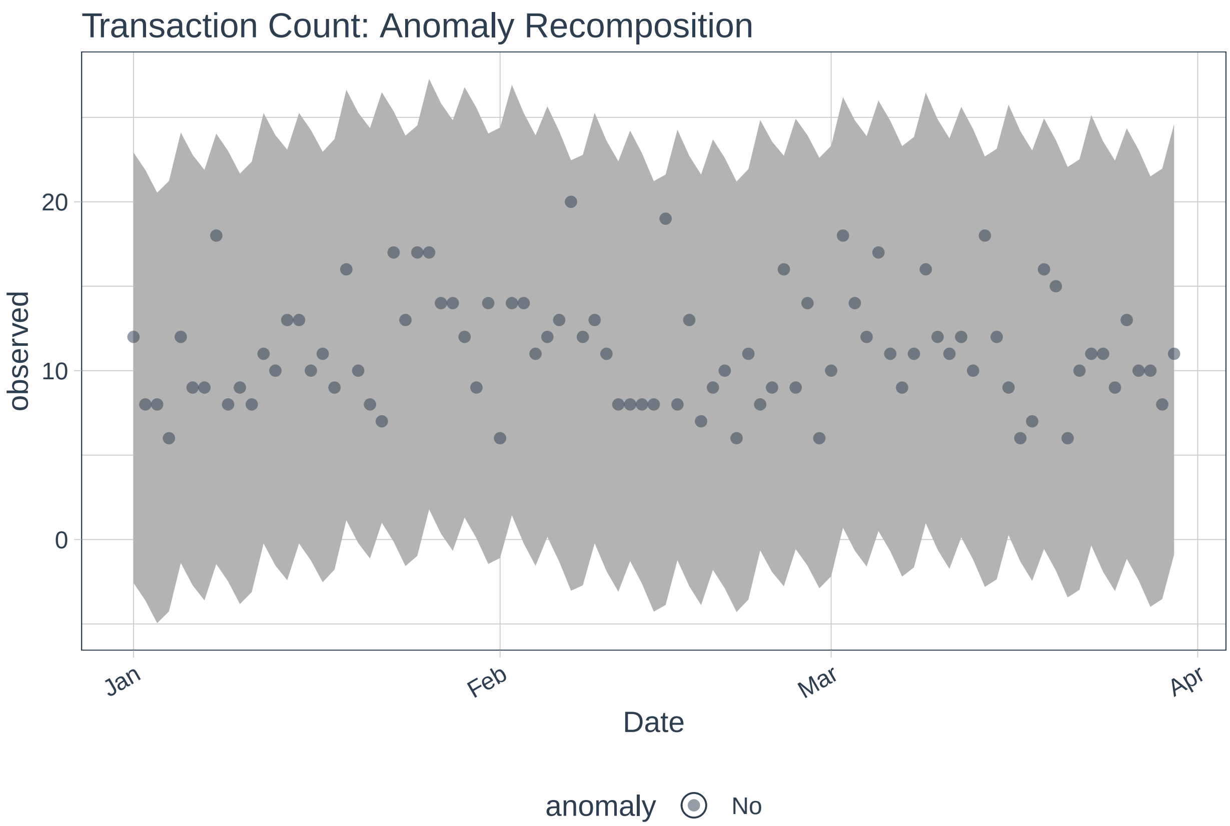



```r
# Visualizing Transaction count.
ano_count %>%
    time_decompose(Count) %>%
    anomalize(remainder) %>%
    time_recompose() %>%
    plot_anomalies(time_recomposed = TRUE, ncol = 3, alpha_dots = 0.5) +
    ggtitle("Transaction Count: Anomaly Recomposition")
```

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

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

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



There were no anomalies that were detected in the number of transactions done per day between January and April.