Part 1 : Dimensionality Reduction

Cherono K

9/19/2020

WEEK 14 IP

1. Defining the Question

In this week's project I'll be working as a Data Analyst at Carrefour Kenya and I'm 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). The project has been divided into four parts where I'll explore a recent marketing dataset by performing various unsupervised learning techniques and later providing recommendations based on your insights.

2. Defining the Metric for Success

Our metrics for success would be:

- Part 1 : Dimensionality Reduction

Performing PCA to perform dimensionality reduction on the dataset by extracting a new set of variables called principal components from an existing large set of variables.

- Part 2 : Feature Selection

Performing analysis and providing insights on the features that contribute the most information to the dataset.

- Part 3 : Association Rules

Identifying relationships between variables in the dataset and providing insights for the analysis.

- Part 4 : Anomaly Detection Check whether there are any anomalies in the given sales dataset. The objective of this task being fraud detection.

3. Understanding the context

Carrefour is a French multinational corporation specialized in retail. As of October 2016 in Kenya, East Africa's largest economy, Carrefour opened its first outlet at the Two Rivers Mall. It is the largest mall in Sub-Sahara Africa with Carrefour as its anchor tenant. The Hub – Karen, a newly opened shopping mall in the Nairobi suburb of Karen also hosts a Carrefour outlet that opened its doors in May 2016. As of today, Carrefour is a major supermarket in Kenya occupying most spaces previously occupied by Nakumatt. It sales a wide variety of products and is known for the cheap prices compared to other retail stores.

4. Recording the Experimental Design

The following are the steps taken to implement the solution :

- Define the question, the metric for success, the context, experimental design taken.
- Read and explore the given datasets.
- Define the appropriateness of the available data to answer the given question.
- Find and deal with outliers and missing data within the dataset.
- Perform Exploratory Data Analysis recording our observations.
- Implementin gthe solution by performing Dimensionality Reduction using PCA, Feature Selection, Association Rules and Anomaly Detection.
- Provide a conclusion and recommendation from the analysis.

5. Data Relevance

Our data is very relevant to our research question. It contains data collected from the Carrefour supermarket which is relevant in implementing the solution.

6. Implementing the Solution

Part 1: Dimensionality Reduction

This section of the project entails reducing our dataset to a low dimensional dataset using the PCA algorithm. We will perform our analysis and provide insights gained from it.

```
#Reading the data as a csv file
data <- read.csv("http://bit.ly/CarreFourDataset")
```

a.) Reading the Data

```
#Checking the first 6 records of the data
head(data)
```

b.) Previewing the Data

```
##    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 last 6 records of the data
tail(data)
```

```
##        Invoice.ID Branch Customer.type Gender           Product.line Unit.price
## 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
## 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
## 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 shape of our data
dim(data)
```

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

```r
#Checking the structure of our data
str(data)
```

```
## '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 ...
```

```
#Checking the class of our data
class(data)
```

```
## [1] "data.frame"
```

```
# Checking the datatypes for each column
columns = colnames(data)
for (column in seq(length(colnames(data)))){
print(columns[column])
print(class(data[, column]))
cat('\n')
}
```

```
## [1] "Invoice.ID"
## [1] "character"
##
## [1] "Branch"
## [1] "character"
##
## [1] "Customer.type"
## [1] "character"
##
## [1] "Gender"
## [1] "character"
##
## [1] "Product.line"
## [1] "character"
##
## [1] "Unit.price"
## [1] "numeric"
##
## [1] "Quantity"
## [1] "integer"
##
## [1] "Tax"
## [1] "numeric"
##
## [1] "Date"
## [1] "character"
```

```
##
## [1] "Time"
## [1] "character"
##
## [1] "Payment"
## [1] "character"
##
## [1] "cogs"
## [1] "numeric"
##
## [1] "gross.margin.percentage"
## [1] "numeric"
##
## [1] "gross.income"
## [1] "numeric"
##
## [1] "Rating"
## [1] "numeric"
##
## [1] "Total"
## [1] "numeric"
```

```r
#checking for unique values in the variables
lapply(data, 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
```

Our data frame has 1000 rows and 16 colmns. Most of he columns hav ethe wrong data types. The Branch, Customer type, Gender, Product Line, Payment and Quantity columns are categorized as character although they are categorical hence should be factor data types. We'll correct this later on.

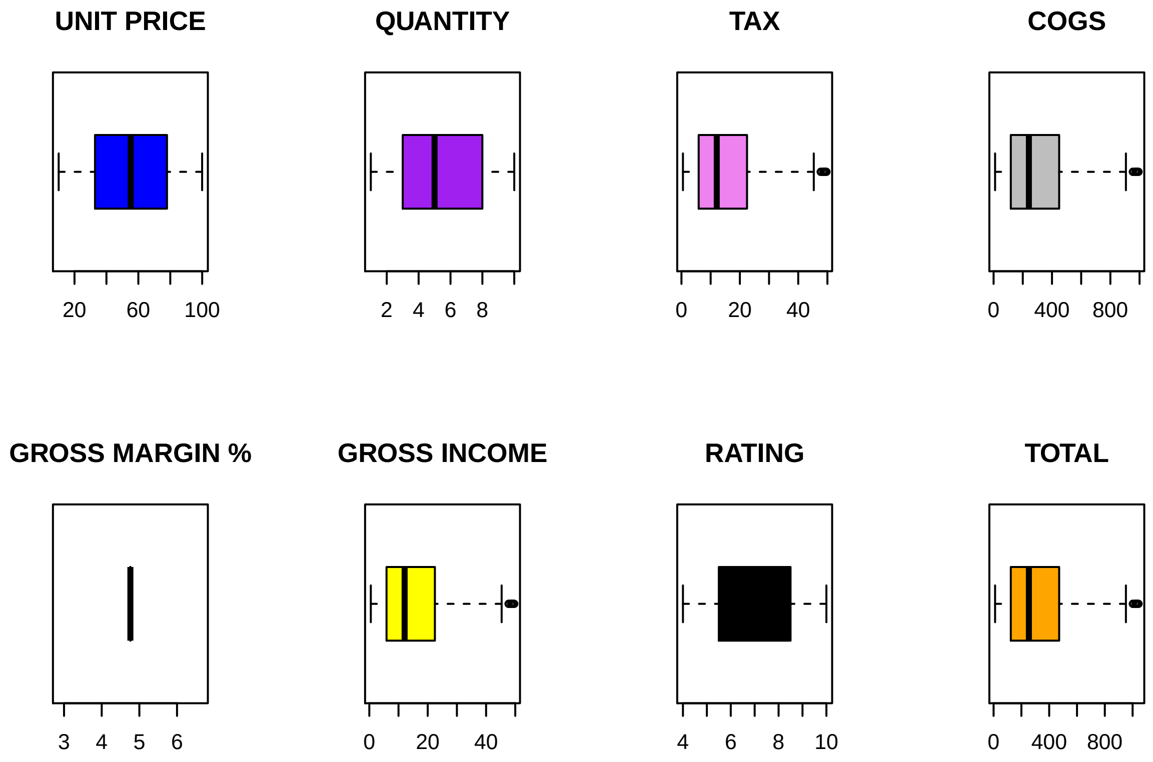
```r
#Checking the number of missing data in our dataset
sum(is.na(data))
```

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

```r
#Checking for duplicated data
duplicated <- data[duplicated(data),]
```

Our data frame has no null values and no duplicated entries.

```r
#Checking for outliers in the numeric columns
par(mfrow=c(2,4))
boxplot(data$Unit.price, horizontal = TRUE, main = toupper("Unit Price"), col = "blue")
boxplot(data$Quantity,horizontal = TRUE, main = toupper("Quantity"), col = "purple")
boxplot(data$Tax, horizontal = TRUE, main = toupper("Tax"), col = "violet")
boxplot(data$cogs,horizontal = TRUE, main = toupper("Cogs"),col = "gray")
boxplot(data$gross.margin.percentage, horizontal = TRUE, main = toupper("Gross Margin %"), col = "green
boxplot(data$gross.income,horizontal = TRUE, main = toupper("Gross Income"), col = "yellow")
boxplot(data$Rating,horizontal = TRUE, main = toupper("Rating"), col = "black")
boxplot(data$Total,horizontal = TRUE, main = toupper("Total"), col = "orange")
```



Outliers are present in our dataset i.e. in the Tax, Cogs, Gross Income and Total column. However, we'll not remove these outliers as they are a huge part of the dataset and they represent different transaction of different customers hence would result to lose of data upon removing them.

c.) Cleaning the data We'll not do much data cleaning here as our data did not have null values, no duplicates and we'll not deal with outliers.

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

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

```r
#Changing the date column from character to date
data$Date <- as.Date(data$Date, "%m/%d/%Y")

#Separating the Year, month and day into different column.
data <- separate(data, "Date", c("Year", "Month", "Day"), sep = "-")
```

```r
#Separating the hour, minutes and seconds into different columns
data <- separate(data, "Time", c("Hour", "Minutes"), sep = ":")
head(data)
```

```
##    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 Year Month Day Hour Minutes     Payment   cogs
## 1        7 26.1415 2019    01  05   13      08     Ewallet 522.83
## 2        5  3.8200 2019    03  08   10      29        Cash  76.40
## 3        7 16.2155 2019    03  03   13      23 Credit card 324.31
## 4        8 23.2880 2019    01  27   20      33     Ewallet 465.76
## 5        7 30.2085 2019    02  08   10      37     Ewallet 604.17
## 6        7 29.8865 2019    03  25   18      30     Ewallet 597.73
##   gross.margin.percentage gross.income Rating    Total
## 1                4.761905      26.1415    9.1 548.9715
## 2                4.761905       3.8200    9.6  80.2200
## 3                4.761905      16.2155    7.4 340.5255
## 4                4.761905      23.2880    8.4 489.0480
## 5                4.761905      30.2085    5.3 634.3785
## 6                4.761905      29.8865    4.1 627.6165
```

```r
#Changing the data type for some columns to factor as they are categorical
cols <- c('Branch' ,'Customer.type', 'Gender','Product.line','Payment')
data[,cols] <- lapply(data[,cols] , factor)

#Changing some character data types to numeric
data$Quantity <- as.numeric(as.character(data$Quantity))
data$Year <- as.numeric(as.character(data$Year))
data$Month <- as.numeric(as.character(data$Month))
data$Day <- as.numeric(as.character(data$Day))
data$Hour <- as.numeric(as.character(data$Hour))
data$Minutes <- as.numeric(as.character(data$Minutes))
```

```r
# Checking the datatypes for each column to see if the changes have been made
columns = colnames(data)
for (column in seq(length(colnames(data)))){
print(columns[column])
print(class(data[, column]))
cat('\n')
}
```

```
## [1] "Invoice.ID"
## [1] "character"
##
## [1] "Branch"
## [1] "factor"
##
```

```
## [1] "Customer.type"
## [1] "factor"
##
## [1] "Gender"
## [1] "factor"
##
## [1] "Product.line"
## [1] "factor"
##
## [1] "Unit.price"
## [1] "numeric"
##
## [1] "Quantity"
## [1] "numeric"
##
## [1] "Tax"
## [1] "numeric"
##
## [1] "Year"
## [1] "numeric"
##
## [1] "Month"
## [1] "numeric"
##
## [1] "Day"
## [1] "numeric"
##
## [1] "Hour"
## [1] "numeric"
##
## [1] "Minutes"
## [1] "numeric"
##
## [1] "Payment"
## [1] "factor"
##
## [1] "cogs"
## [1] "numeric"
##
## [1] "gross.margin.percentage"
## [1] "numeric"
##
## [1] "gross.income"
## [1] "numeric"
##
## [1] "Rating"
## [1] "numeric"
##
## [1] "Total"
## [1] "numeric"
```

```
#Dropping the gross margin percentage and year columns as they are constants throughout the dataset als
drop <- c("Invoice.ID", "gross.margin.percentage","Year")
data = data[,!(names(data) %in% drop)]
```

```r
head(data)
```

```
##   Branch Customer.type Gender           Product.line Unit.price Quantity
## 1      A        Member Female      Health and beauty      74.69        7
## 2      C        Normal Female Electronic accessories      15.28        5
## 3      A        Normal   Male     Home and lifestyle      46.33        7
## 4      A        Member   Male      Health and beauty      58.22        8
## 5      A        Normal   Male      Sports and travel      86.31        7
## 6      C        Normal   Male Electronic accessories      85.39        7
##       Tax Month Day Hour Minutes     Payment   cogs gross.income Rating
## 1 26.1415     1   5   13       8     Ewallet 522.83      26.1415    9.1
## 2  3.8200     3   8   10      29        Cash  76.40       3.8200    9.6
## 3 16.2155     3   3   13      23 Credit card 324.31      16.2155    7.4
## 4 23.2880     1  27   20      33     Ewallet 465.76      23.2880    8.4
## 5 30.2085     2   8   10      37     Ewallet 604.17      30.2085    5.3
## 6 29.8865     3  25   18      30     Ewallet 597.73      29.8865    4.1
##      Total
## 1 548.9715
## 2  80.2200
## 3 340.5255
## 4 489.0480
## 5 634.3785
## 6 627.6165
```

d.) Exploratory Data Analysis We'll perform EDA just to understand how variables are distributed in our data frame as well us to understand the relationship between different variables in our data frame.

```r
#We'll use the summary function to check some measures of disersion and central tendency of our dataset
summary(data)
```

i.) Univariate Analysis

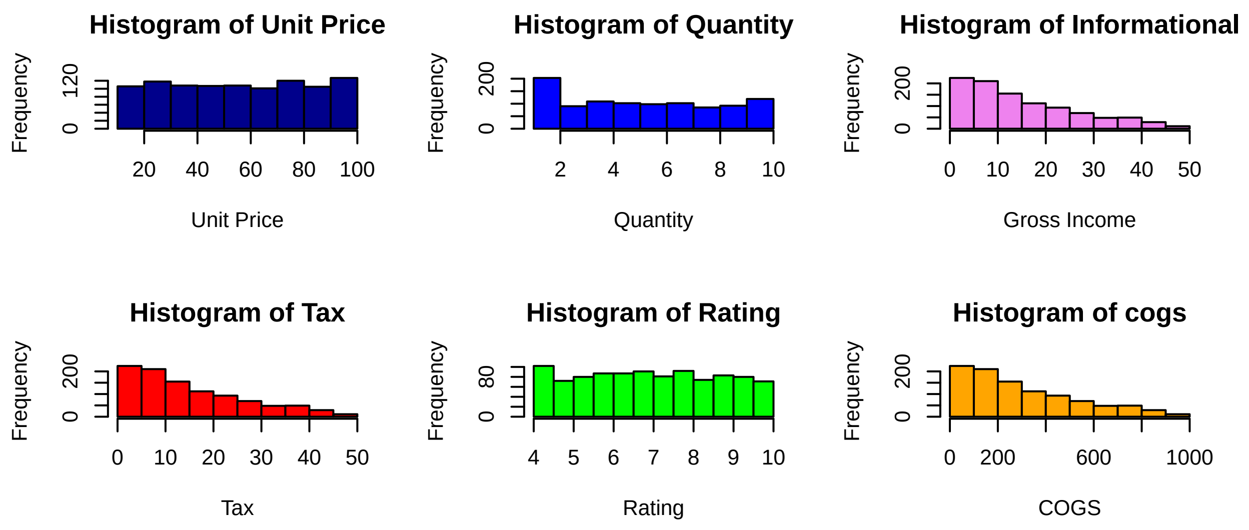
```
##  Branch  Customer.type    Gender                    Product.line
##  A:340   Member:501    Female:501   Electronic accessories:170
##  B:332   Normal:499    Male  :499   Fashion accessories   :178
##  C:328                              Food and beverages    :174
##                                     Health and beauty     :152
##                                     Home and lifestyle    :160
##                                     Sports and travel     :166
##    Unit.price       Quantity          Tax              Month
##  Min.   :10.08   Min.   : 1.00   Min.   : 0.5085   Min.   :1.000
##  1st Qu.:32.88   1st Qu.: 3.00   1st Qu.: 5.9249   1st Qu.:1.000
##  Median :55.23   Median : 5.00   Median :12.0880   Median :2.000
##  Mean   :55.67   Mean   : 5.51   Mean   :15.3794   Mean   :1.993
##  3rd Qu.:77.94   3rd Qu.: 8.00   3rd Qu.:22.4453   3rd Qu.:3.000
##  Max.   :99.96   Max.   :10.00   Max.   :49.6500   Max.   :3.000
##       Day             Hour          Minutes             Payment
##  Min.   : 1.00   Min.   :10.00   Min.   : 0.0   Cash       :344
##  1st Qu.: 8.00   1st Qu.:12.00   1st Qu.:16.0   Credit card:311
##  Median :15.00   Median :15.00   Median :30.0   Ewallet    :345
##  Mean   :15.26   Mean   :14.91   Mean   :30.1
##  3rd Qu.:23.00   3rd Qu.:18.00   3rd Qu.:44.0
##  Max.   :31.00   Max.   :20.00   Max.   :59.0
##       cogs         gross.income         Rating           Total
##  Min.   : 10.17   Min.   : 0.5085   Min.   : 4.000   Min.   :  10.68
##  1st Qu.:118.50   1st Qu.: 5.9249   1st Qu.: 5.500   1st Qu.: 124.42
##  Median :241.76   Median :12.0880   Median : 7.000   Median : 253.85
##  Mean   :307.59   Mean   :15.3794   Mean   : 6.973   Mean   : 322.97
##  3rd Qu.:448.90   3rd Qu.:22.4453   3rd Qu.: 8.500   3rd Qu.: 471.35
##  Max.   :993.00   Max.   :49.6500   Max.   :10.000   Max.   :1042.65
```

ii.) Bivariate nalysis

- Histograms

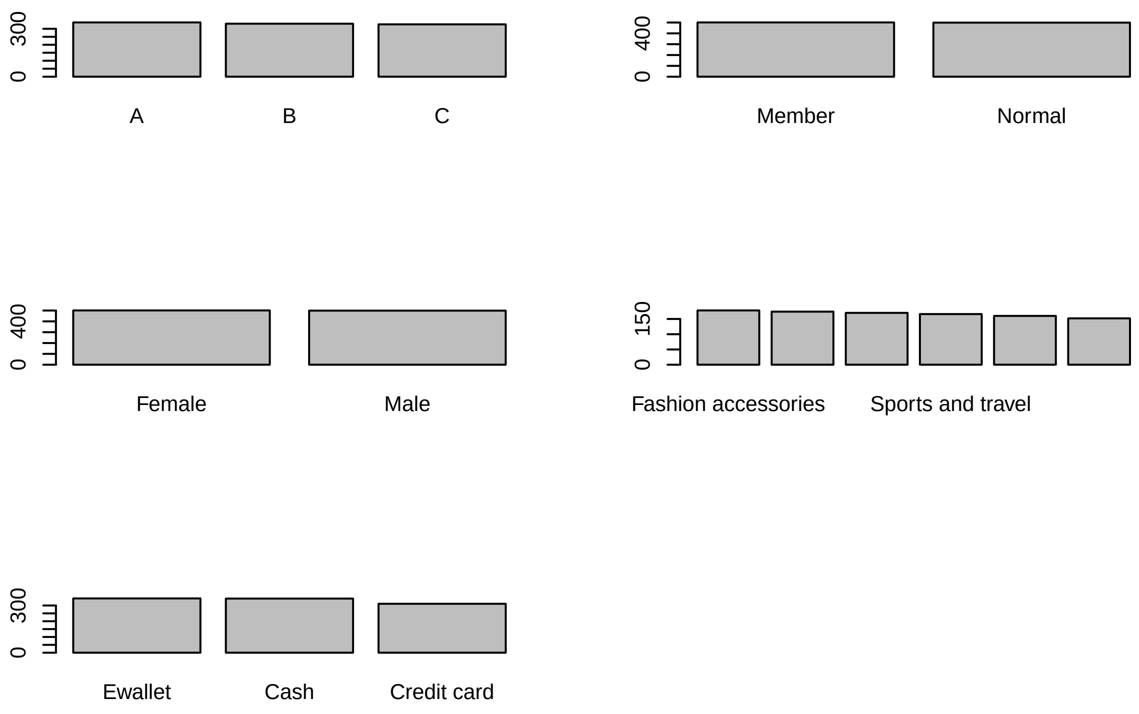
We'll plot histograms of each of the columns to see their distribution.

```r
#Plotting a histogram for numerical variables to understand their distribution
par(mfrow=c(3,3))
hist(data$Unit.price, main = "Histogram of Unit Price", xlab = "Unit Price" ,col = "darkblue")
hist(data$Quantity, main = "Histogram of Quantity", xlab = "Quantity", col="blue")
hist(data$gross.income, main = "Histogram of Informational", xlab = "Gross Income", col = "violet")
hist(data$Tax, main = "Histogram of Tax", xlab = "Tax",col="red")
hist(data$Rating, main = "Histogram of Rating", xlab = "Rating", col = "green")
hist(data$cogs, main = "Histogram of cogs", xlab = "COGS", col = "orange")
```



Majority of the numerical variables are skewed to the right while some try to depict a not so elaborate normal didtribution.

```r
#Plotting a barplot for the categorical variables
par(mfrow=c(3,2))
barplot(sort(table(data$Branch),decreasing=T))
barplot(sort(table(data$Customer.type),decreasing=T))
barplot(sort(table(data$Gender),decreasing=T))
barplot(sort(table(data$Product.line),decreasing=T))
barplot(sort(table(data$Payment),decreasing=T))
```

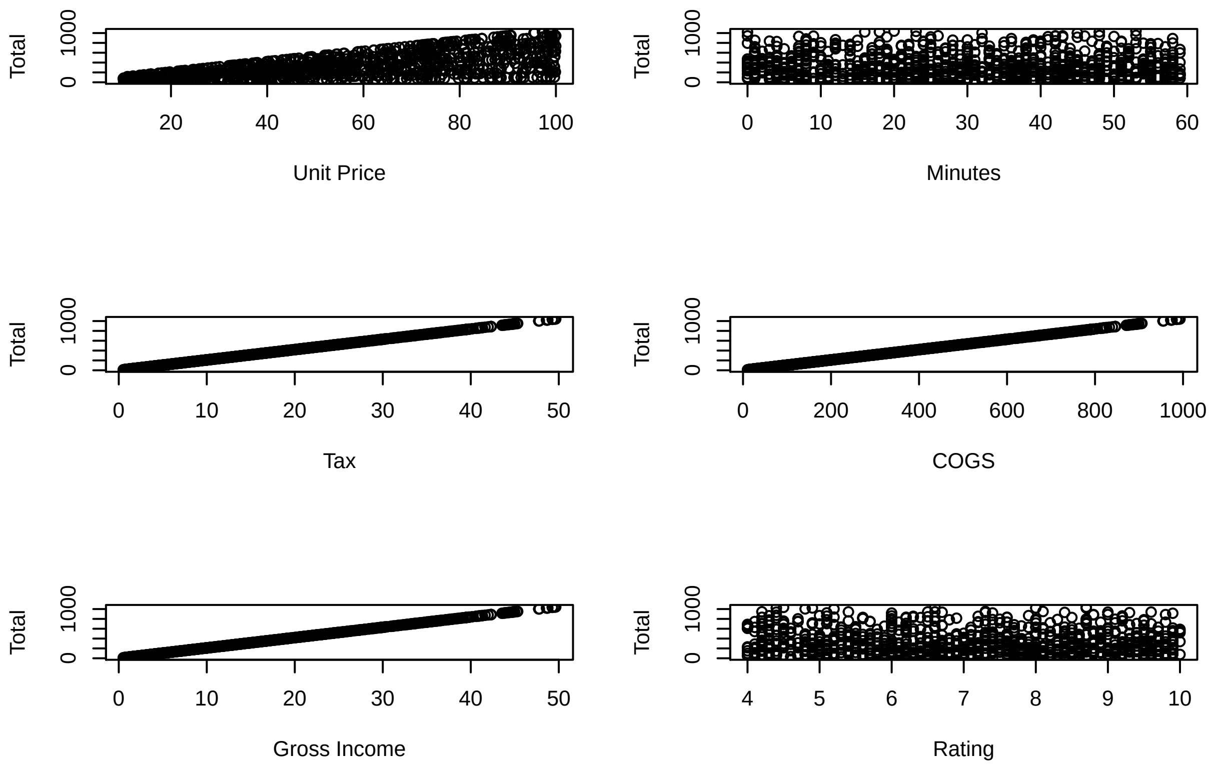


Most of the categorical variables are distribute equaly i.e. each category is the same e.g. the male are just as many as the females, people who pay with Ewallet are just as many as the ones who pay with cash and credit card same with the customer type and Branch. The Product line seems to differ a little bit but they are all in thesame range.

- Scatter Plots

We'll plot scatter plots to understand how each numerical variable relates to the total amount of the transaction.

```r
#Plotting a scatter plot of each column and their relation with the Total
par(mfrow=c(3,2))
plot(data$Unit.price, data$Total, xlab="Unit Price", ylab="Total")
plot(data$Minutes, data$Total, xlab="Minutes", ylab="Total")
plot(data$Tax, data$Total, xlab="Tax", ylab="Total")
plot(data$cogs, data$Total, xlab="COGS", ylab="Total")
plot(data$gross.income, data$Total, xlab="Gross Income", ylab="Total")
plot(data$Rating, data$Total, xlab="Rating", ylab="Total")
```

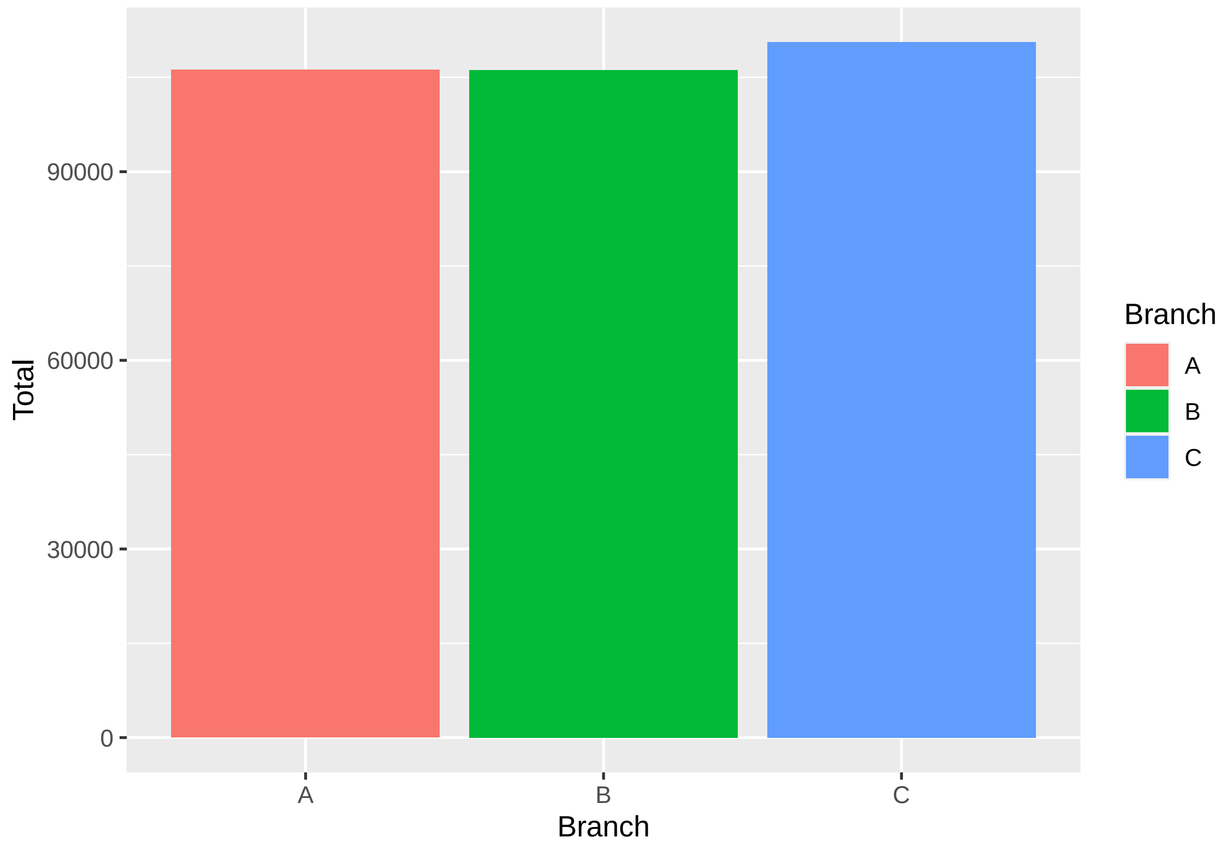


The unit price, tax, cost of goods acquired and gross income tend to increase with the increase in the total amount of the transaction.

- Barplots

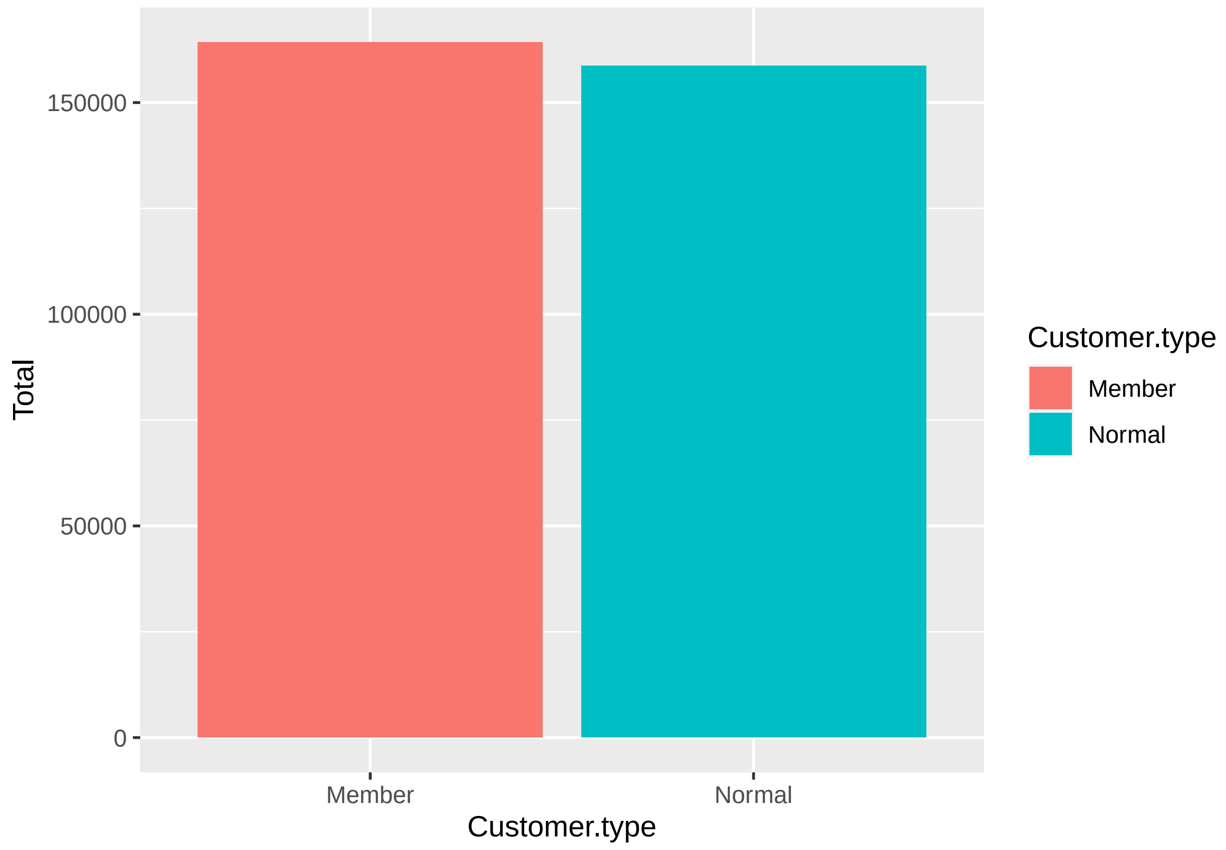
We'll plot bar plots to show how the categorical variables affect the total amount of the transactions.

```r
#Plotting a bar plot for the type of Branch vs. the total
library(ggplot2)
ggplot(data = data, aes(x = Branch, y = Total)) + geom_col(aes(fill = Branch))
```



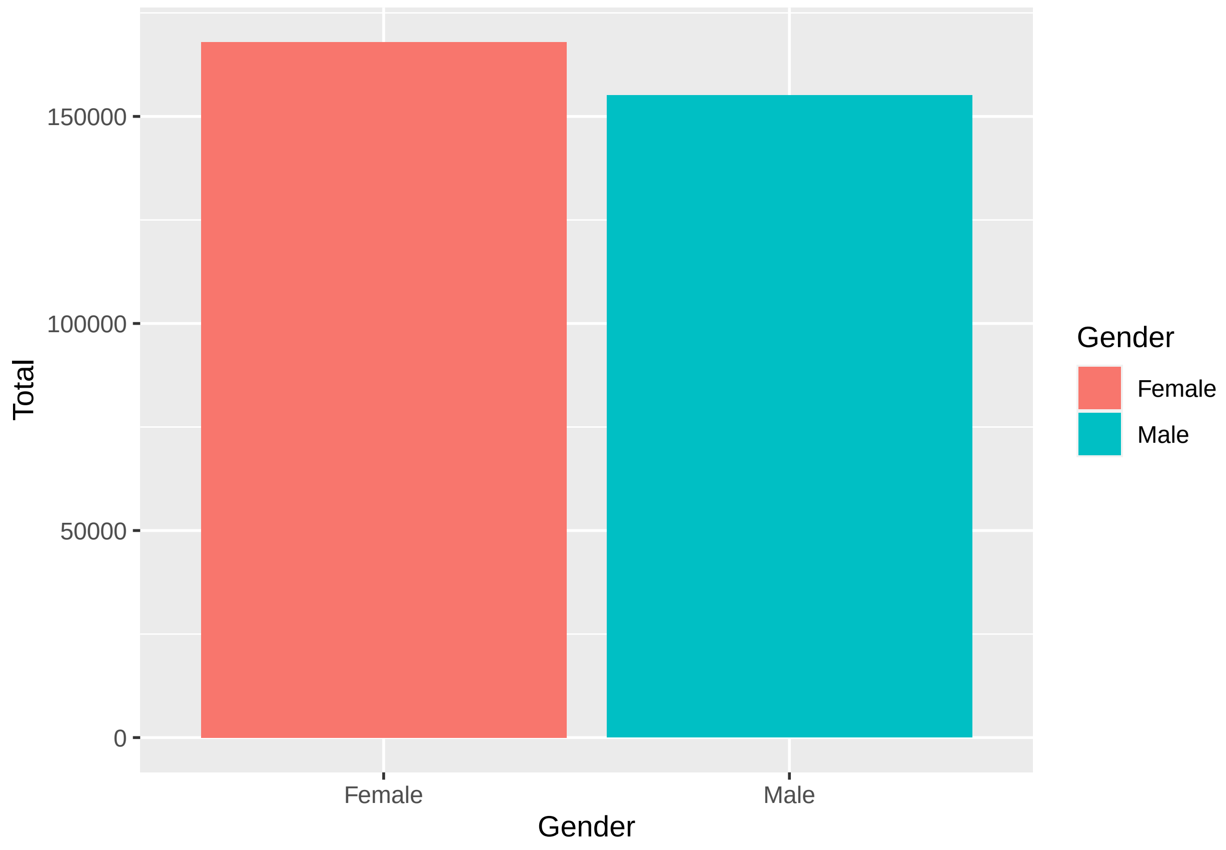
BrancH C has a lot of total amount compared to branch A and B.

```r
# Customer Type vs. Total
ggplot(data = data, aes(x = Customer.type, y = Total)) + geom_col(aes(fill = Customer.type))
```



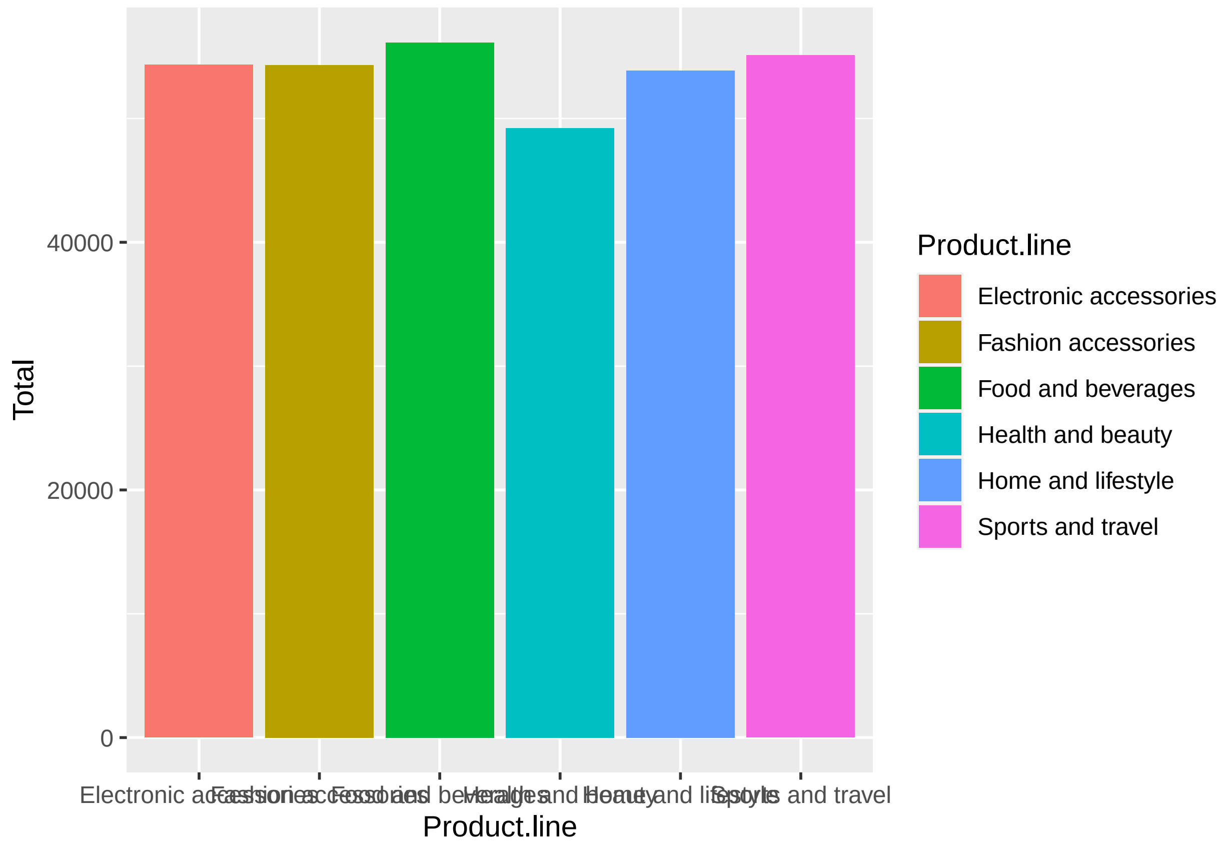
Member customers tend to spend more than the normal customers.

```r
#Gender cs. Total
ggplot(data = data, aes(x = Gender, y = Total)) + geom_col(aes(fill = Gender))
```



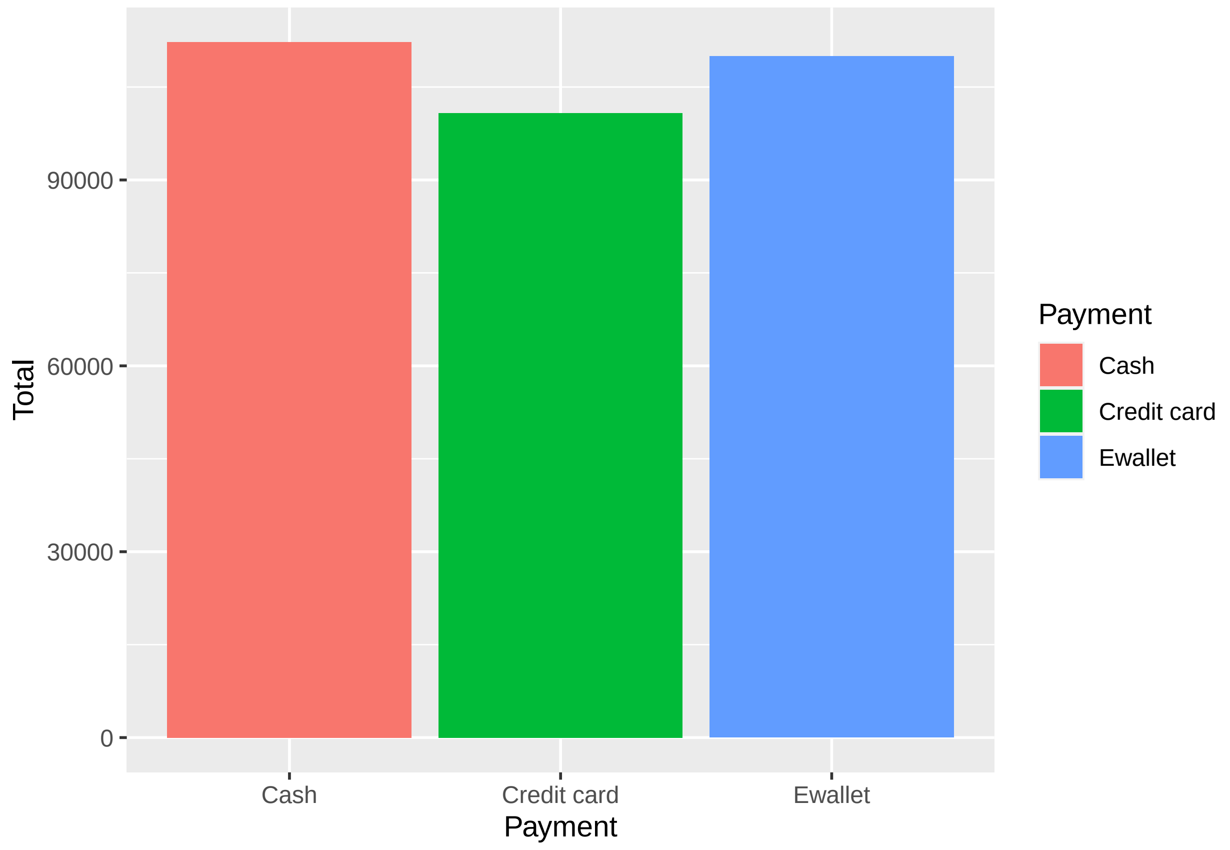
Females spend more than Males.

```r
#Product Line vs. Total
ggplot(data = data, aes(x = Product.line, y = Total)) + geom_col(aes(fill = Product.line))
```



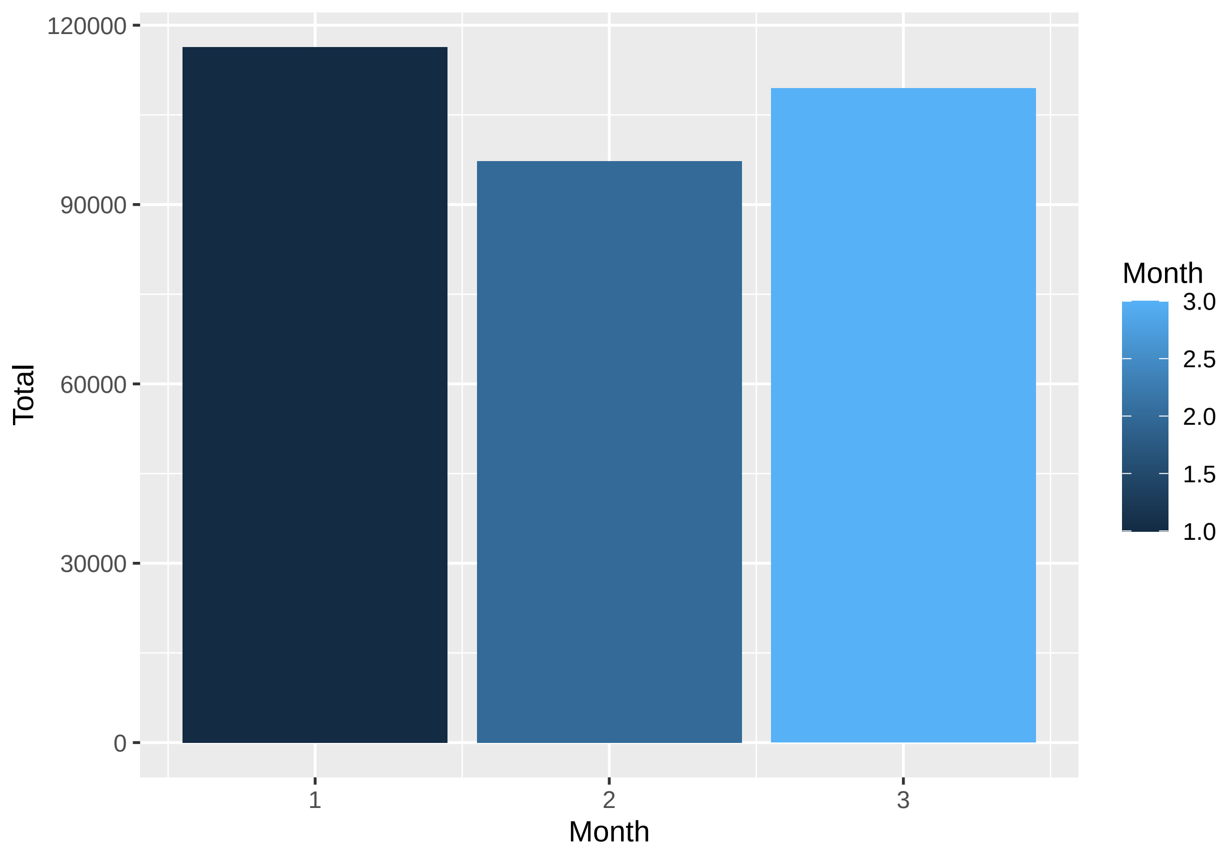
Food and Beverages and Sports and Trvale are the products line which has a lot of spending.

```
#Payment vs Total
ggplot(data = data, aes(x = Payment, y = Total)) + geom_col(aes(fill = Payment))
```



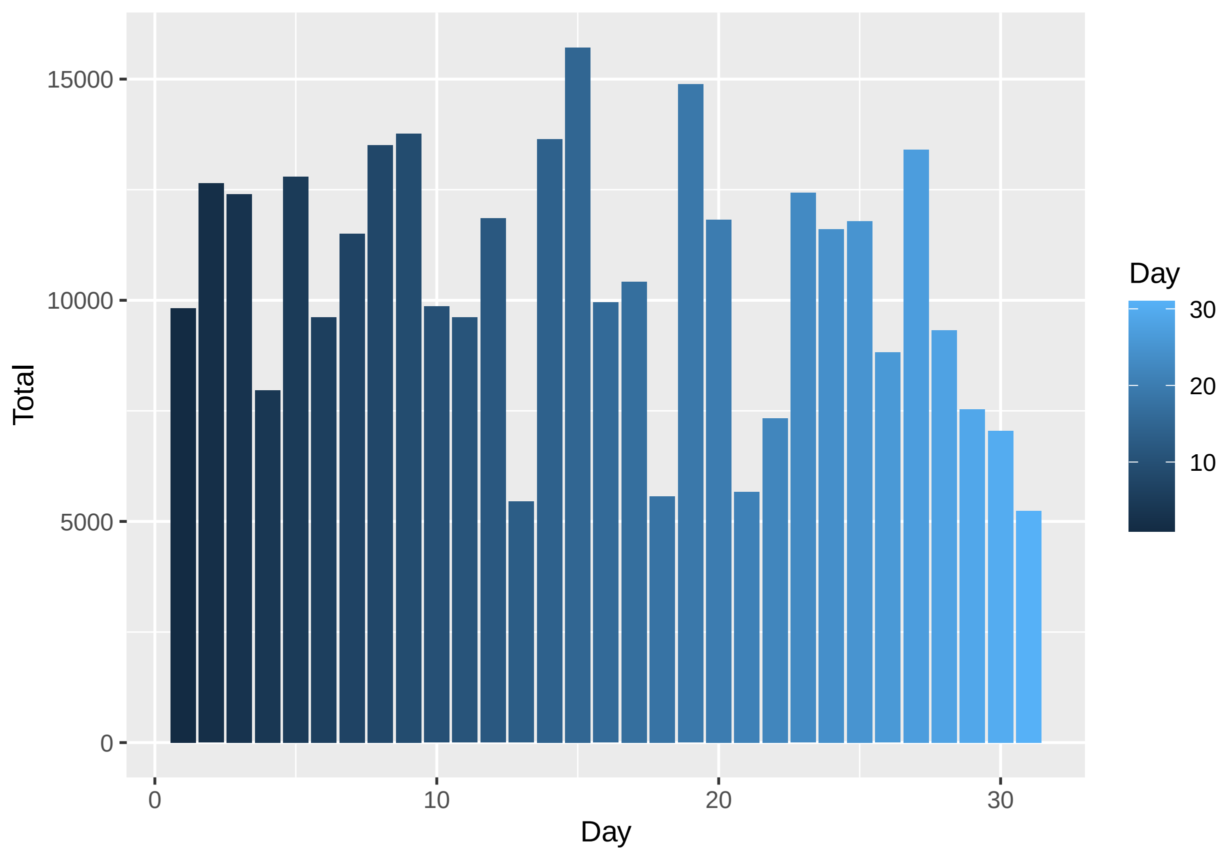
Those who pay with cash spend more followed by those who pay with Ewallet then those who pay with Credit Card contribute less to the total amount spent.

```
#Month vs. Total
ggplot(data = data, aes(x = Month, y = Total)) + geom_col(aes(fill = Month))
```



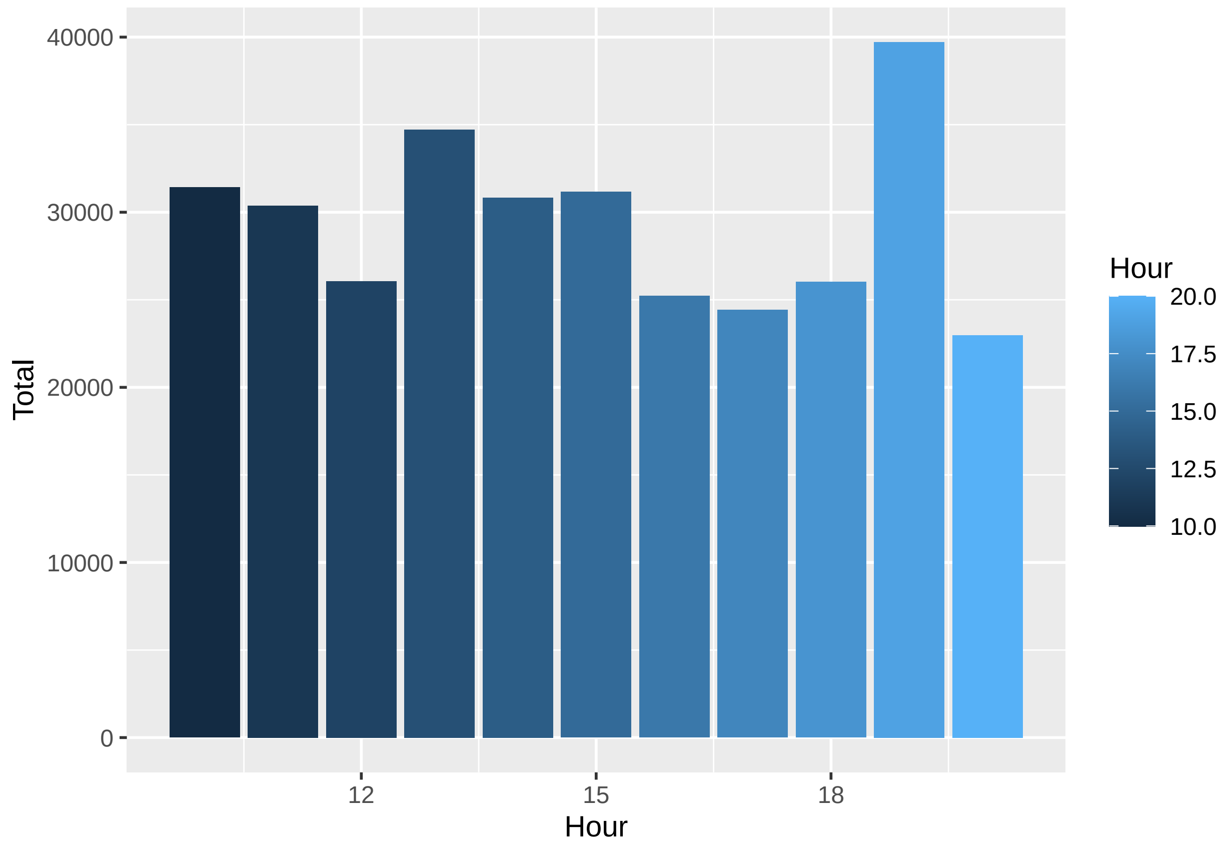
A lot of shopping is done in January followed by March then February is the least.

```
#Day vs. Total
ggplot(data = data, aes(x = Day, y = Total)) + geom_col(aes(fill = Day))
```



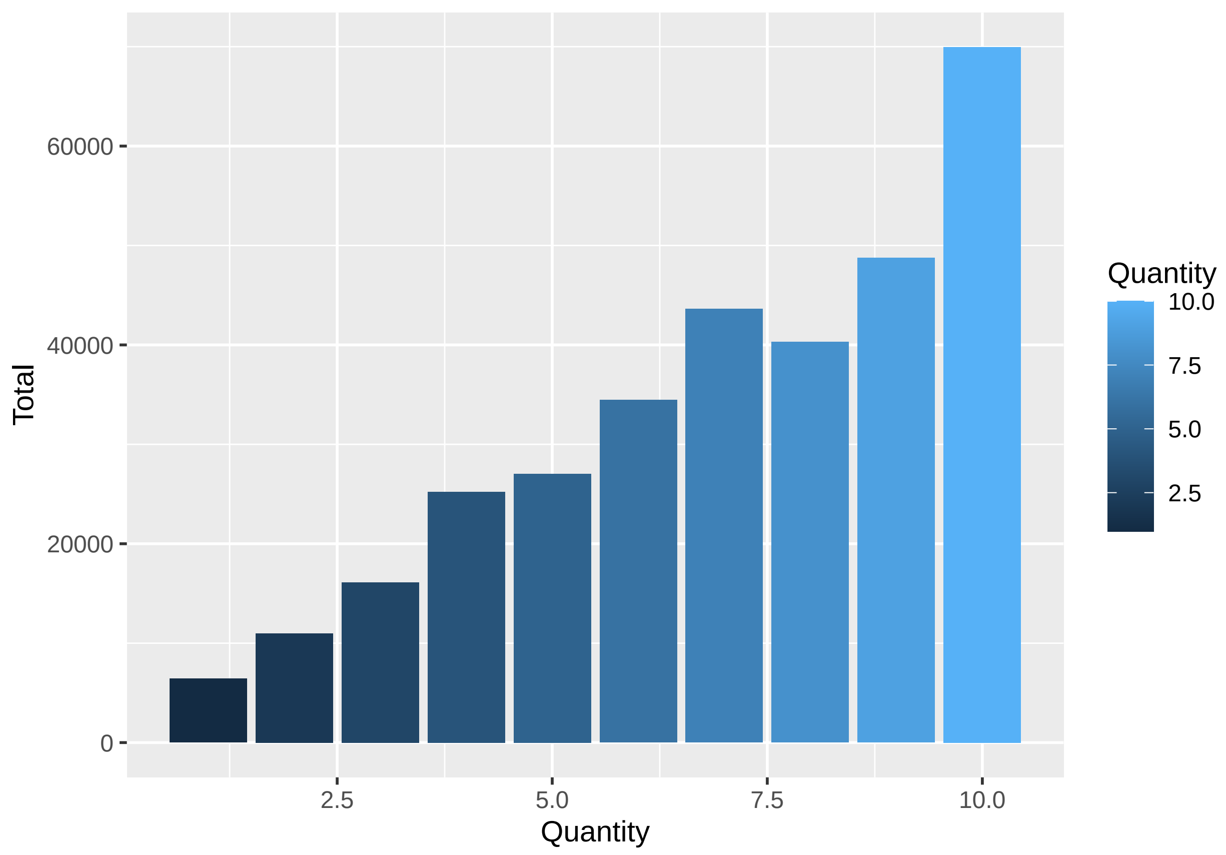
A lot of spending is one at the beginning of the month an din the middle of the month.

```
#Hour vs. Total
ggplot(data = data, aes(x = Hour, y = Total)) + geom_col(aes(fill = Hour))
```



A lot of people spend more in the evenings and early mornings.

```
#Quantity vs. Total
ggplot(data = data, aes(x = Quantity, y = Total)) + geom_col(aes(fill = Quantity))
```



The higher the Quantity of goods the more the total amount.

iii.) Correlation The correlation of each numeric variable will help in understanding the association of these random variables.

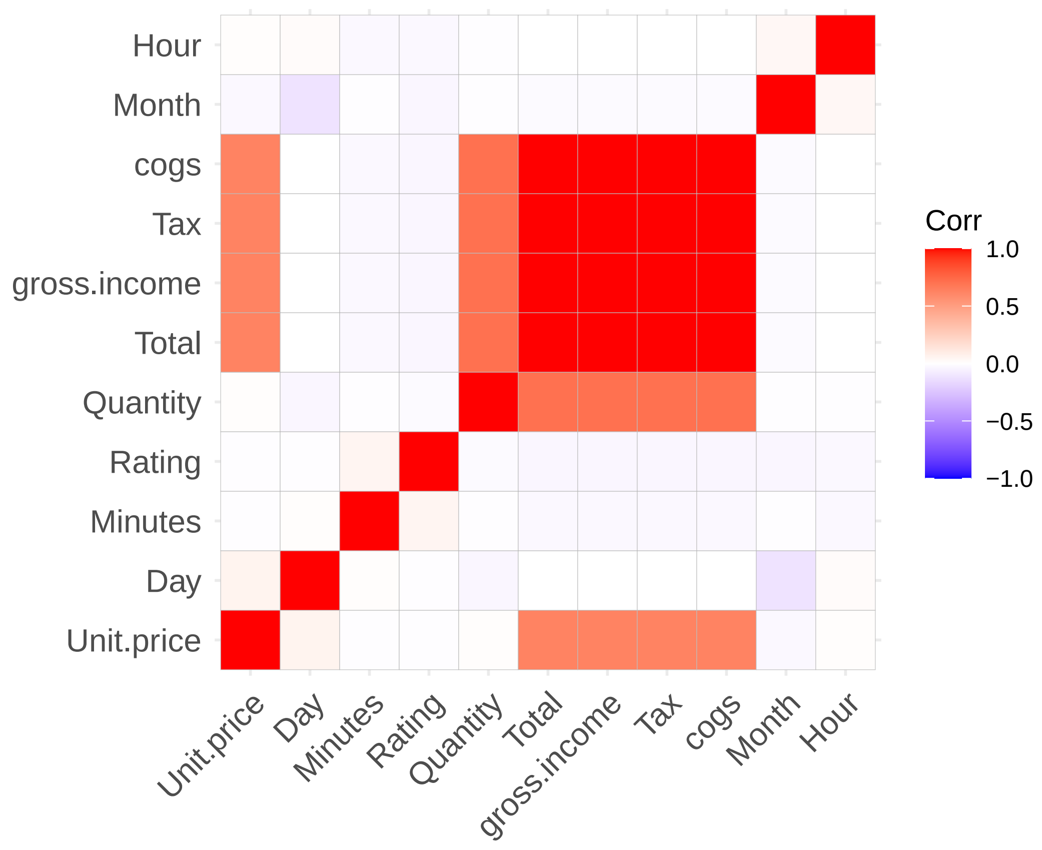
```
correlation <- round(cor(select_if(data, is.numeric)), 2)
head(correlation)
```

```
##            Unit.price Quantity   Tax Month   Day  Hour Minutes  cogs
## Unit.price       1.00     0.01  0.63 -0.03  0.06  0.01   -0.01  0.63
## Quantity         0.01     1.00  0.71 -0.01 -0.04 -0.01   -0.01  0.71
## Tax              0.63     0.71  1.00 -0.02  0.00  0.00   -0.03  1.00
## Month           -0.03    -0.01 -0.02  1.00 -0.12  0.04   -0.01 -0.02
## Day              0.06    -0.04  0.00 -0.12  1.00  0.02    0.01  0.00
## Hour             0.01    -0.01  0.00  0.04  0.02  1.00   -0.03  0.00
##            gross.income Rating Total
## Unit.price         0.63  -0.01  0.63
## Quantity           0.71  -0.02  0.71
## Tax                1.00  -0.04  1.00
## Month             -0.02  -0.04 -0.02
## Day                0.00  -0.01  0.00
## Hour               0.00  -0.03  0.00
```

The negative values imply negative correclation with ecah other while poitive imply poitive correlation. e.g. Rating is negatively correlated with all other variables.

We'll plot a heatmap to visualize the correlation matrix of our numeric variables.

```r
library(ggcorrplot)
ggcorrplot(correlation, hc.order = T)
```



e.) Principal Component Analysis We'll perform and visualize PCA in the given dataset.

```r
# Selecting the numerical data (excluding the categorical variables)
#Extract numerical and integer columns only
data2 <- select_if(data,is.numeric)
str(data2)
```

```
## 'data.frame':    1000 obs. of  11 variables:
##  $ Unit.price  : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ Quantity    : num  7 5 7 8 7 7 6 10 2 3 ...
##  $ Tax         : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ Month       : num  1 3 3 1 2 3 2 2 1 2 ...
##  $ Day         : num  5 8 3 27 8 25 25 24 10 20 ...
##  $ Hour        : num  13 10 13 20 10 18 14 11 17 13 ...
##  $ Minutes     : num  8 29 23 33 37 30 36 38 15 27 ...
##  $ 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 ...
```

```r
# We then pass the data to the prcomp().
#We also set two arguments, center and scale, to be FALSE and TRUE respectively then preview our object
data.pca <- prcomp(data2, center = FALSE, scale. = TRUE)
summary(data.pca)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6    PC7
## Standard deviation     3.0404 0.96601 0.49704 0.45316 0.43985 0.32131 0.2249
## Proportion of Variance 0.8404 0.08483 0.02246 0.01867 0.01759 0.00939 0.0046
## Cumulative Proportion  0.8404 0.92518 0.94764 0.96631 0.98390 0.99329 0.9979
##                            PC8       PC9      PC10      PC11
## Standard deviation     0.15257 3.411e-16 1.409e-16 7.442e-17
## Proportion of Variance 0.00212 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00 1.000e+00 1.000e+00
```

As a result we obtain 11 principal components, each which explain a percentage of the total variation of the dataset. PC1 explains 84% of the total variance, which means that more than three-quarters of the information in the dataset (11 variables) can be encapsulated by just that one Principal Component. PC2 explains 8.4% of the variance, PC2 - 2.2% and so forth and so forth.

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

```
## List of 5
##  $ sdev    : num [1:11] 3.04 0.966 0.497 0.453 0.44 ...
##  $ rotation: num [1:11, 1:11] -0.31 -0.309 -0.309 -0.291 -0.278 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "Unit.price" "Quantity" "Tax" "Month" ...
##   .. ..$ : chr [1:11] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : logi FALSE
##  $ scale   : Named num [1:11] 61.68 6.24 19.34 2.16 17.57 ...
##   ..- attr(*, "names")= chr [1:11] "Unit.price" "Quantity" "Tax" "Month" ...
##  $ x       : num [1:1000, 1:11] -3.32 -1.94 -2.83 -3.76 -3.83 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:11] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

Here we note that our pca object: The center point (center), scaling (scale), standard deviation(sdev) of each principal component. The relationship (correlation or anticorrelation, etc) between the initial variables and the principal components (rotation). The values of each sample in terms of the principal components (x)

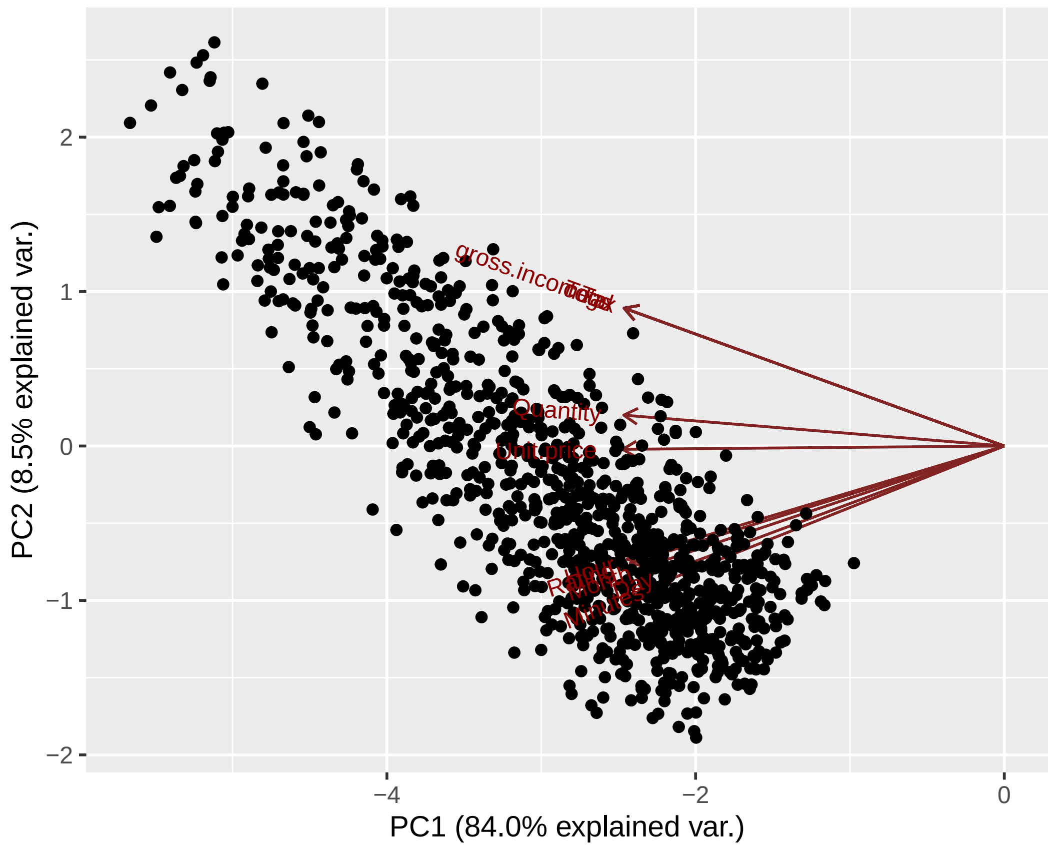
```r
#We will now plot our pca.
#Loading our ggbiplot library
library(devtools)
```

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

```r
library(ggbiplot)
```

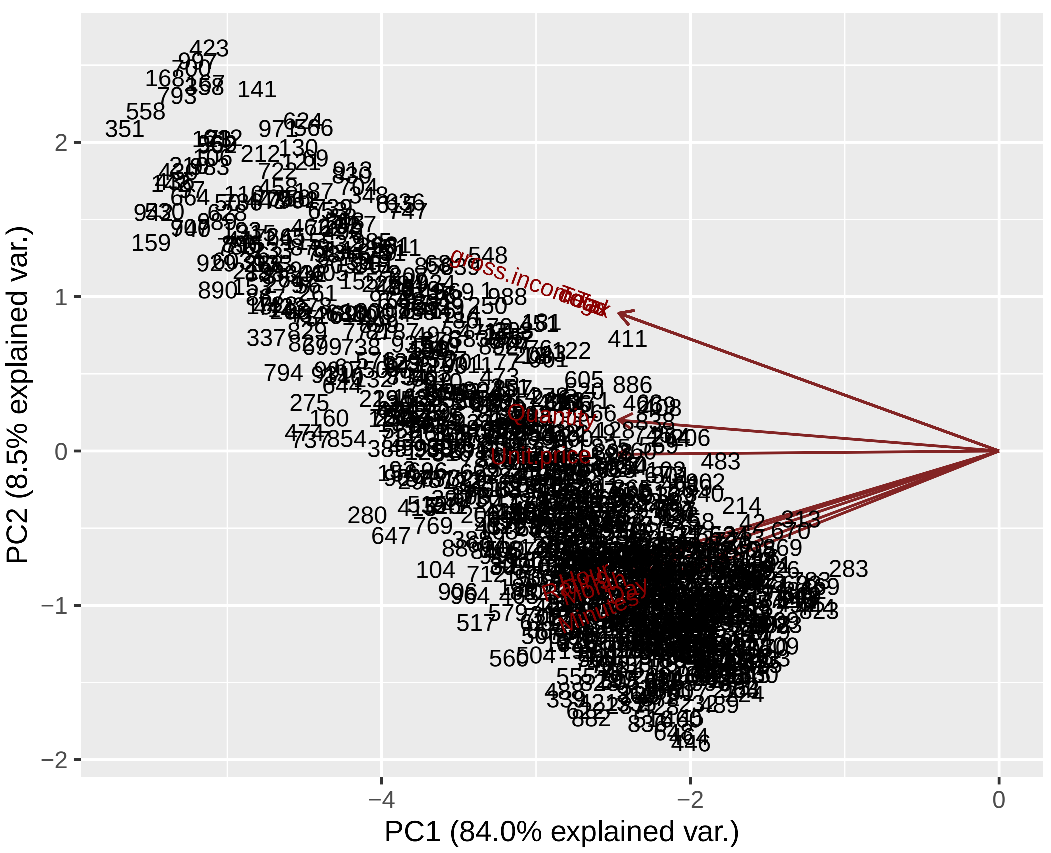
```
## Loading required package: plyr

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

## 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: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

## Loading required package: grid
```

```r
set.seed(123)
ggbiplot(data.pca, labels=rownames(data.pca),ellipse = TRUE,obs.scale=1,var.scale=1)
```



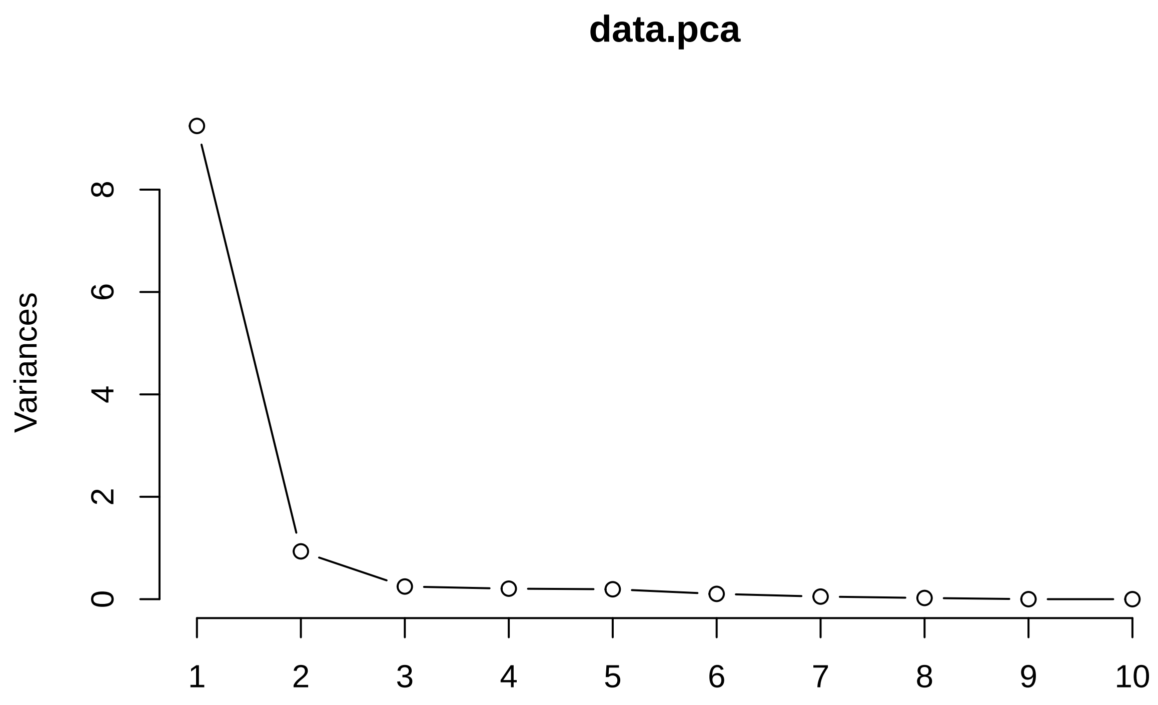
From the graph, though not so clear, we see that the variables Rating, Hour and Minutes contribute to PC1, with higher values in those variables moving the samples to the right on the plot.

```
# Adding more detail to the plot, we provide arguments rownames as labels
ggbiplot(data.pca, labels=rownames(data2), 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(data.pca, type="l")
```



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

Part 2: Feature Selection

This section requires you to perform feature selection through the use of the unsupervised learning methods learned earlier this week. You will be required to perform your analysis and provide insights on the features that contribute the most information to the dataset.

```
data <- read.csv("http://bit.ly/CarreFourDataset")
head(data)
```

```
##    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
```

```
library(tidyverse)
#Changing the date column from character to date
data$Date <- as.Date(data$Date, "%m/%d/%Y")

#Separating the Year, month and day into different column.
data <- separate(data, "Date", c("Year", "Month", "Day"), sep = "-")

#Separating the hour, minutes and seconds into different columns
data <- separate(data, "Time", c("Hour", "Minutes"), sep = ":")
#Changing the data type for some columns to factor as they are categorical
cols <- c('Branch' ,'Customer.type', 'Gender','Product.line','Payment')
data[,cols] <- lapply(data[,cols] , factor)

#Changing some character data types to numeric
data$Quantity <- as.numeric(as.character(data$Quantity))
data$Year <- as.numeric(as.character(data$Year))
data$Month <- as.numeric(as.character(data$Month))
data$Day <- as.numeric(as.character(data$Day))
data$Hour <- as.numeric(as.character(data$Hour))
data$Minutes <- as.numeric(as.character(data$Minutes))
#Dropping the gross margin percentage and year columns as they are constants throughout the dataset als
drop <- c("Invoice.ID", "gross.margin.percentage","Year")
data = data[,!(names(data) %in% drop)]
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
# Selecting the numerical data (excluding the categorical variables)
#Extract numerical and integer columns only
data2 <- select_if(data,is.numeric)
correlationMatrix <- cor(data2)
# Find attributes that are highly correlated
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75)
highlyCorrelated
```

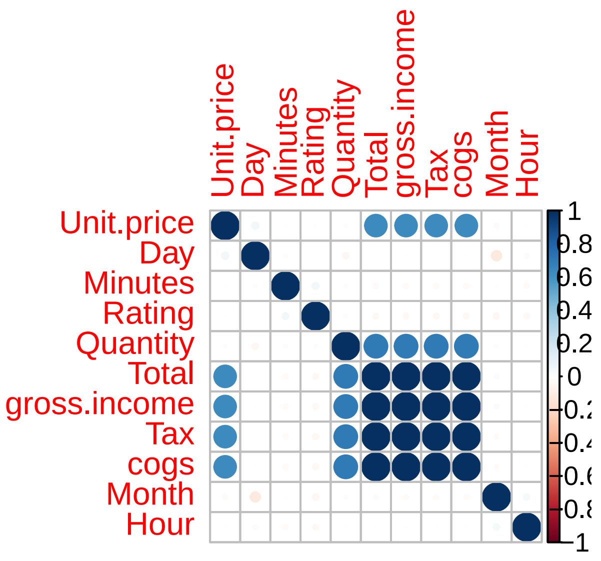
```
## [1] 3 8 9
```

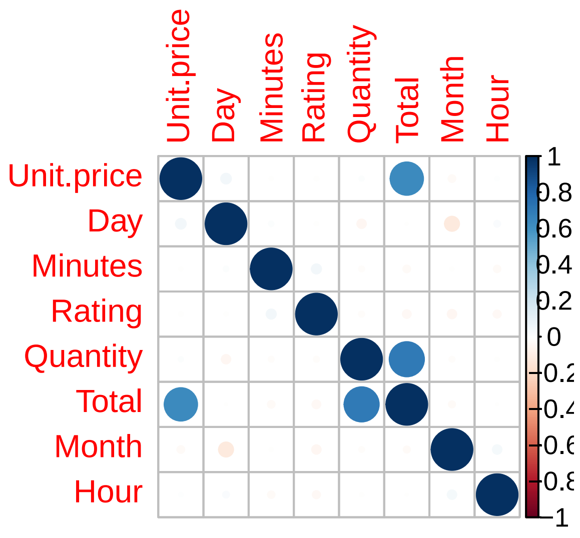
```
names(data2[,highlyCorrelated])
```

```
## [1] "Tax"          "cogs"         "gross.income"
```

The higly correlated variables are Tax, cogs(cost of goods sold) and gross income. We can remove the variables with a higher correlation and compare the results graphically.

```
# Removing Redundant Features
data3 <- data2[-highlyCorrelated]

# Performing our graphical comparison
par(mfrow = c(1, 2))
corrplot(correlationMatrix, order = "hclust")
corrplot(cor(data3), order = "hclust")
```





Removing highly correlated variables result to less coreelated variables. Hence the selected features are Unit Price, Day, Minutes, Rating, Quantity, Month and Hour.

There are no more highly correlated variables.

Part 3 : Association Rules

This section will require that you create association rules that will allow you to identify relationships between variables in the dataset. You are provided with a separate dataset that comprises groups of items that will be associated with others. Just like in the other sections, you will also be required to provide insights for your analysis.

```r
# Loading the arules library
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
path <-"http://bit.ly/SupermarketDatasetII"

association<-read.transactions(path, sep = ",")
```

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

```r
association
```

```
## transactions in sparse format with
##  7501 transactions (rows) and
##  119 items (columns)
```

```r
#Checking the shape of the data
dim(association)
```

```
## [1] 7501  119
```

```r
#Displaying the structure of our dataset
str(association)
```

```
## Formal class 'transactions' [package "arules"] with 3 slots
##   ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. ..@ i       : int [1:29358] 0 1 3 32 38 47 52 53 59 64 ...
##   .. .. ..@ p       : int [1:7502] 0 20 23 24 26 31 32 34 37 40 ...
##   .. .. ..@ Dim     : int [1:2] 119 7501
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : NULL
##   .. .. .. ..$ : NULL
##   .. .. ..@ factors : list()
##   ..@ itemInfo   :'data.frame':  119 obs. of  1 variable:
##   .. ..$ labels: chr [1:119] "almonds" "antioxydant juice" "asparagus" "avocado" ...
##   ..@ itemsetInfo:'data.frame':  0 obs. of  0 variables
```

```
# Verifying the object's class
# This should show us association as the type of data that we will need
class(df)
```

```
## [1] "function"
```

```
# Previewing our first 5 transactions
inspect(association[1:5])
```

```
##     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}
```

```
# If we wanted to preview the items that make up our dataset,
# alternatively we can do the following
# ---
```

```
#
items<-as.data.frame(itemLabels(association))
colnames(items) <- "Item"
head(items, 10)
```

```
##                   Item
## 1              almonds
## 2   antioxydant juice
## 3            asparagus
## 4              avocado
## 5          babies food
## 6                bacon
## 7       barbecue sauce
## 8            black tea
## 9          blueberries
## 10          body spray
```

```
# Generating a summary of the dataset
# This would give us some information such as the most purchased items,
# distribution of the item sets (no. of items purchased in each transaction), etc.
summary(association)
```

```
## 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
```

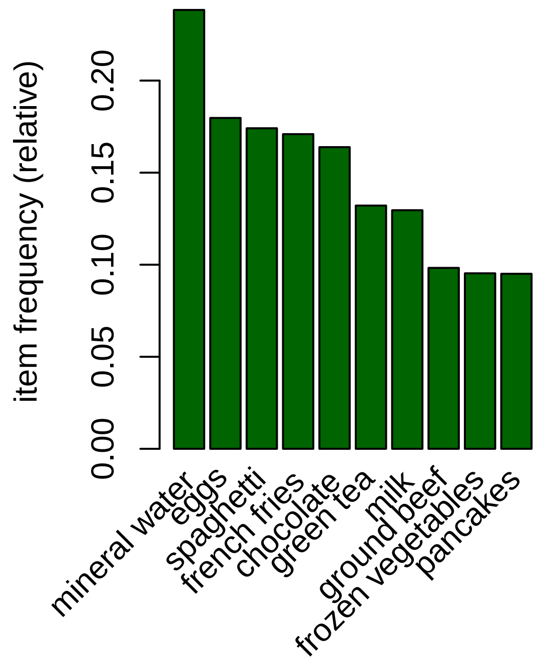
```
# Exploring the frequency of some articles
# i.e. transacations ranging from 5 to 15 and performing
# some operation in percentage terms of the total transactions
itemFrequency(association[, 5:15],type = "absolute")
```

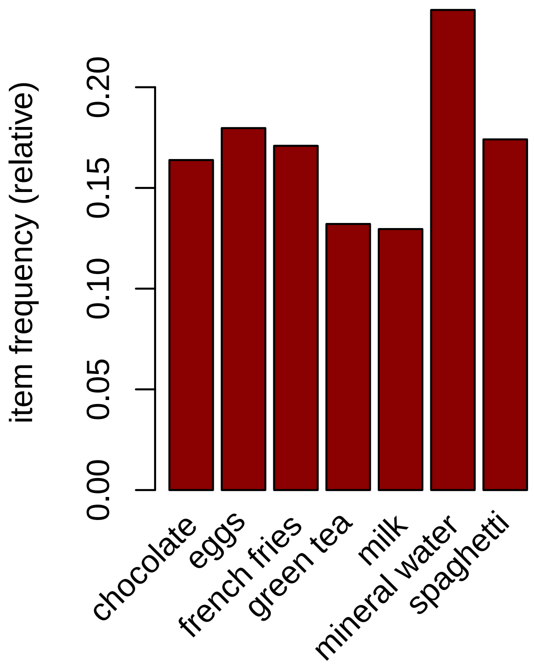
```
##    babies food          bacon barbecue sauce       black tea    blueberries
##             34             65             81             107             69
##     body spray        bramble       brownies       bug spray   burger sauce
##             86             14            253              65             44
##        burgers
##            654
```

```r
round(itemFrequency(association[, 5:15],type = "relative")*100,2)
```

```
##    babies food          bacon barbecue sauce       black tea    blueberries
##           0.45           0.87           1.08            1.43           0.92
##     body spray        bramble       brownies       bug spray   burger sauce
##           1.15           0.19           3.37            0.87           0.59
##        burgers
##           8.72
```

```r
# Producing a chart of frequencies and filtering
# to consider only items with a minimum percentage
# of support/ considering a top x of items

# Displaying top 10 most common items in the transactions dataset
# and the items whose relative importance is at least 10%
par(mfrow = c(1, 2))

# plot the frequency of items
itemFrequencyPlot(association, topN = 10,col="darkgreen")
itemFrequencyPlot(association, support = 0.1,col="darkred")
```





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.

The items whose relative importance is at least 10% are chocolate, eggs, french fries, green tea, milk, minearl water and spaghetti.

```
# Building a model based on association rules
# using the apriori function

# We use Min Support as 0.001 and confidence as 0.8

rules <- apriori (association, 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 [0.01s].
## writing ... [74 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules
```

```
## set of 74 rules
```

We use measures of significance and interest on the rules, determining which ones are interesting and which to discard.

However since we built the model using 0.001 Min support and confidence as 0.8 we obtained 74 rules.

However, in order to illustrate the sensitivity of the model to these two parameters, we will see what happens if we increase the support or lower the confidence level.

```
# Building a apriori model with Min Support as 0.002 and confidence as 0.8.
rules2 <- apriori (association,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 ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [115 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [2 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules2
```

```
## set of 2 rules
```

```
# Building apriori model with Min Support as 0.002 and confidence as 0.6.
rules3 <- apriori (association, 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 ...[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 [0.01s].
## writing ... [545 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
rules3
```

```
## set of 545 rules
```

```r
#Performing an exploration of our model using the summary function
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
##  association          7501   0.001        0.8
```

This gives us information about the model i.e. the size of rules, depending on the items that contain these rules.

In this case, most rules have 3 and 4 items though some rules do have upto 6.

More statistical information such as support, lift and confidence is also provided.

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

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

```
##      lhs                               rhs             support     confidence
## [1]  {frozen smoothie,spinach}      => {mineral water} 0.001066524 0.8888889
## [2]  {bacon,pancakes}               => {spaghetti}     0.001733102 0.8125000
## [3]  {nonfat milk,turkey}           => {mineral water} 0.001199840 0.8181818
## [4]  {ground beef,nonfat milk}      => {mineral water} 0.001599787 0.8571429
## [5]  {mushroom cream sauce,pasta} => {escalope}      0.002532996 0.9500000
## [6]  {milk,pasta}                   => {shrimp}        0.001599787 0.8571429
## [7]  {cooking oil,fromage blanc}    => {mineral water} 0.001199840 0.8181818
## [8]  {black tea,salmon}             => {mineral water} 0.001066524 0.8000000
## [9]  {black tea,frozen smoothie}    => {milk}          0.001199840 0.8181818
## [10] {red wine,tomato sauce}        => {chocolate}     0.001066524 0.8000000
##      coverage    lift      count
## [1]  0.001199840  3.729058  8
## [2]  0.002133049  4.666587 13
## [3]  0.001466471  3.432428  9
## [4]  0.001866418  3.595877 12
## [5]  0.002666311 11.976387 19
## [6]  0.001866418 11.995203 12
## [7]  0.001466471  3.432428  9
## [8]  0.001333156  3.356152  8
## [9]  0.001466471  6.313973  9
## [10] 0.001333156  4.882669  8
```

The above shows that a customer who gets frozen smoothie and spinach, their chances of geting mineral water as well is 88% and so forth.

```
# Ordering these rules by a criteria such as the level of confidence
# then looking at the first five rules.

rules<-sort(rules, by="confidence", decreasing=TRUE)
inspect(rules[1:5])
```

```
##     lhs                                          rhs             support
## [1] {french fries,mushroom cream sauce,pasta} => {escalope}      0.001066524
## [2] {ground beef,light cream,olive oil}       => {mineral water} 0.001199840
## [3] {cake,meatballs,mineral water}            => {milk}          0.001066524
## [4] {cake,olive oil,shrimp}                   => {mineral water} 0.001199840
## [5] {mushroom cream sauce,pasta}              => {escalope}      0.002532996
##     confidence coverage    lift      count
## [1] 1.00       0.001066524 12.606723  8
## [2] 1.00       0.001199840  4.195190  9
## [3] 1.00       0.001066524  7.717078  8
## [4] 1.00       0.001199840  4.195190  9
## [5] 0.95       0.002666311 11.976387 19
```

```r
#Creating a subset of rules concerning milk

#This would tell us the items that the customers bought before purchasing milk
milk <- subset(rules, subset = rhs %pin% "milk")

# Then order by confidence
milk <- sort(milk, by="confidence", decreasing=TRUE)
inspect(milk[1:5])
```

```
##     lhs                                   rhs    support     confidence
## [1] {cake,meatballs,mineral water}    => {milk} 0.001066524 1.0000000
## [2] {escalope,hot dogs,mineral water} => {milk} 0.001066524 0.8888889
## [3] {meatballs,whole wheat pasta}     => {milk} 0.001333156 0.8333333
## [4] {black tea,frozen smoothie}       => {milk} 0.001199840 0.8181818
## [5] {burgers,ground beef,olive oil}   => {milk} 0.001066524 0.8000000
##     coverage    lift     count
## [1] 0.001066524 7.717078  8
## [2] 0.001199840 6.859625  8
## [3] 0.001599787 6.430898 10
## [4] 0.001466471 6.313973  9
## [5] 0.001333156 6.173663  8
```

This shows that customers who bought cake, meatballs and mineral water automatically bought milk as there confidence is 100%.

```r
#Determining items that customers might buy if they previously bought milk?

# Subset the rules
milk <- subset(rules, subset = lhs %pin% "milk")

# Order by confidence
milk<-sort(milk, by="confidence", decreasing=TRUE)

# inspect top 5
inspect(milk[15:19])
```

```
##     lhs                                 rhs             support     confidence
## [1] {chocolate,hot dogs,milk}       => {mineral water} 0.001066524 0.8
## [2] {avocado,burgers,milk}          => {spaghetti}     0.001066524 0.8
## [3] {cookies,green tea,milk}        => {french fries}  0.001066524 0.8
## [4] {cake,eggs,milk,turkey}         => {mineral water} 0.001066524 0.8
## [5] {chocolate,eggs,milk,olive oil} => {mineral water} 0.001066524 0.8
##     coverage    lift     count
## [1] 0.001333156 3.356152 8
## [2] 0.001333156 4.594793 8
## [3] 0.001333156 4.680811 8
## [4] 0.001333156 3.356152 8
## [5] 0.001333156 3.356152 8
```

Mineral water is what most customers buy after immediately buying milk with a confidence of 80%

Part 4 : Anomaly Detection

We'll also check whether there are any anomalies in the given sales dataset. The objective of this task being fraud detection.

```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(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library(tibble)
library(tidyverse)
library(anomalize)
#anom <- read.csv("http://bit.ly/CarreFourSalesDataset")
anom <- data.table::fread("http://bit.ly/CarreFourSalesDataset")
head(anom)
```

```
##         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
str(anom)
```

```
## Classes 'data.table' and '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 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
names(anom) <- c("Date", "Sales")
anom$Date <- as.Date(anom$Date, "%m/%d/%Y")
anom$Sales <- as.numeric(anom$Sales)
```

```r
summary(anom)
```

```
##       Date                 Sales
##  Min.   :2019-01-01   Min.   :  10.68
##  1st Qu.:2019-01-24   1st Qu.: 124.42
##  Median :2019-02-13   Median : 253.85
##  Mean   :2019-02-14   Mean   : 322.97
##  3rd Qu.:2019-03-08   3rd Qu.: 471.35
##  Max.   :2019-03-30   Max.   :1042.65
```

```
#Creating a count column based on the number of times the Date appears
library(dplyr)
anom2 <- anom %>% add_count(Date, name = "count")
anom2
```

```
##             Date     Sales count
##    1: 2019-01-05  548.9715    12
##    2: 2019-03-08   80.2200    11
##    3: 2019-03-03  340.5255    14
##    4: 2019-01-27  489.0480    14
##    5: 2019-02-08  634.3785    12
##   ---
##  996: 2019-01-29   42.3675    12
##  997: 2019-03-02 1022.4900    18
##  998: 2019-02-09   33.4320    13
##  999: 2019-02-22   69.1110    11
## 1000: 2019-02-18  649.2990     7
```

```
colSums(is.na(anom2))
```

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

```
library(tibbletime)
```

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

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

```
#Convert the data frame to tibble
anom3 <- as_tibble(anom2)
#Covert the date column to a time tibble
anom3 <- as_tbl_time(anom3, Date)
Date <- anom3$Date
#class(anom3)
columns = colnames(anom3)
for (column in seq(length(colnames(anom3)))){
print(columns[column])
print(class(anom3[, column]))
cat('\n')
}
```

```
## [1] "Date"
## [1] "tbl_time"   "tbl_df"     "tbl"        "data.frame"
##
## [1] "Sales"
## [1] "tbl_df"     "tbl"        "data.frame"
##
## [1] "count"
## [1] "tbl_df"     "tbl"        "data.frame"
```

```r
library(tibbletime)
anom3 <- as_tbl_time(anom3, index = Date)
```

Upon running the code below to build an anomaly detection model, I kept getting an error; Error: Error time_decompose(): Object is not of class `tbl_df` or `tbl_time`.

anom3 %>% as_period("monthly") time_decompose(count) %>% anomalize(remainder) %>% time_recompose() %>% plot_anomalies(time_recomposed = TRUE, ncol = 3, alpha_dots = 0.5)

Attempts to correct the error were unfruitful hence the model was not built and we could therefore not build a fraud detection model.