Dimensionality Reduction & Feature Selection

Geoffrey Chege

2022-06-10

1. Introduction

1.1 Defining the question

- I am a Data analyst at Carrefour Kenya and I am 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).

1.2 Metric for success

- Be able to reduce the dataset to a low dimensional dataset using the t-SNE algorithm or PCA.

1.3 Understanding the context

- Carrefour operates different store formats, as well as multiple online offerings to meet the growing needs of its diversified customer base.
- In line with the brand's commitment to provide the widest range of quality products and value for money, Carrefour offers an unrivalled choice of more than 500,000 food and non-food products, and a locally inspired exemplary customer experience to create great moments for everyone every day.

1.4 Recording the experimental design

- Problem Definition.
- Loading the necessary libraries and the dataset.
- Data Cleaning.
- Exploratory Data Analysis:
  - Univariate Analysis.
  - Bivariate Analysis.
- Part 1: Dimensionality Reduction using t-Distributed Stochastic Neighbor Embedding (t-sne).
- Part 2: Feature Engineering using unsupervised learning.
- Recommendations.

1.5 Data Relevance

- Link to the dataset: http://bit.ly/SupermarketDatasetII

2. Loading the necessary libraries and the dataset.

```
library(ggplot2)
library(Rtsne)
library(e1071)
library(lattice)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caret)
library(superml)
```

```
## Loading required package: R6
```

```
library(CatEncoders)
```

```
##
## Attaching package: 'CatEncoders'
```

```
## The following object is masked from 'package:base':
##
##     transform
```

```
library(FSelector)

library(tidyr)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(warn = -1)
library(RColorBrewer)

library(DataExplorer)
library(Hmisc)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following object is masked from 'package:e1071':
##
##     impute

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

```
library(pastecs)
```

```
##
## Attaching package: 'pastecs'

## The following object is masked from 'package:magrittr':
##
##     extract

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(psych)
```

```
##
## Attaching package: 'psych'

## The following object is masked from 'package:Hmisc':
##
##     describe

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(factoextra)
```

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

```
library(dplyr)
```

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

## The following objects are masked from 'package:pastecs':
##
##     first, last

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

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

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

```
library(ggcorrplot)
library(clustvarsel)
```

```
## Loading required package: mclust

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

##
## Attaching package: 'mclust'

## The following object is masked from 'package:psych':
##
##     sim

## Package 'clustvarsel' version 2.3.4

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

```
library(mclust)
library("cluster")
```

```
df <- read.csv("C:/Users/user/Downloads/Supermarket_Dataset_1 - Sales Data.csv")
head(df)
```

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

3. Data Cleaning.

Checking the structure of the data.

```
str(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" "
##  $ 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 ...
```

- For the analysis, I will need to convert the character columns into factors.

Data Cleaning:

```
df$Invoice.ID <- as.factor(df$Invoice.ID)
df$Branch <- as.factor(df$Branch)
df$Customer.type <- as.factor(df$Customer.type)
df$Gender <- as.factor(df$Gender)
df$Product.line <- as.factor(df$Product.line)
df$Payment <- as.factor(df$Payment)
df$Date <- as.Date(df$Date, format = "%m/%d/%y")

str(df) #confirming the changes
```

```
## 'data.frame':    1000 obs. of  16 variables:
##  $ Invoice.ID            : Factor w/ 1000 levels "101-17-6199",..: 815 143 654 19 340 734 316 265 7
##  $ Branch                : Factor w/ 3 levels "A","B","C": 1 3 1 1 1 3 1 3 1 2 ...
##  $ Customer.type         : Factor w/ 2 levels "Member","Normal": 1 2 2 1 2 2 1 2 1 1 ...
##  $ Gender                : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 2 1 1 1 1 ...
##  $ Product.line          : Factor w/ 6 levels "Electronic accessories",..: 4 1 5 4 6 1 1 5 4 3 ...
##  $ Unit.price            : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ Quantity              : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ Tax                   : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ Date                  : Date, format: "2020-01-05" "2020-03-08" ...
##  $ Time                  : chr  "13:08" "10:29" "13:23" "20:33" ...
##  $ Payment               : Factor w/ 3 levels "Cash","Credit card",..: 3 1 2 3 3 3 3 3 2 2 ...
##  $ cogs                  : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ gross.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 ...
```

- Next, I will check for duplicates:

```
# checking for duplicates
df[duplicated(df), ]
```

```
##  [1] Invoice.ID             Branch                 Customer.type
##  [4] Gender                 Product.line           Unit.price
##  [7] Quantity               Tax                    Date
## [10] Time                   Payment                cogs
## [13] gross.margin.percentage gross.income          Rating
## [16] Total
## <0 rows> (or 0-length row.names)
```

- There are no duplicates in the dataset.
- Checking for mssing values:

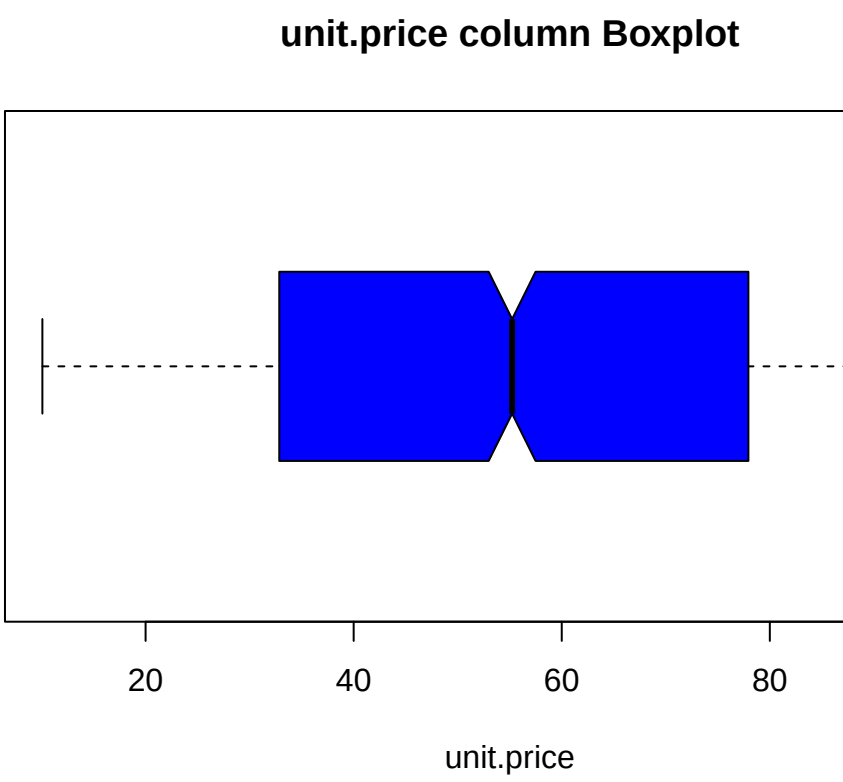
```
# checking for missing values
colSums(is.na(df))
```

```
##               Invoice.ID                  Branch           Customer.type
##                        0                       0                       0
##                   Gender            Product.line              Unit.price
##                        0                       0                       0
##                 Quantity                     Tax                    Date
##                        0                       0                       0
##                     Time                 Payment                    cogs
##                        0                       0                       0
## gross.margin.percentage            gross.income                  Rating
##                        0                       0                       0
##                    Total
##                        0
```

- There are no missing values in the dataset.

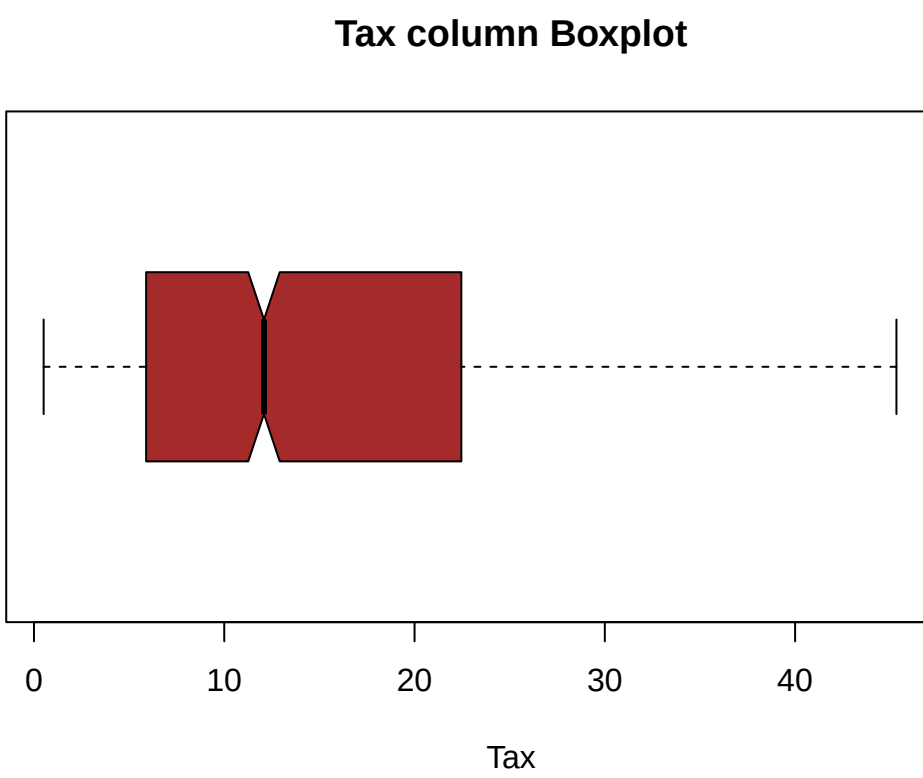
Outliers

- I will use box plots to check for outliers.



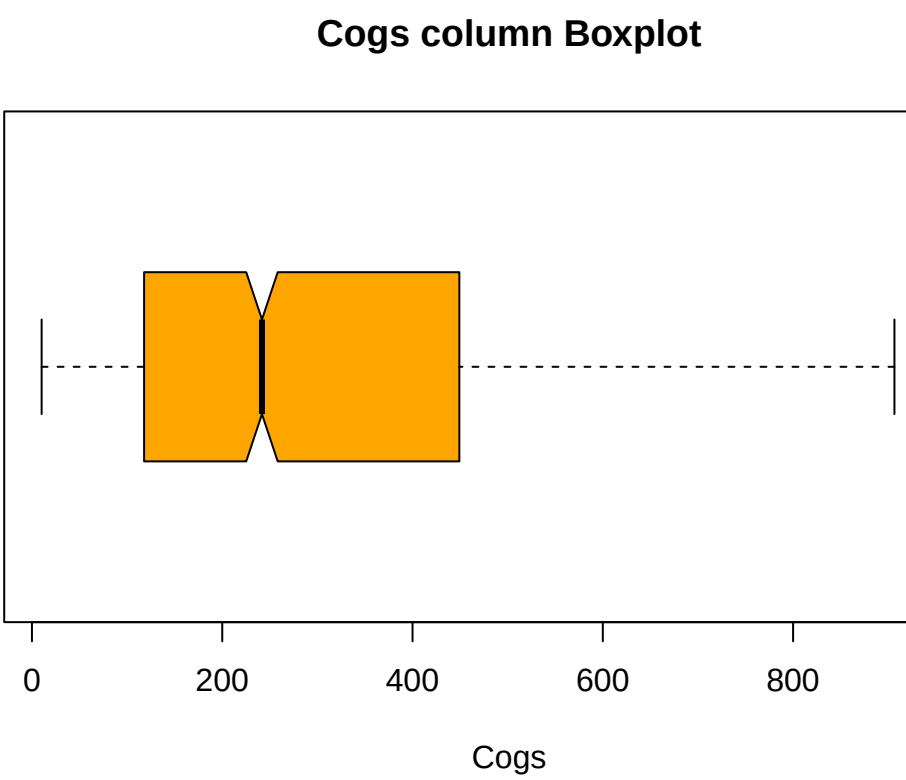
Boxplot for "unit.price" column

- There are no outliers in the "unit.price" column.



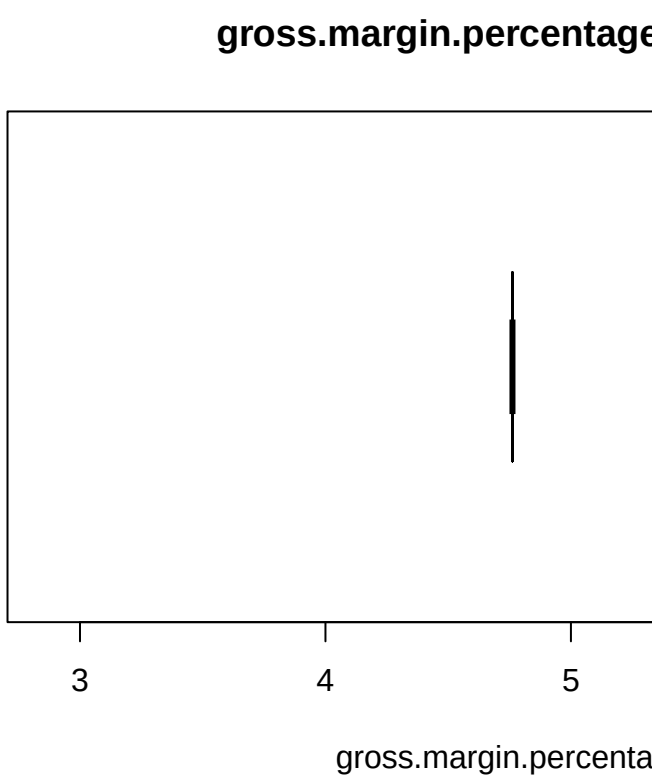
Boxplot for "Tax" column

- There are outliers in the "Tax" column. I will keep them in my analysis because they represent true values in the data.



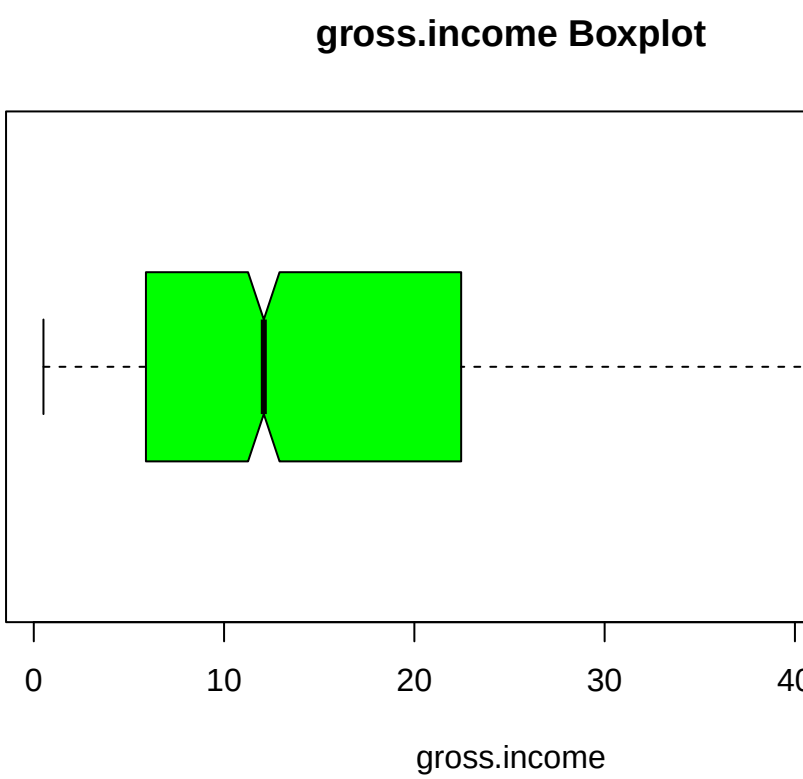
Boxplot for "Cogs" column

- There are outliers in the "Cogs" column. I will keep them in my analysis because they represent true values in the data.



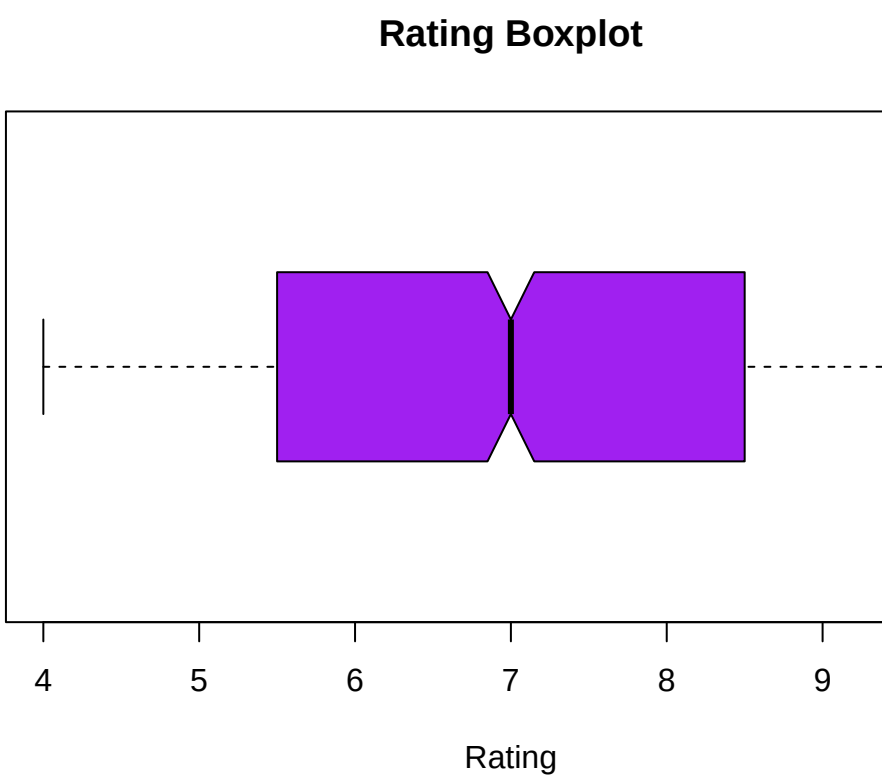
Boxplot for "gross.margin.percentage" column

- There are no outliers in the "Tax" column.



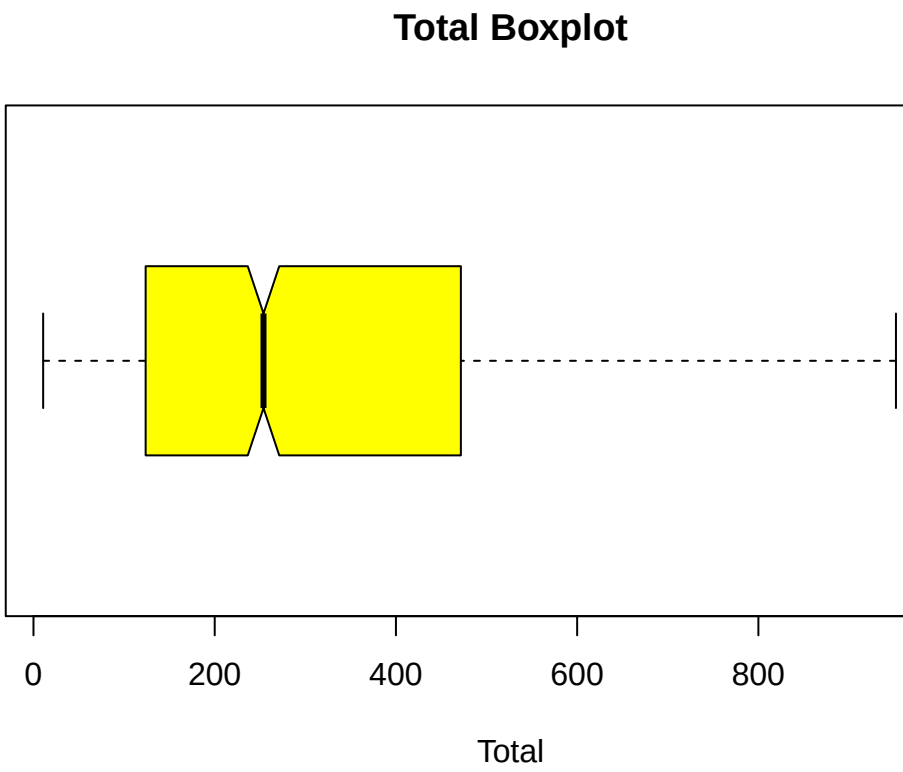
Boxplot for "gross.income" column

- There are outliers in the "gross.income" column. I will keep them in my analysis because they represent true values in the data.



Boxplot for "Rating" column

- There are no outliers in the "Rating" column.



Boxplot for "Total" column

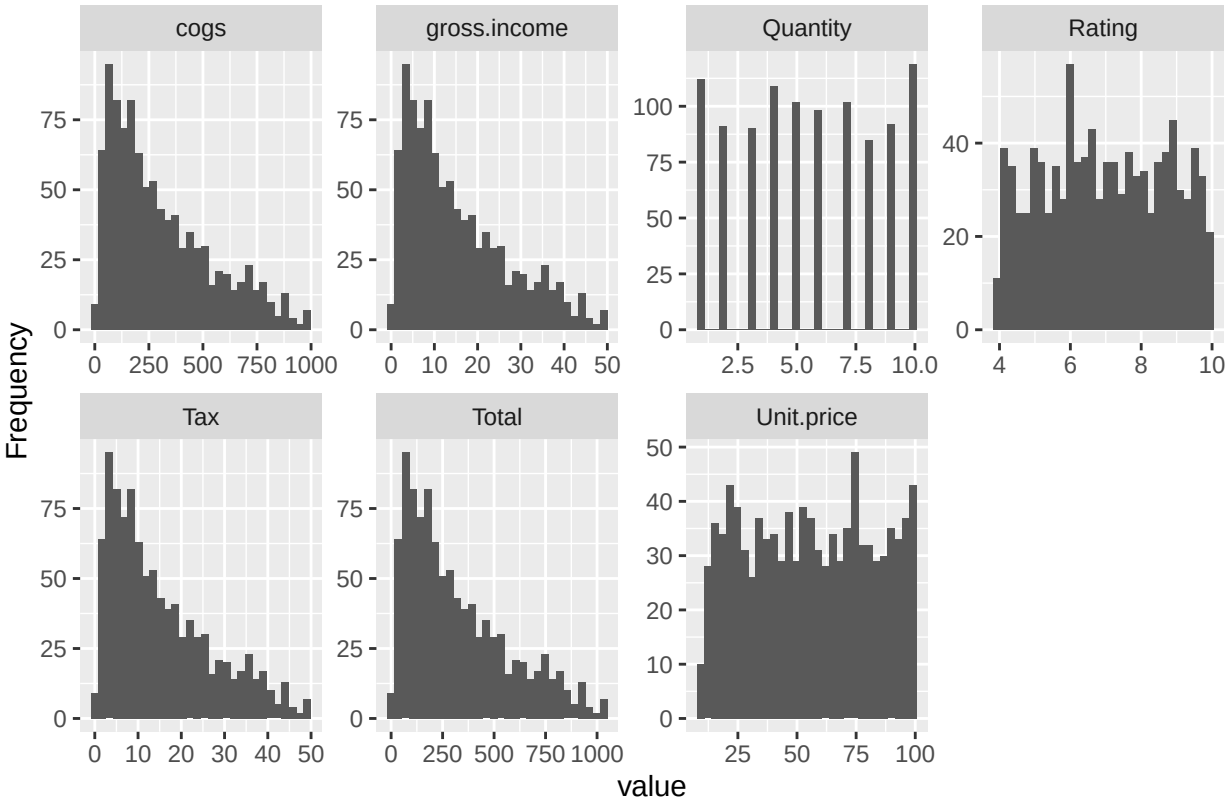
- There are outliers in the "Total" column. I will keep them in my analysis because they represent true values in the data.

4. Exploaratory Data Analysis.

4.1 Univariate Analysis.

Distributions

```
plot_histogram(df)
```



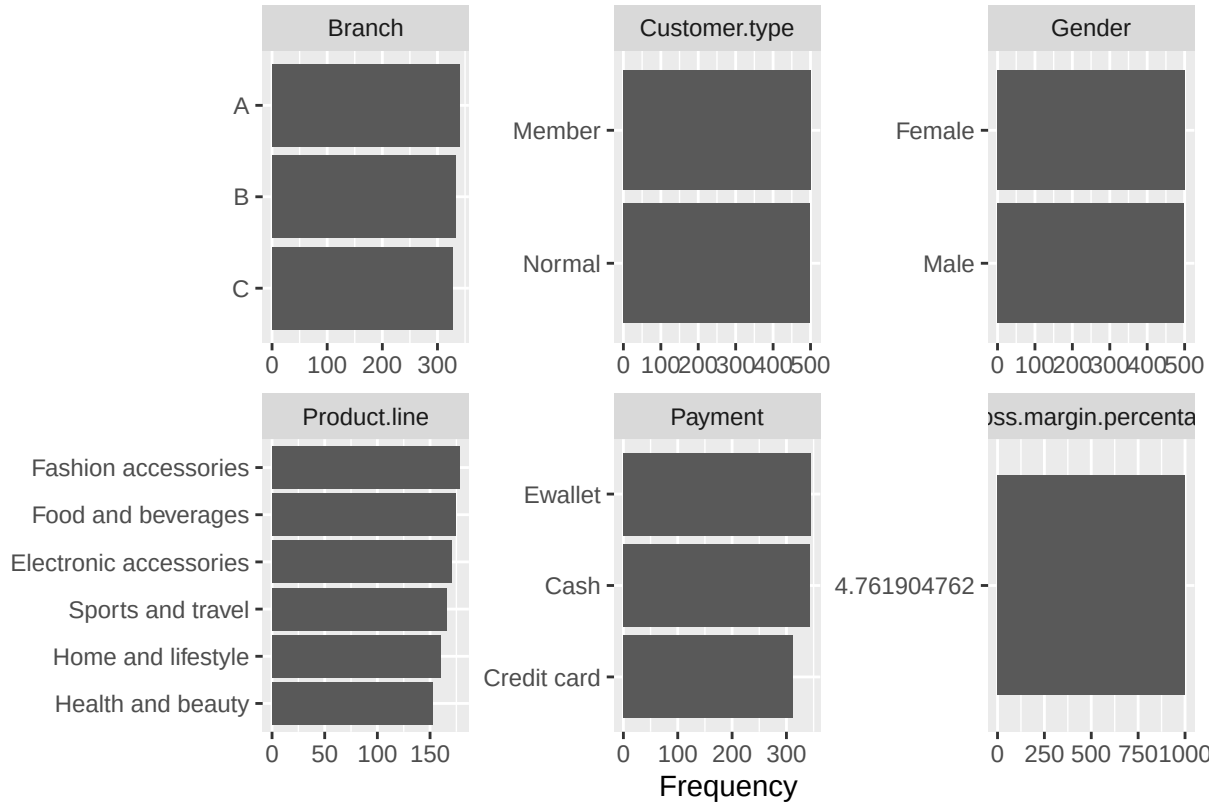
Histograms:

- From the histograms, we get the following insights:
  - Cogs, gross.income, Tax and Total columns are positively skewed, meaning we expect the mean will be greater than the median.
  - Unit.price and Rating columns have fairly even distribution.

```
plot_bar(df)
```

Bar Plots:

```
## 3 columns ignored with more than 50 categories.
## Invoice.ID: 1000 categories
## Date: 89 categories
## Time: 506 categories
```



- From the bar plots, we observe that Branch, Customer.type, Gender, Product.line and Payment columns have an even distribution.

Description and Summary of Data:

- Description:

```
describe(df)
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

##                          vars    n   mean     sd median trimmed    mad   min
## Invoice.ID*                 1 1000 500.50 288.82 500.50  500.50 370.65  1.00
## Branch*                     2 1000   1.99   0.82   2.00    1.99   1.48  1.00
## Customer.type*              3 1000   1.50   0.50   1.00    1.50   0.00  1.00
## Gender*                     4 1000   1.50   0.50   1.00    1.50   0.00  1.00
## Product.line*               5 1000   3.45   1.72   3.00    3.44   1.48  1.00
## Unit.price                  6 1000  55.67  26.49  55.23   55.62  33.37 10.08
## Quantity                    7 1000   5.51   2.92   5.00    5.51   2.97  1.00
## Tax                         8 1000  15.38  11.71  12.09   14.00  11.13  0.51
## Date                        9 1000    NaN     NA     NA     NaN     NA   Inf
## Time*                      10 1000 252.18 147.07 249.00  252.49 190.51  1.00
```

```
## Payment*                   11 1000   2.00   0.83   2.00    2.00   1.48  1.00
## cogs                       12 1000 307.59 234.18 241.76  279.91 222.65 10.17
## gross.margin.percentage    13 1000   4.76   0.00   4.76    4.76   0.00  4.76
## gross.income               14 1000  15.38  11.71  12.09   14.00  11.13  0.51
## Rating                     15 1000   6.97   1.72   7.00    6.97   2.22  4.00
## Total                      16 1000 322.97 245.89 253.85  293.91 233.78 10.68
##                              max   range skew kurtosis   se
## Invoice.ID*              1000.00  999.00 0.00    -1.20 9.13
## Branch*                     3.00    2.00 0.02    -1.51 0.03
## Customer.type*              2.00    1.00 0.00    -2.00 0.02
## Gender*                     2.00    1.00 0.00    -2.00 0.02
## Product.line*               6.00    5.00 0.06    -1.28 0.05
## Unit.price                 99.96   89.88 0.01    -1.22 0.84
## Quantity                   10.00    9.00 0.01    -1.22 0.09
## Tax                        49.65   49.14 0.89    -0.09 0.37
## Date                        -Inf    -Inf   NA       NA   NA
## Time*                     506.00  505.00 0.00    -1.25 4.65
## Payment*                    3.00    2.00 0.00    -1.55 0.03
## cogs                      993.00  982.83 0.89    -0.09 7.41
## gross.margin.percentage     4.76    0.00  NaN      NaN 0.00
## gross.income               49.65   49.14 0.89    -0.09 0.37
## Rating                     10.00    6.00 0.01    -1.16 0.05
## Total                    1042.65 1031.97 0.89    -0.09 7.78
```

- Summary:

```
summary(df)
```

```
##        Invoice.ID  Branch  Customer.type    Gender
##  101-17-6199:  1   A:340   Member:501    Female:501
##  101-81-4070:  1   B:332   Normal:499    Male  :499
##  102-06-2002:  1   C:328
##  102-77-2261:  1
##  105-10-6182:  1
##  105-31-1824:  1
##  (Other)    :994
##                  Product.line   Unit.price       Quantity          Tax
##  Electronic accessories:170   Min.   :10.08   Min.   : 1.00   Min.   : 0.5085
##  Fashion accessories   :178   1st Qu.:32.88   1st Qu.: 3.00   1st Qu.: 5.9249
##  Food and beverages    :174   Median :55.23   Median : 5.00   Median :12.0880
##  Health and beauty     :152   Mean   :55.67   Mean   : 5.51   Mean   :15.3794
##  Home and lifestyle    :160   3rd Qu.:77.94   3rd Qu.: 8.00   3rd Qu.:22.4453
##  Sports and travel     :166   Max.   :99.96   Max.   :10.00   Max.   :49.6500
##
##       Date                Time                   Payment         cogs
##  Min.   :2020-01-01   Length:1000        Cash       :344   Min.   : 10.17
##  1st Qu.:2020-01-24   Class :character   Credit card:311   1st Qu.:118.50
##  Median :2020-02-13   Mode  :character   Ewallet    :345   Median :241.76
##  Mean   :2020-02-14                                       Mean   :307.59
##  3rd Qu.:2020-03-08                                       3rd Qu.:448.90
##  Max.   :2020-03-30                                       Max.   :993.00
##
##  gross.margin.percentage  gross.income         Rating           Total
```

```
##  Min.   :4.762          Min.   : 0.5085   Min.   : 4.000   Min.   :  10.68
##  1st Qu.:4.762          1st Qu.: 5.9249   1st Qu.: 5.500   1st Qu.: 124.42
##  Median :4.762          Median :12.0880   Median : 7.000   Median : 253.85
##  Mean   :4.762          Mean   :15.3794   Mean   : 6.973   Mean   : 322.97
##  3rd Qu.:4.762          3rd Qu.:22.4453   3rd Qu.: 8.500   3rd Qu.: 471.35
##  Max.   :4.762          Max.   :49.6500   Max.   :10.000   Max.   :1042.65
##
```

A function to get the mode

```
# a function for code
mode <- function(x){
  uniqx <- unique(x)
  uniqx[which.max(tabulate(match(x, uniqx)))]
}
```

Unit.price Column

- From the summary and description, we can gather the following about the Unit.price column:
  - Mean: 55.67
  - Median: 55.23
  - Skewness: 0.01
  - Kurtosis: -1.22
- The mode is:

```
mode(df$Unit.price)
```

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

Quantity Column

- From the summary and description, we can gather the following about the Quantity column:
  - Mean: 5.51
  - Median: 5.00
  - Skewness: 0.01
  - Kurtosis: -1.22
- The mode is:

```
mode(df$Quantity)
```

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

Tax Column

- From the summary and description, we can gather the following about the Tax column:
  - Mean: 15.3794
  - Median: 12.0880
  - Skewness: 0.89
  - Kurtosis: -0.09
- The mode is:

```
mode(df$Tax)
```

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

Cogs column

- From the summary and description, we can gather the following about the Cogs column:
  - Mean: 307.59
  - Median: 241.76
  - Skewness: 0.89
  - Kurtosis: -0.09
- The mode is:

```
mode(df$cogs)
```

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

gross.income column

- From the summary and description, we can gather the following about the gross.income column:
  - Mean: 15.3794
  - Median: 12.0880
  - Skewness: 0.89
  - Kurtosis: -0.09
- The mode is:

```
mode(df$gross.income)
```

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

Rating column

- From the summary and description, we can gather the following about the Rating column:
  - Mean: 6.973
  - Median: 7.000
  - Skewness: 0.01
  - Kurtosis: -1.16
- The mode is:

```
mode(df$Rating)
```

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

Total column

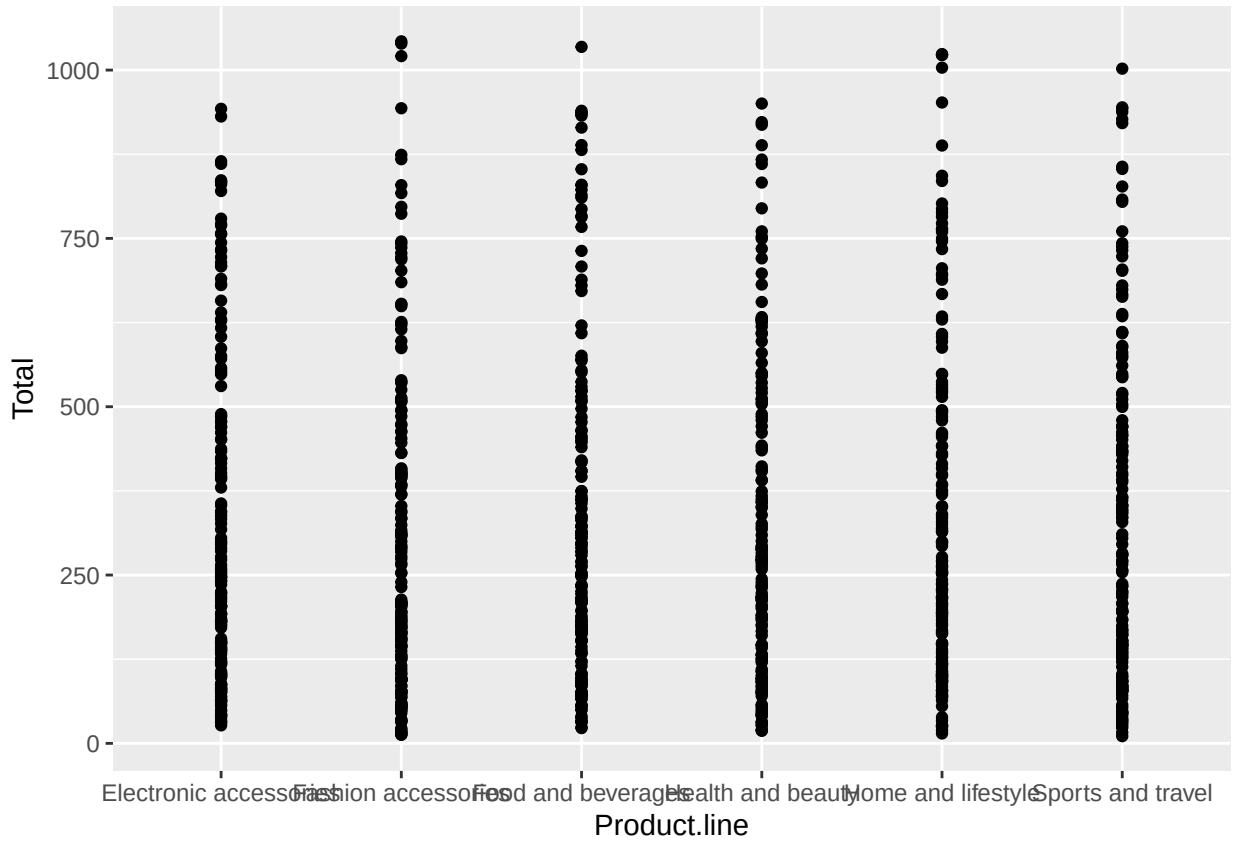
- From the summary and description, we can gather the following about the Total column:
  - Mean: 322.97
  - Median: 253.85
  - Skewness: 0.89
  - Kurtosis: -0.09
- The mode is:

```
mode(df$Total)
```

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

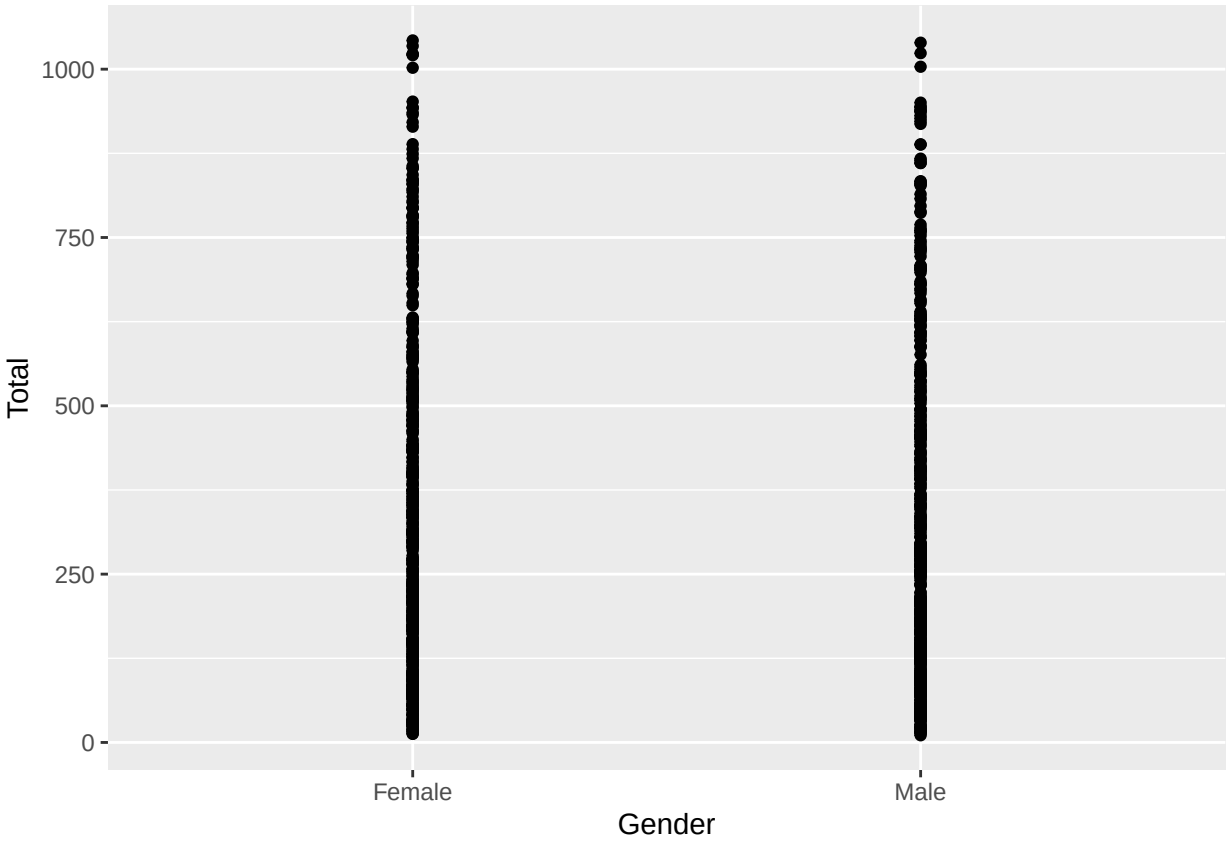
4.2 Bivariate Analysis

```
ggplot(df, aes(x=Product.line, y=Total)) +
  geom_point()
```



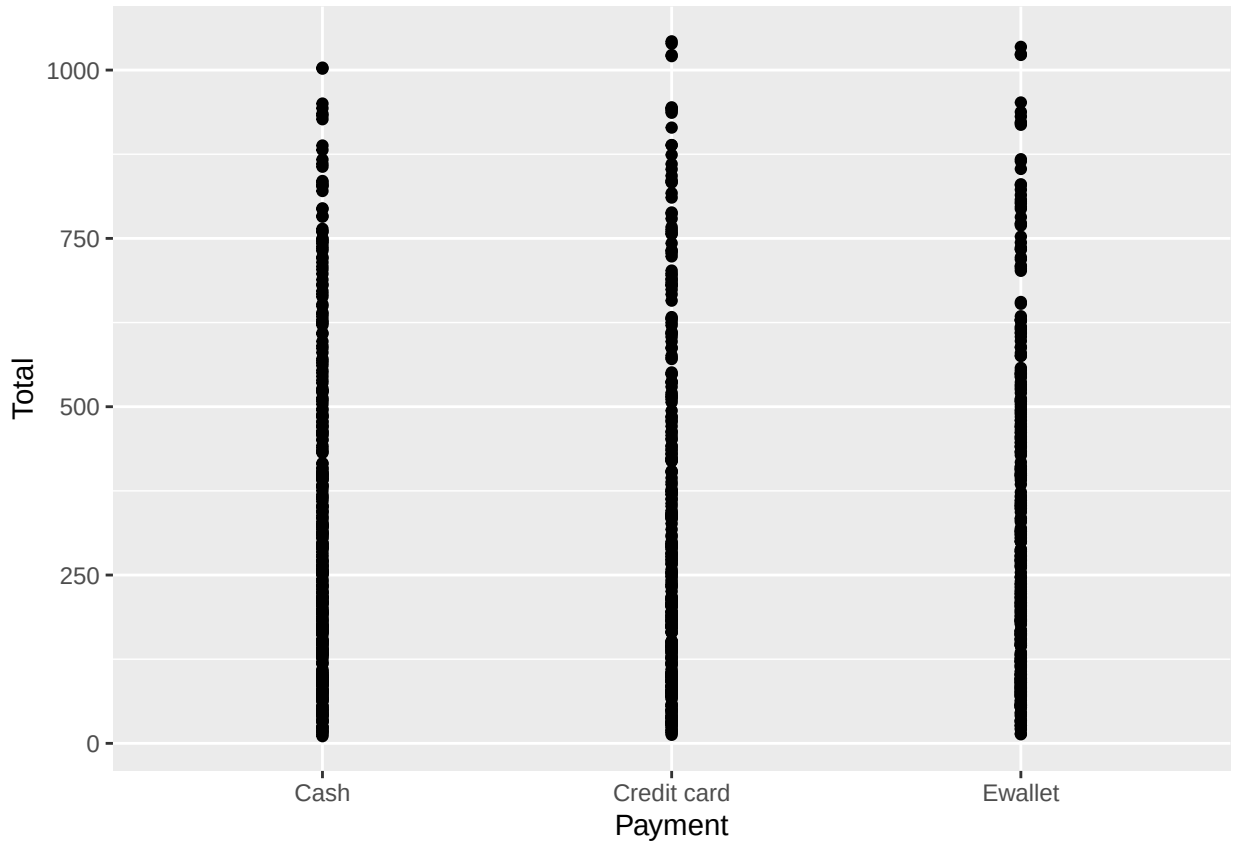
- Fashion Accessories have the highest Total prices while health and beauty products have lower prices.

```
ggplot(df ,aes(Gender, Total)) +
  geom_point()
```



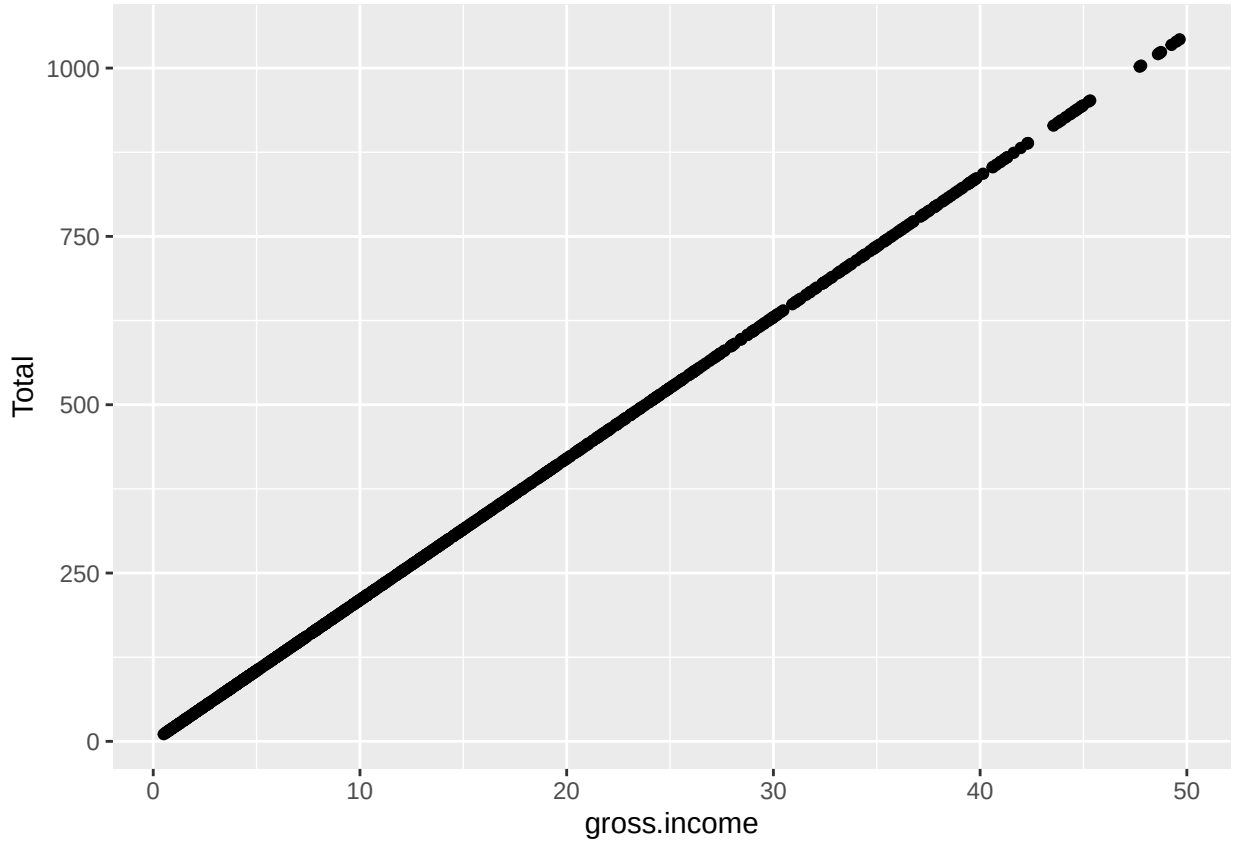
- Total Price is equally distributed in terms of gender

```
ggplot(df, aes(Payment, Total)) +
  geom_point()
```



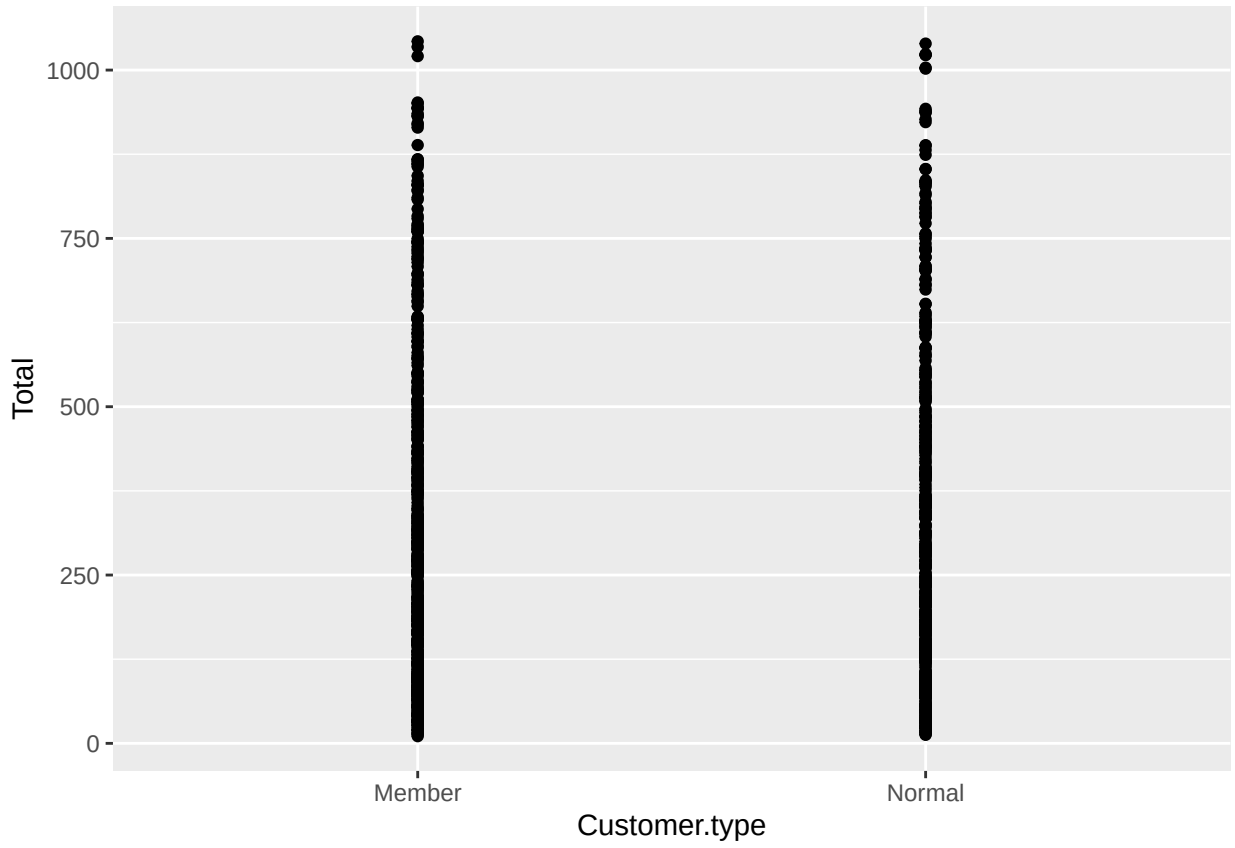
- The payment methods are nearly identical for the total prices of items at checkouts, with Credit card payments being used for more expensive transactions.

```
ggplot(df, aes(gross.income, Total)) +
  geom_point()
```



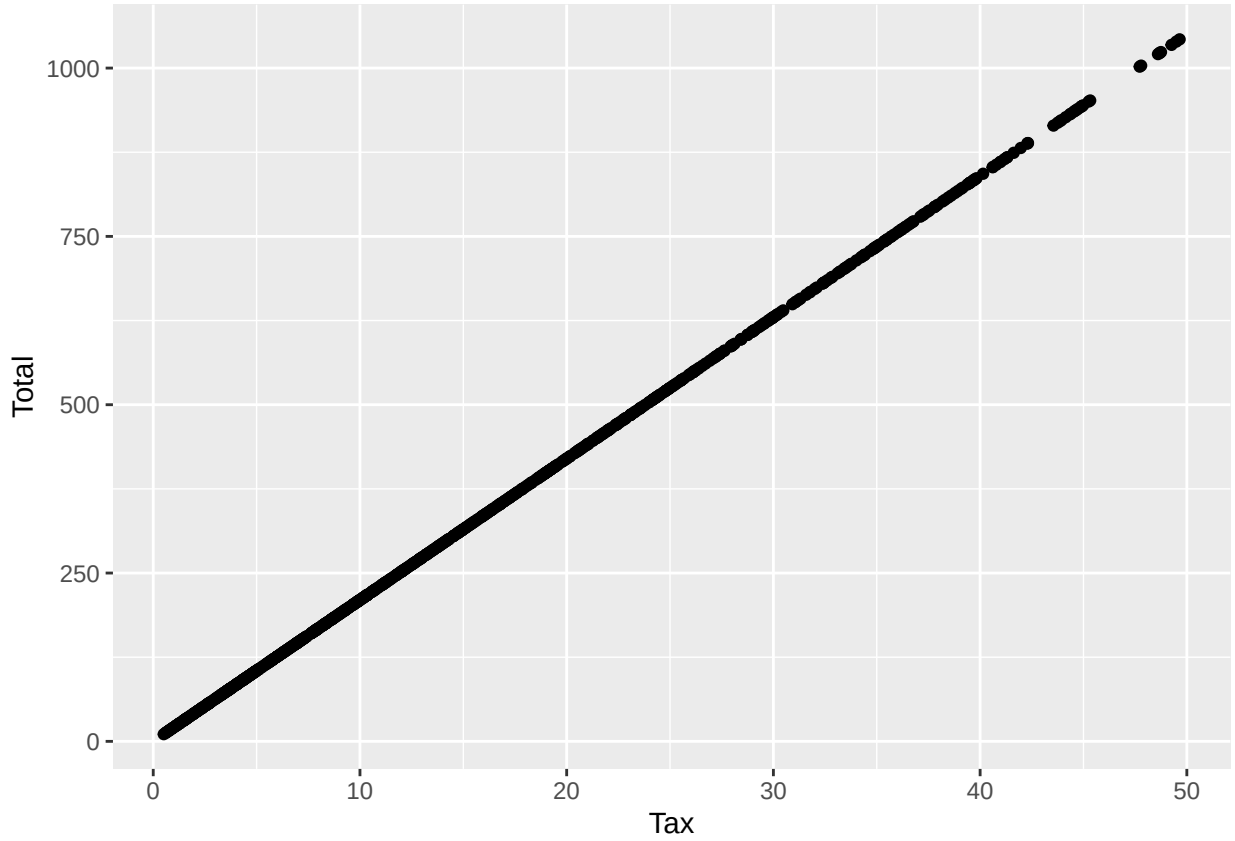
- There is a positive linear relationship between the total at checkout and the consumers gross income.

```
ggplot(df, aes(Customer.type , Total)) +
  geom_point()
```



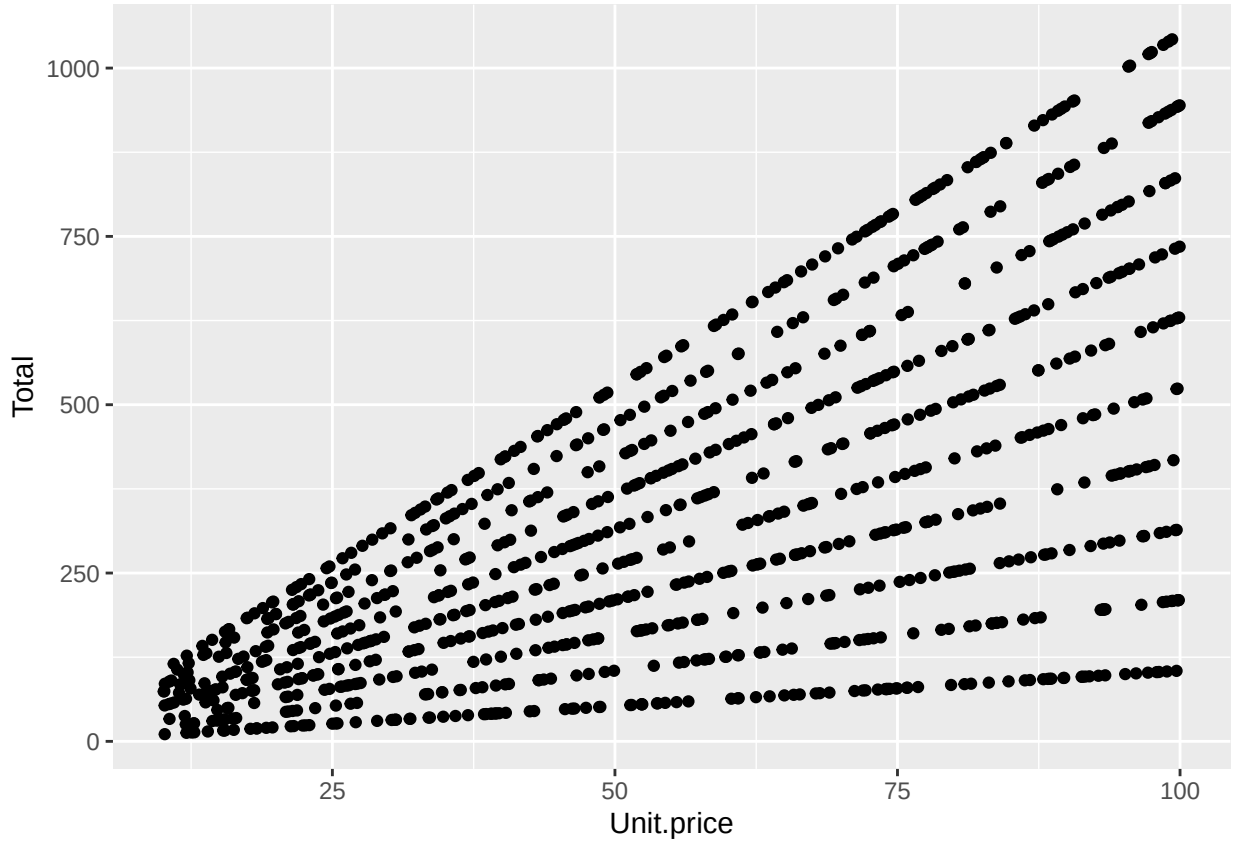
- Members and non members have a nearly equal distribution in expenditure with Members having no breaks in prices.

```
ggplot(df, aes(Tax, Total)) +
  geom_point()
```



- There is a positive linear relationship between tax and total price. As expected, the higher the tax on items, the more they cost.

```
ggplot(df, aes(Unit.price, Total)) +
  geom_point()
```



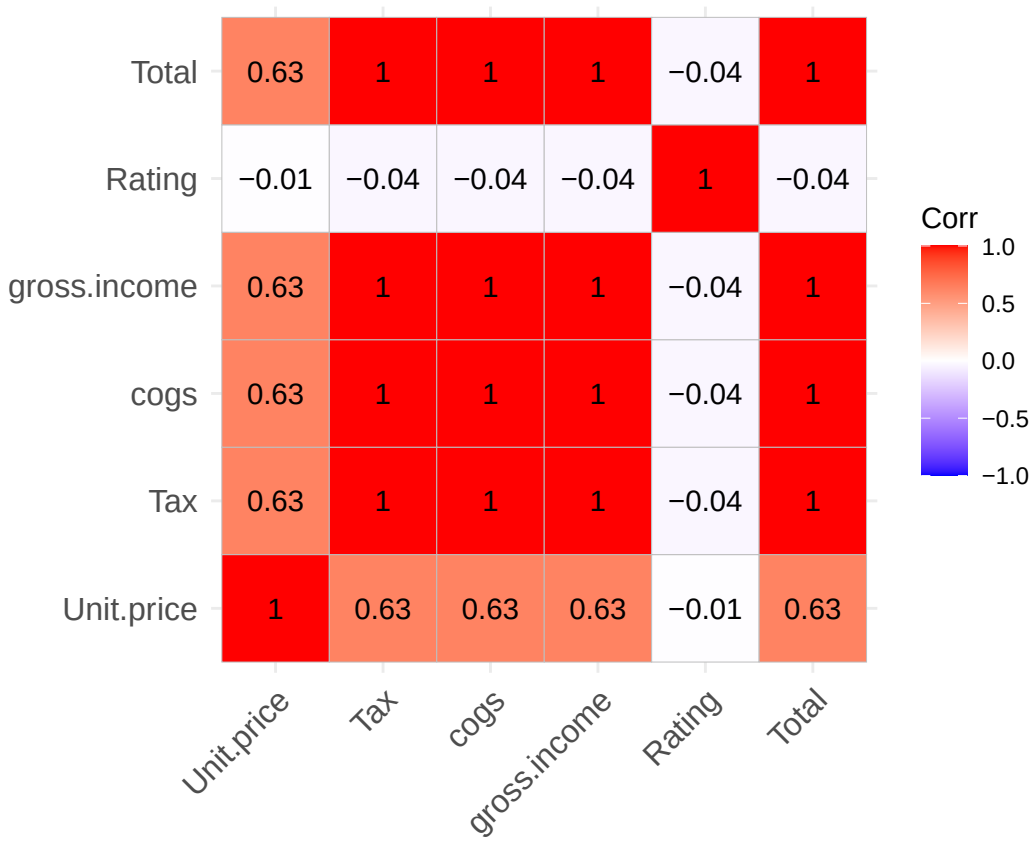
- There are several positive linear relationships with the Unit Price variable: the higher it is the higher the total price is.

Heatmap

```
# Heat map
# Checking the relationship between the variables


# Using Numeric variables only
numeric_tbl <- df %>%
  select_if(is.numeric) %>%
  select(Unit.price, Tax, cogs, gross.income, Rating, Total)

# Calculate the correlations
corr <- cor(numeric_tbl, use = "complete.obs")
ggcorrplot(round(corr, 2),
           type = "full", lab = T)
```



- The following variables show strong correlation:
  - Unit Price with Tax, cogs, gross.income and Total.Strong correlation of 0.63.
  - Tax with cogs, gross.income and Total. Very strong correlation of 1.
  - gross.income to Unit.price, Tax, cogs and Total. Very strong correlation of 1.

5. Part 1: Dimensionality Reduction

- This section of the project entails reducing the dataset to a low dimensional dataset using the t-SNE algorithm or PCA. I will perform analysis and provide insights gained.

```
# Label Encoding branch column and storing in a copy
branch <- LabelEncoder.fit(df$Branch)
df$Branch <- transform(branch, factor(df$Branch))

# Label Encoding Gender column and storing in a copy
gender <- LabelEncoder.fit(df$Gender)
df$Gender <- transform(gender, factor(df$Gender))

# Label Encoding Customer.type column and storing in a copy
customer <- LabelEncoder.fit(df$Customer.type)
df$Customer.type <- transform(customer, factor(df$Customer.type))

# Label Encoding product.line column and storing in a copy
```

```
product <- LabelEncoder.fit(df$Product.line)
df$Product.line <- transform(product, factor(df$Product.line))

# Label Encoding payment column and storing in a copy
pay <- LabelEncoder.fit(df$Payment)
df$Payment <- transform(pay, factor(df$Payment))
```

```
# for plotting

colors = rainbow(length(unique(df$Total)))
names(colors) = unique(df$Total)
```

```
# Executing the algorithm on curated data
model <- Rtsne(df, dims=2, perplexity=30, verbose= TRUE, max_iter=500)
```

```
## Performing PCA
## Read the 1000 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.22 seconds (sparsity = 0.103032)!
## Learning embedding...
## Iteration 50: error is 61.747332 (50 iterations in 0.22 seconds)
## Iteration 100: error is 54.912890 (50 iterations in 0.13 seconds)
## Iteration 150: error is 53.915607 (50 iterations in 0.14 seconds)
## Iteration 200: error is 53.503042 (50 iterations in 0.14 seconds)
## Iteration 250: error is 53.305316 (50 iterations in 0.14 seconds)
## Iteration 300: error is 0.750867 (50 iterations in 0.13 seconds)
## Iteration 350: error is 0.563777 (50 iterations in 0.13 seconds)
## Iteration 400: error is 0.510818 (50 iterations in 0.13 seconds)
## Iteration 450: error is 0.491542 (50 iterations in 0.12 seconds)
## Iteration 500: error is 0.482684 (50 iterations in 0.13 seconds)
## Fitting performed in 1.41 seconds.
```

```
# getting the duration of the execution

exeTimeTsne <- system.time(Rtsne(df, dims = 2, perplexity=30, verbose=TRUE, max_iter = 500))
```

```
## Performing PCA
## Read the 1000 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.23 seconds (sparsity = 0.103032)!
## Learning embedding...
## Iteration 50: error is 62.877945 (50 iterations in 0.18 seconds)
## Iteration 100: error is 54.728343 (50 iterations in 0.15 seconds)
## Iteration 150: error is 53.752615 (50 iterations in 0.15 seconds)
## Iteration 200: error is 53.483152 (50 iterations in 0.14 seconds)
## Iteration 250: error is 53.357962 (50 iterations in 0.15 seconds)
```

```
## Iteration 300: error is 0.722726 (50 iterations in 0.14 seconds)
## Iteration 350: error is 0.543779 (50 iterations in 0.13 seconds)
## Iteration 400: error is 0.500963 (50 iterations in 0.13 seconds)
## Iteration 450: error is 0.487666 (50 iterations in 0.13 seconds)
## Iteration 500: error is 0.478725 (50 iterations in 0.15 seconds)
## Fitting performed in 1.44 seconds.
```

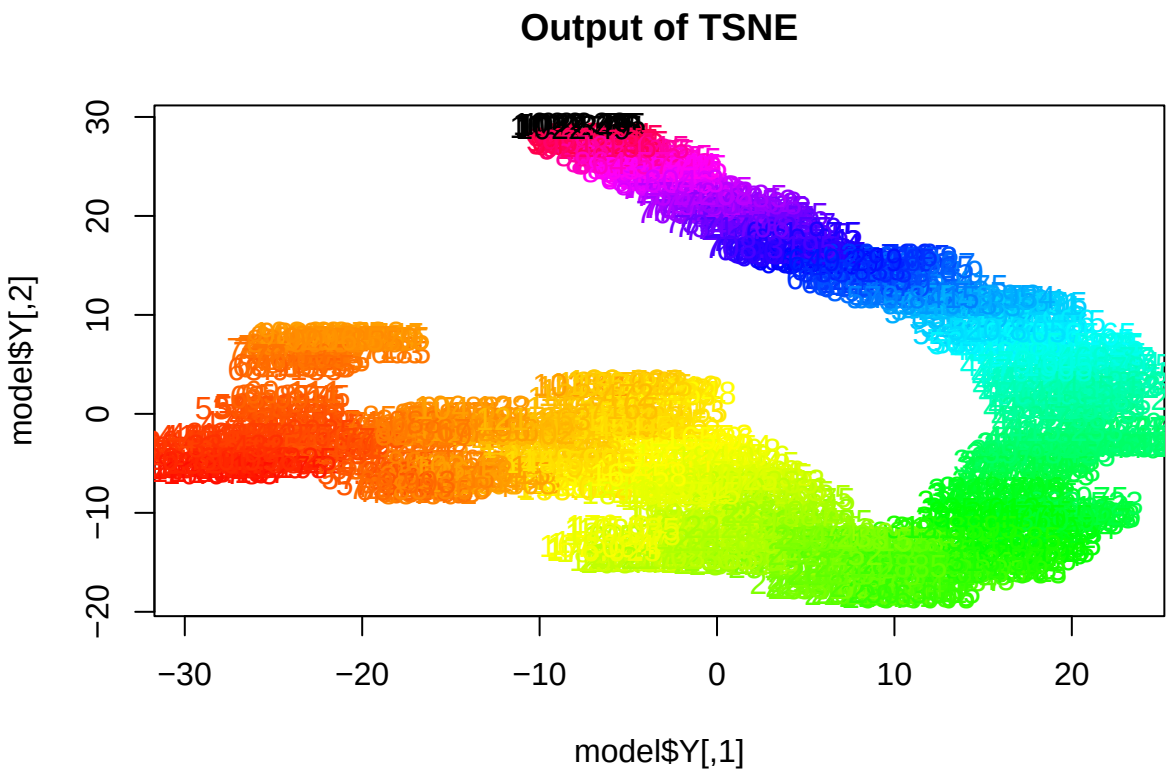
```
summary(model)
```

```
##                     Length Class  Mode
## N                      1   -none- numeric
## Y                   2000   -none- numeric
## costs               1000   -none- numeric
## itercosts             10   -none- numeric
## origD                  1   -none- numeric
## perplexity             1   -none- numeric
## theta                  1   -none- numeric
## max_iter               1   -none- numeric
## stop_lying_iter        1   -none- numeric
## mom_switch_iter        1   -none- numeric
## momentum               1   -none- numeric
## final_momentum         1   -none- numeric
## eta                    1   -none- numeric
## exaggeration_factor    1   -none- numeric
```

```
head(model$Y)
```

```
##            [,1]       [,2]
## [1,]  13.181539  10.336149
## [2,] -17.401304  -1.925566
## [3,]  17.941413 -11.292040
## [4,]  17.454094   6.141992
## [5,]   7.908610  14.902107
## [6,]   9.429351  15.967652
```

```
plot(model$Y, t='n', main="Output of TSNE")
text(model$Y, labels=df$Total, col=colors[df$Total] )
```



6. Part 2: Feature Selection

6.1 Filter Methods

```
# Using Numeric variables only
numeric_table <- df %>%
  select_if(is.numeric) %>%
  select(Unit.price, Tax, cogs, gross.income, Rating, Total)
```

```
corrMat <- cor(numeric_table)

# highly correlated features
high <- findCorrelation(corrMat, cutoff = 0.75)

# names of highly correlated features
names(numeric_table[, high])
```

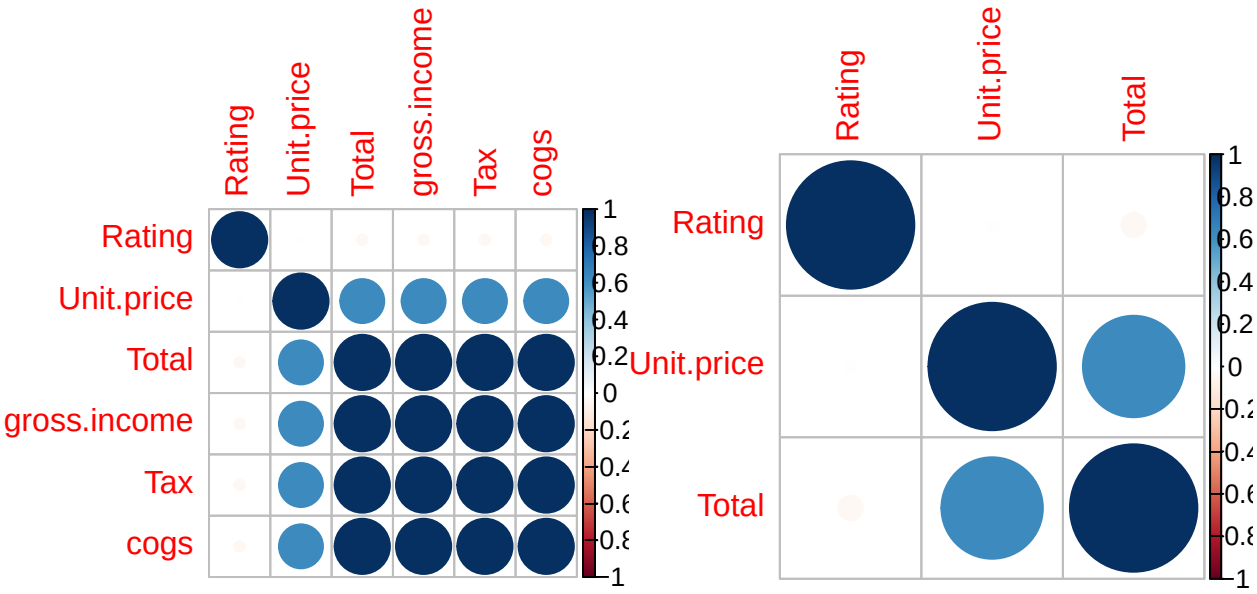
```
## [1] "Tax"          "cogs"         "gross.income"
```

```
#  Removing Tax, cogs and gross.income
numeric_table2 <- df %>%
  select_if(is.numeric) %>%
  select(Unit.price, Rating, Total)
```

```
# data set without highly correlated variables
c2 <- numeric_table[-high]

# plotting
par(mfrow = c(1, 2))

corrplot(corrMat, order = "hclust")
corrplot(cor(c2), order = "hclust")
```



6.2 Feature Ranking

```
# From the FSelector package, we use the correlation coefficient as a unit of valuation.
# This would be one of the several algorithms contained
# in the FSelector package that can be used rank the variables.
# ---
#
Scores <- linear.correlation(Total~.,numeric_table)
Scores
```

```
##              attr_importance
## Unit.price         0.6339621
## Tax                1.0000000
## cogs               1.0000000
```

```
## gross.income       1.0000000
## Rating             0.0364417
```

```
# From the output above, we observe a list containing
# rows of variables on the left and score on the right.
# In order to make a decision, we define a cutoff
# i.e. suppose we want to use the top 5 representative variables,
# through the use of the cutoff.k function included in the FSelector package.
# Alternatively, we could define our cutoff visually
# but in cases where there are few variables than in high dimensional datasets.
#
# cutoff.k: The algorithms select a subset from a ranked attributes.
# ---
#
Subset <- cutoff.k(Scores, 4)
as.data.frame(Subset)
```

```
##         Subset
## 1          Tax
## 2         cogs
## 3 gross.income
## 4   Unit.price
```

```
# We could also set cutoff as a percentage which would indicate
# that we would want to work with the percentage of the best variables.
# ---
#
Subset2 <-cutoff.k.percent(Scores, 0.4)
as.data.frame(Subset2)
```

```
##   Subset2
## 1     Tax
## 2    cogs
```

```
# Instead of using the scores for the correlation coefficient,
# we can use an entropy - based approach as shown below;
# ---
#
Scores2 <- information.gain(Total~., numeric_table)

# Choosing Variables by cutoffSubset <- cutoff.k(Scores2, 5)
# ---
#
Subset3 <- cutoff.k(Scores2, 5)
as.data.frame(Subset3)
```

```
##        Subset3
## 1          Tax
## 2         cogs
## 3 gross.income
## 4   Unit.price
## 5       Rating
```

7. Conclusion

- Using Feature Ranking method with information gain of all variables being used as a metric of comparison, the Branch, Customer Type, Gender, Product Line and Unit Price columns would be the best to use for modeling a regressor with respect to Rating.