Week 14 - Dimensionality Reduction

Wilkister Mbaka

2022-07-29

CarreFour Marketing - Dimensionality Reduction

Definining The Question

Specifying the Question

1. This section of the project entails reducing your dataset to a low dimensional dataset using the t-SNE algorithm or PCA.
2. You will be required to perform your analysis and provide insights gained from your analysis.

Metric of success

- Importing the data
- Cleaning the data
- performing a thorough EDA
- Performing Dimensionality Reduction

Data relevance

The data has been provided by the supermarket itself

Understanding the context

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). Your project has been divided into four parts where you'll explore a recent marketing dataset by performing various unsupervised learning techniques and later providing recommendations based on your insights.

Experimental design

The experimental design will involve the following steps:

- Dealing with missing values.
- Dropping variables of low variance.
- Use of decision trees to tackle missing values, outliers and identifying significant variables.
- Use of random forest to select a smaller subset of input features.
- Using the Pearson correlation matrix to identify and later drop variables with high correlation.
- Performing backward feature elimination.
- Performing factor analysis to group high correlated variables.
- Using Principal Component Analysis (PCA).

Reading The Data

```
# Importing Libraries
library (tidyr)
library(naniar)
library (ggplot2)
library (e1071)
library (corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(factoextra)
```

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

```
library(NbClust)
library(superml)
```

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

```
#installing packages
library(data.table)
#
#Loading the dataset
df <- fread("http://bit.ly/CarreFourDataset")
```

Checking The Data

```
# Preview the data
head(df)
```

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

```
## 3:      16.2155    7.4 340.5255
## 4:      23.2880    8.4 489.0480
## 5:      30.2085    5.3 634.3785
## 6:      29.8865    4.1 627.6165
```

```r
# Preview the data
tail(df)
```

```
##     Invoice ID Branch Customer type Gender           Product line Unit price
## 1: 652-49-6720      C        Member Female Electronic accessories      60.95
## 2: 233-67-5758      C        Normal   Male      Health and beauty      40.35
## 3: 303-96-2227      B        Normal Female     Home and lifestyle      97.38
## 4: 727-02-1313      A        Member   Male     Food and beverages      31.84
## 5: 347-56-2442      A        Normal   Male     Home and lifestyle      65.82
## 6: 849-09-3807      A        Member Female    Fashion accessories      88.34
##    Quantity     Tax      Date  Time Payment   cogs gross margin percentage
## 1:        1  3.0475 2/18/2019 11:40 Ewallet  60.95                4.761905
## 2:        1  2.0175 1/29/2019 13:46 Ewallet  40.35                4.761905
## 3:       10 48.6900  3/2/2019 17:16 Ewallet 973.80                4.761905
## 4:        1  1.5920  2/9/2019 13:22    Cash  31.84                4.761905
## 5:        1  3.2910 2/22/2019 15:33    Cash  65.82                4.761905
## 6:        7 30.9190 2/18/2019 13:28    Cash 618.38                4.761905
##    gross income Rating     Total
## 1:       3.0475    5.9   63.9975
## 2:       2.0175    6.2   42.3675
## 3:      48.6900    4.4 1022.4900
## 4:       1.5920    7.7   33.4320
## 5:       3.2910    4.1   69.1110
## 6:      30.9190    6.6  649.2990
```

```r
# Dimensionanity of the data
dim(df)
```

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

The dataframe has 1000 rows and 16 columns

Tidying The Dataset

```r
# check the column names
colnames(df)
```

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

```r
# standardize column names with standard naming convention ie lowercase and replace spaces with '_'
# replace the spaces with underscores using gsub() function
names(df) <- gsub(" ","_", names(df))

# The column names have a mixture of uppercase and lowercase charachers we should correct that and
#make all the characters lowercase.
names(df) <- tolower(names(df))
# Confirmation
colnames(df)
```

```
##  [1] "invoice_id"              "branch"
##  [3] "customer_type"           "gender"
##  [5] "product_line"            "unit_price"
##  [7] "quantity"                "tax"
##  [9] "date"                    "time"
## [11] "payment"                 "cogs"
## [13] "gross_margin_percentage" "gross_income"
## [15] "rating"                  "total"
```

```r
# Let us find the datatypes of the data
str(df)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  16 variables:
##  $ invoice_id             : chr  "750-67-8428" "226-31-3081" "631-41-3108" "123-19-1176" ...
##  $ branch                 : chr  "A" "C" "A" "A" ...
##  $ customer_type          : chr  "Member" "Normal" "Normal" "Member" ...
##  $ gender                 : chr  "Female" "Female" "Male" "Male" ...
##  $ product_line           : chr  "Health and beauty" "Electronic accessories" "Home and lifestyle" "
##  $ unit_price             : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ quantity               : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ tax                    : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ date                   : chr  "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ time                   : chr  "13:08" "10:29" "13:23" "20:33" ...
##  $ payment                : chr  "Ewallet" "Cash" "Credit card" "Ewallet" ...
##  $ cogs                   : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ gross_margin_percentage: num  4.76 4.76 4.76 4.76 4.76 ...
##  $ gross_income           : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ rating                 : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
##  $ total                  : num  549 80.2 340.5 489 634.4 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The dataset has character, integer and numerical datatypes Time and date are in the incorrect format

```r
# Change date to date format
df$date <- as.Date(df$date, "%m/%d/%Y")

# Change time to time format

df$time <- as.ITime(df$time)

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
## 1:        7 26.1415 2019-01-05 13:08:00     Ewallet 522.83
## 2:        5  3.8200 2019-03-08 10:29:00        Cash  76.40
## 3:        7 16.2155 2019-03-03 13:23:00 Credit card 324.31
## 4:        8 23.2880 2019-01-27 20:33:00     Ewallet 465.76
## 5:        7 30.2085 2019-02-08 10:37:00     Ewallet 604.17
## 6:        7 29.8865 2019-03-25 18:30:00     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
#Finding the total number of missing values in each column
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

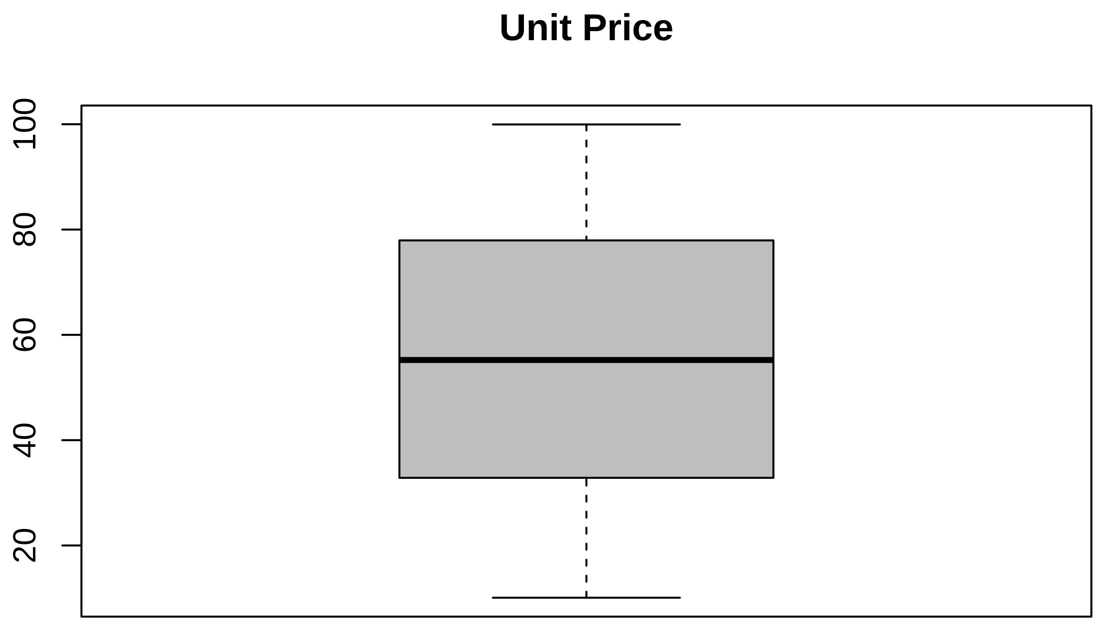
```r
# Cheking for duplicates
df_dup <- df[duplicated(df),]
df_dup
```

```
## Empty data.table (0 rows and 16 cols): invoice_id,branch,customer_type,gender,product_line,unit_pric
```

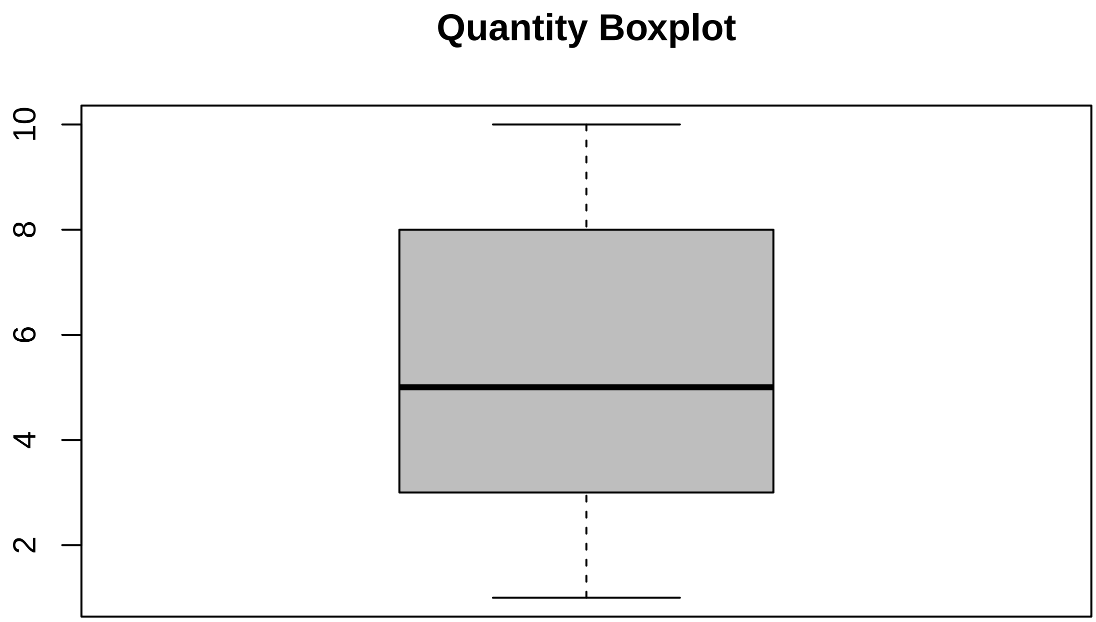
There is no duplicate data in this dataset

Checking for outliers

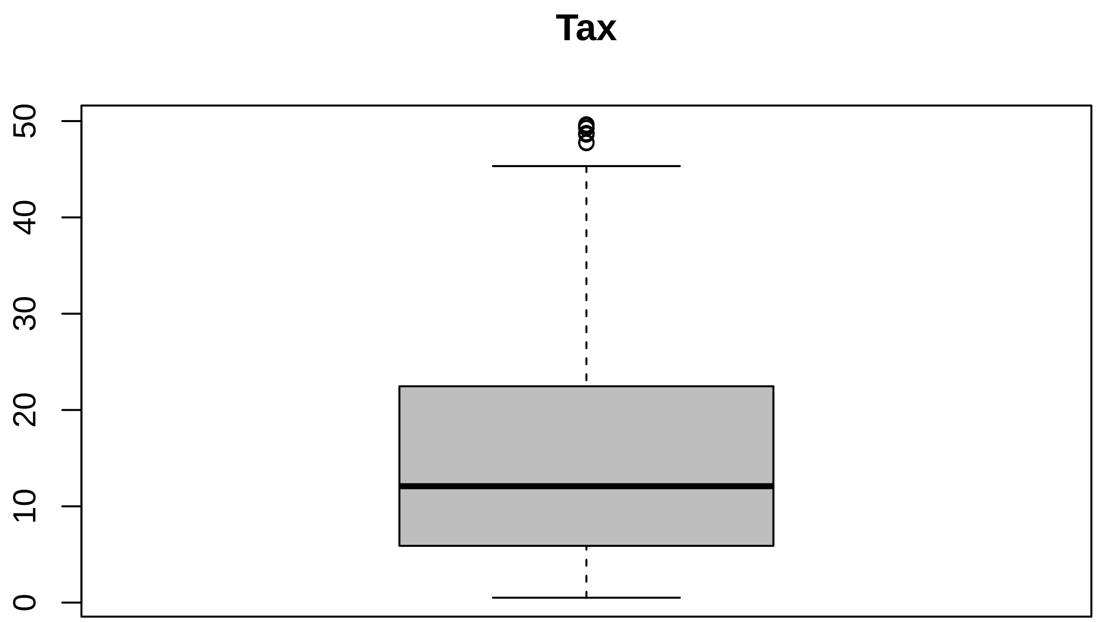
```r
# Plotting boxplots to check for outliers
boxplot(df$unit_price,col='grey', main = 'Unit Price')
```



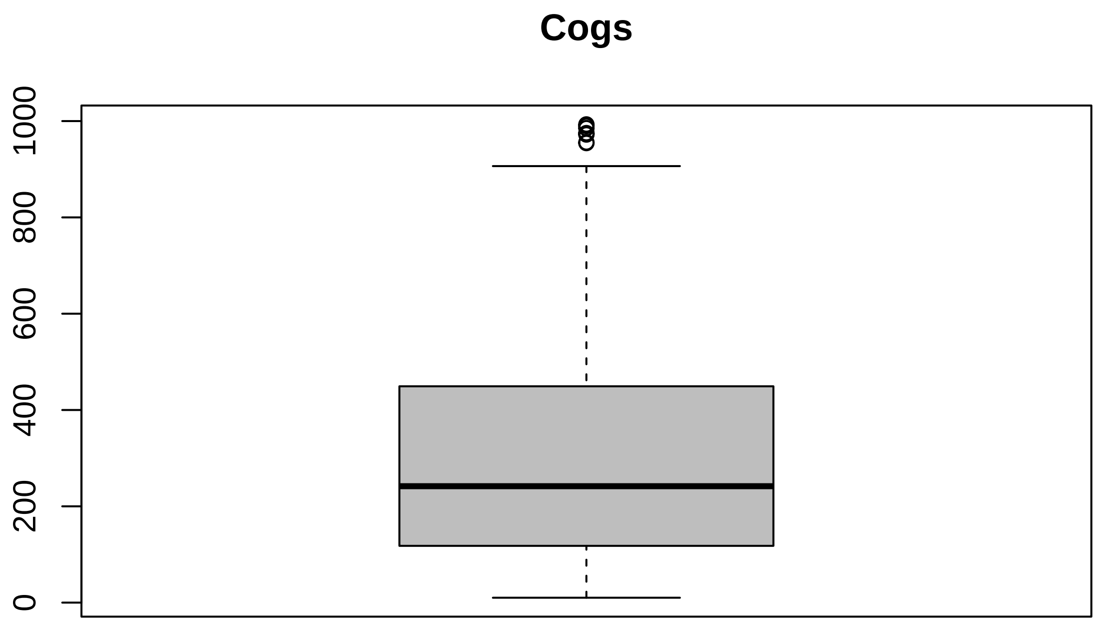

```
boxplot(df$quantity,col='grey', main = 'Quantity Boxplot')
```



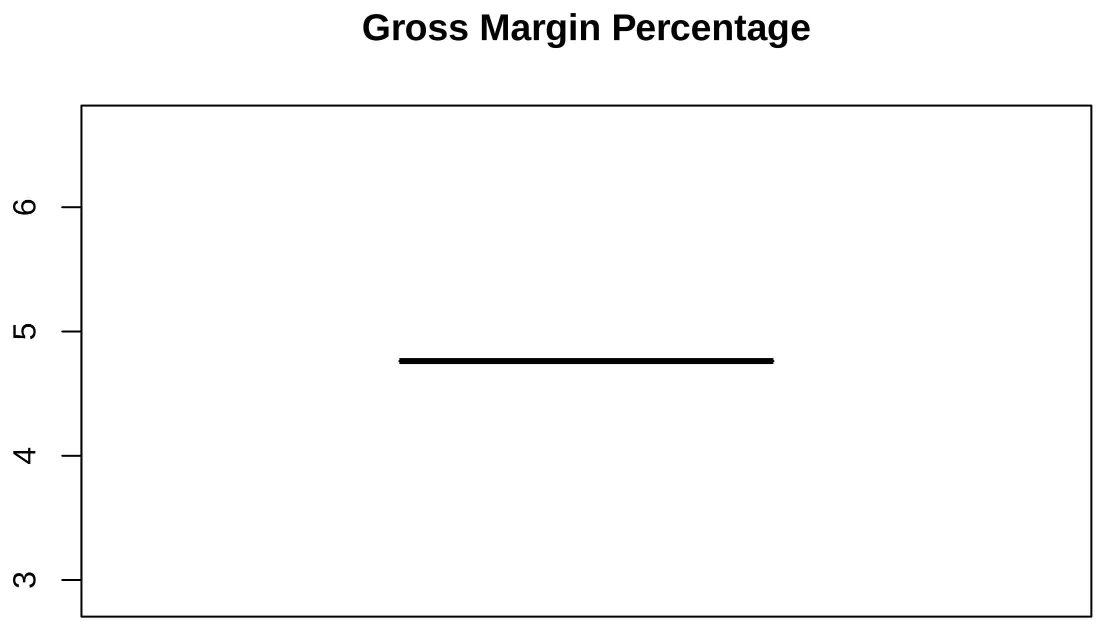

```
boxplot(df$tax,col='grey', main = 'Tax')
```



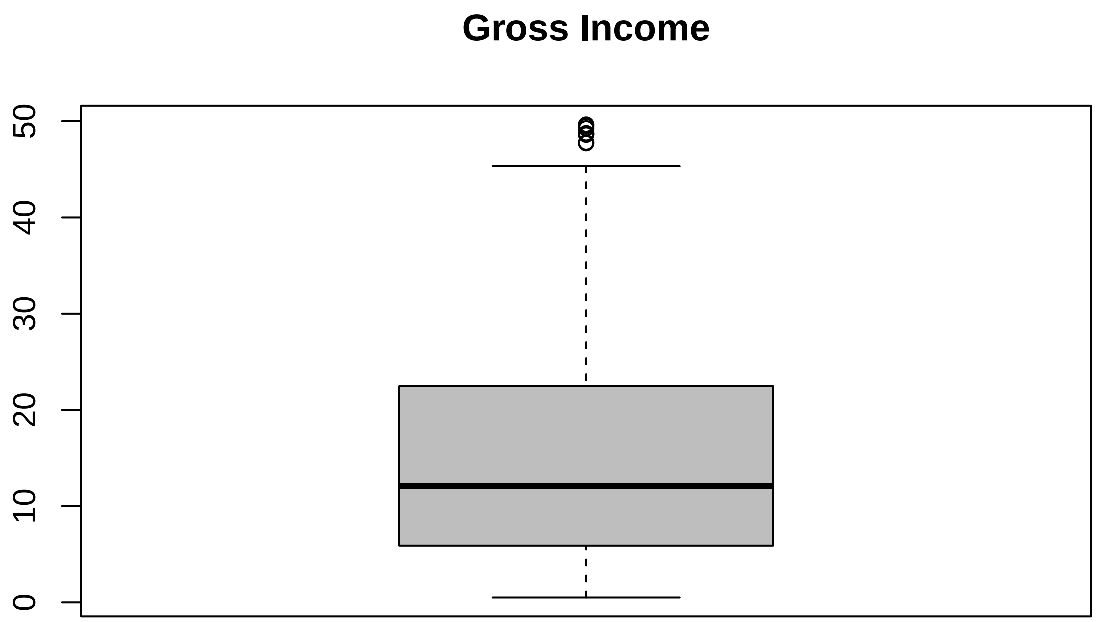

```
boxplot(df$cogs,col='grey', main = 'Cogs')
```



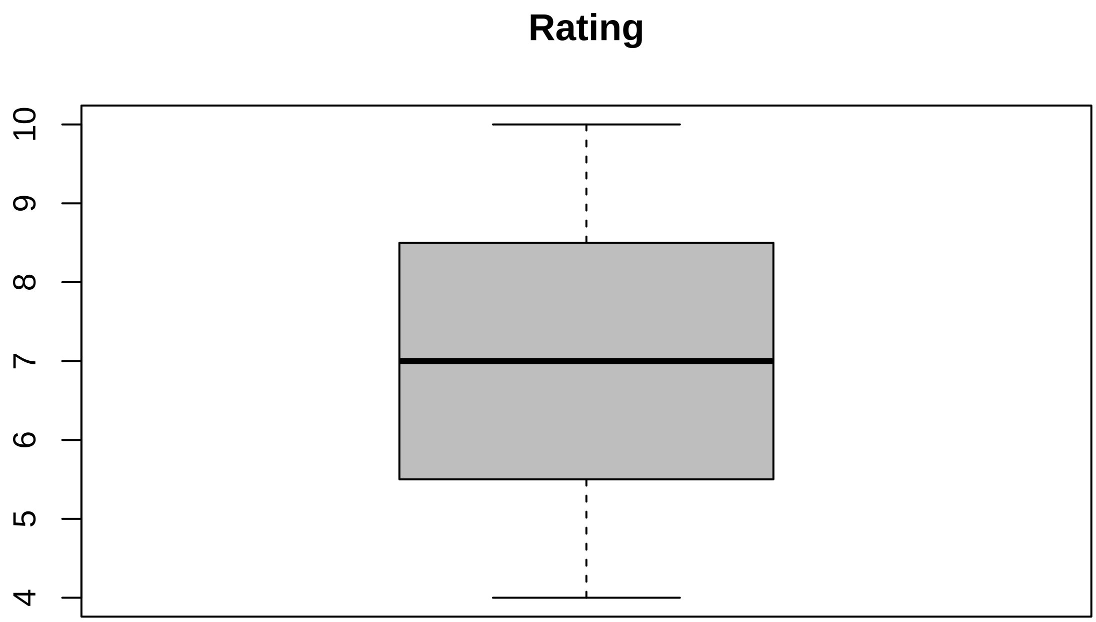

```
boxplot(df$gross_margin_percentage,col='grey', main = 'Gross Margin Percentage')
```



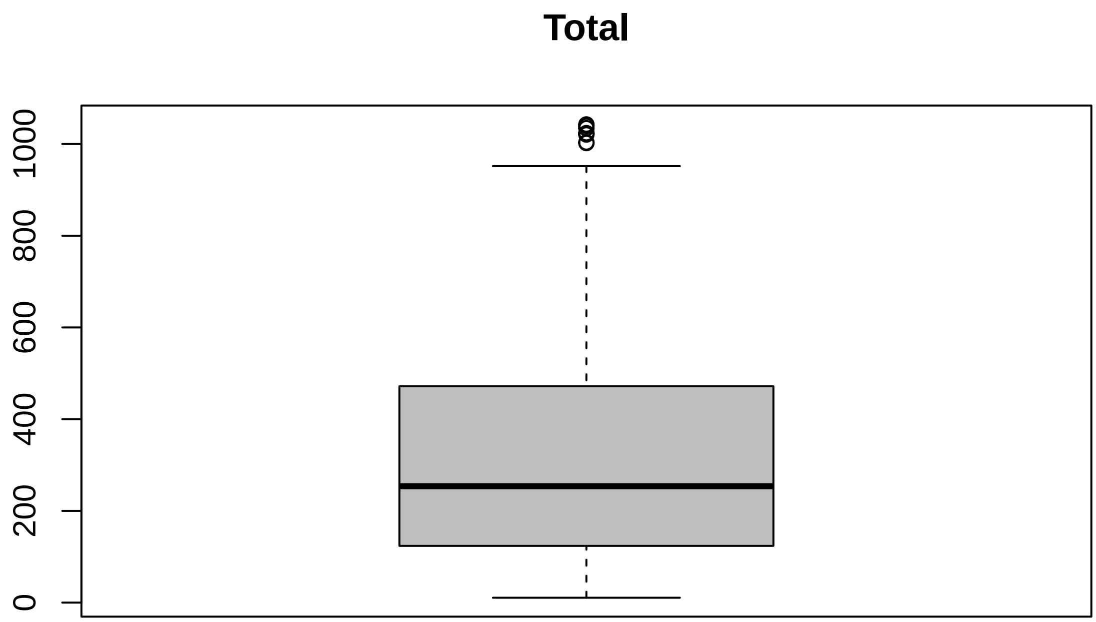

```
boxplot(df$gross_income,col='grey', main = 'Gross Income')
```




```
boxplot(df$rating,col='grey', main = 'Rating')
```




```
boxplot(df$total,col='grey', main = 'Total')
```



Tax, Cogs, Gross Income, Total has some outliers but we will leave them because they are actual representation of the data

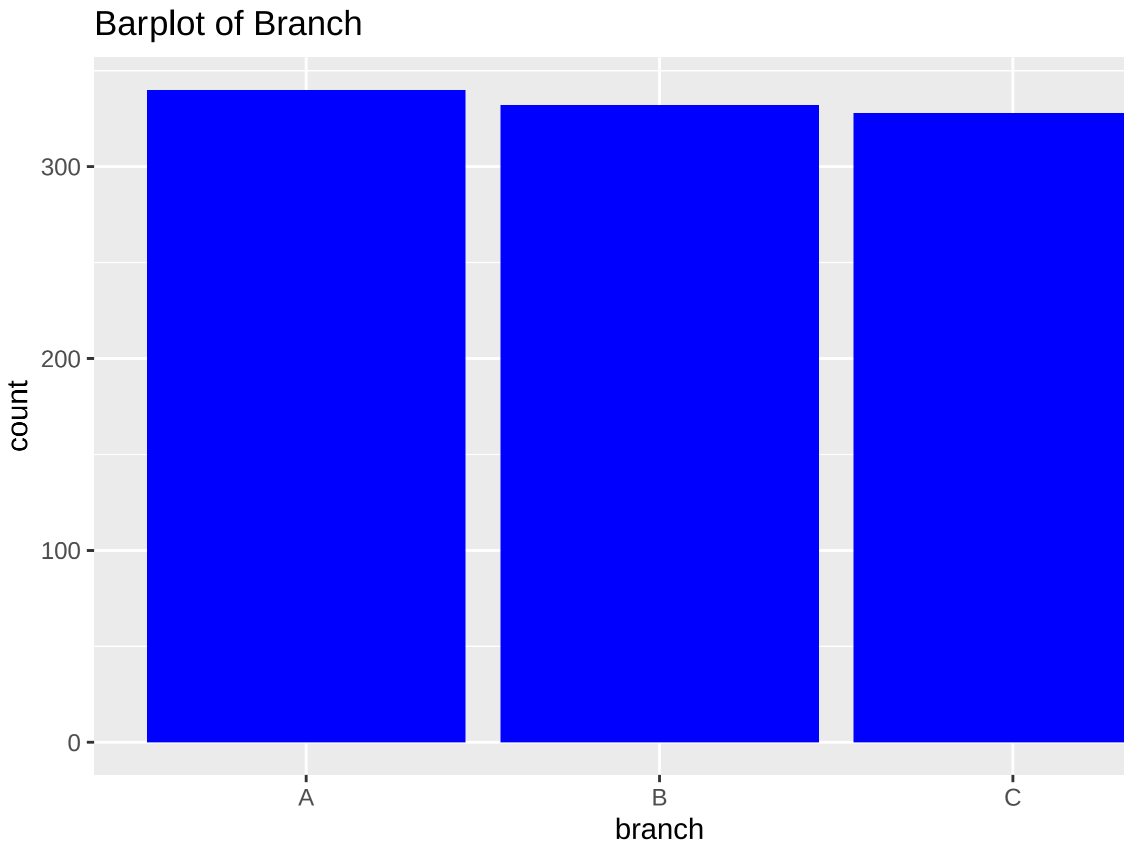
```r
# removing irrelevant column - gross_margin_percentage it has the same amount through out
setDT(df)[, c( 'gross_margin_percentage') := NULL]
# check the dimensions of the dataframe after cleaning
dim(df)
```

```
## [1] 1000   15
```

Exploratory Data Analysis

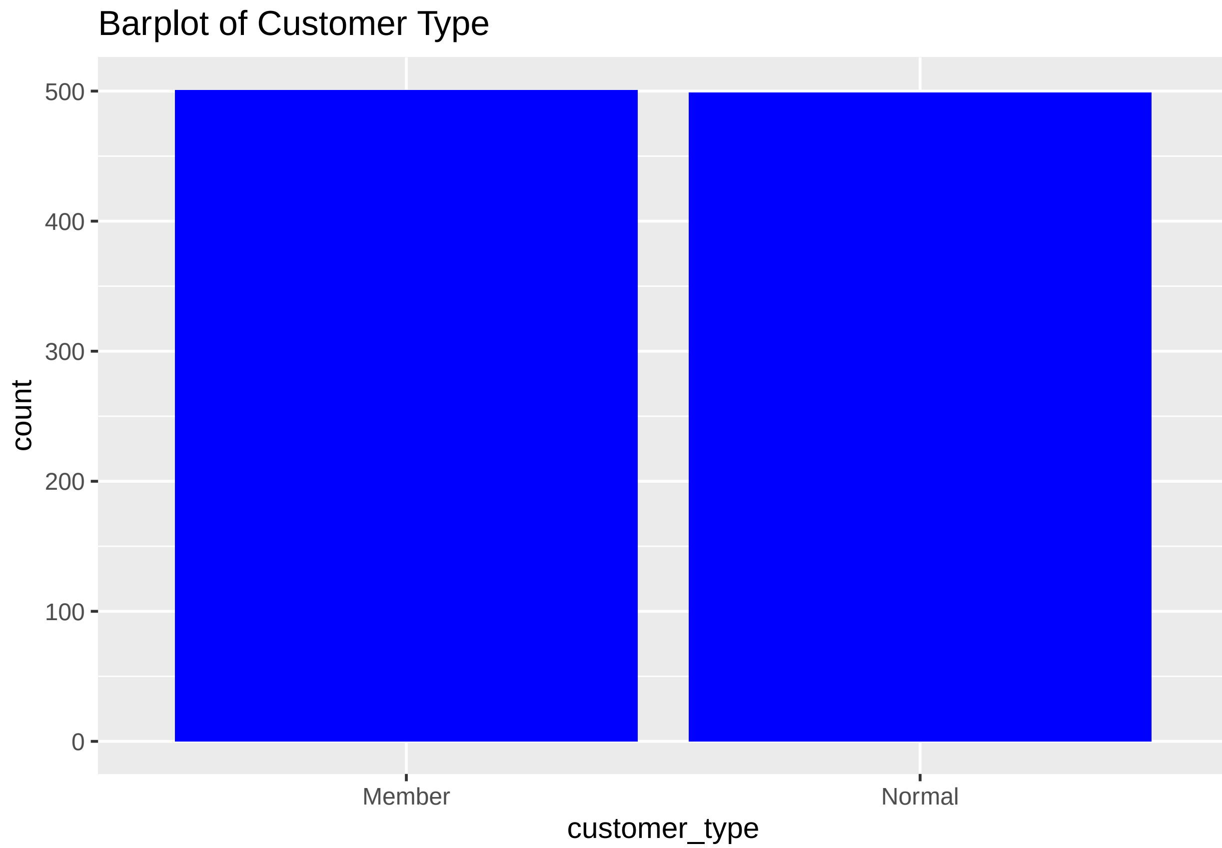
Univariate Analysis

```r
# Frequency of Branch column

ggplot(df, aes(x = branch)) +
  geom_bar(fill="blue") +  ggtitle('Barplot of Branch')
```



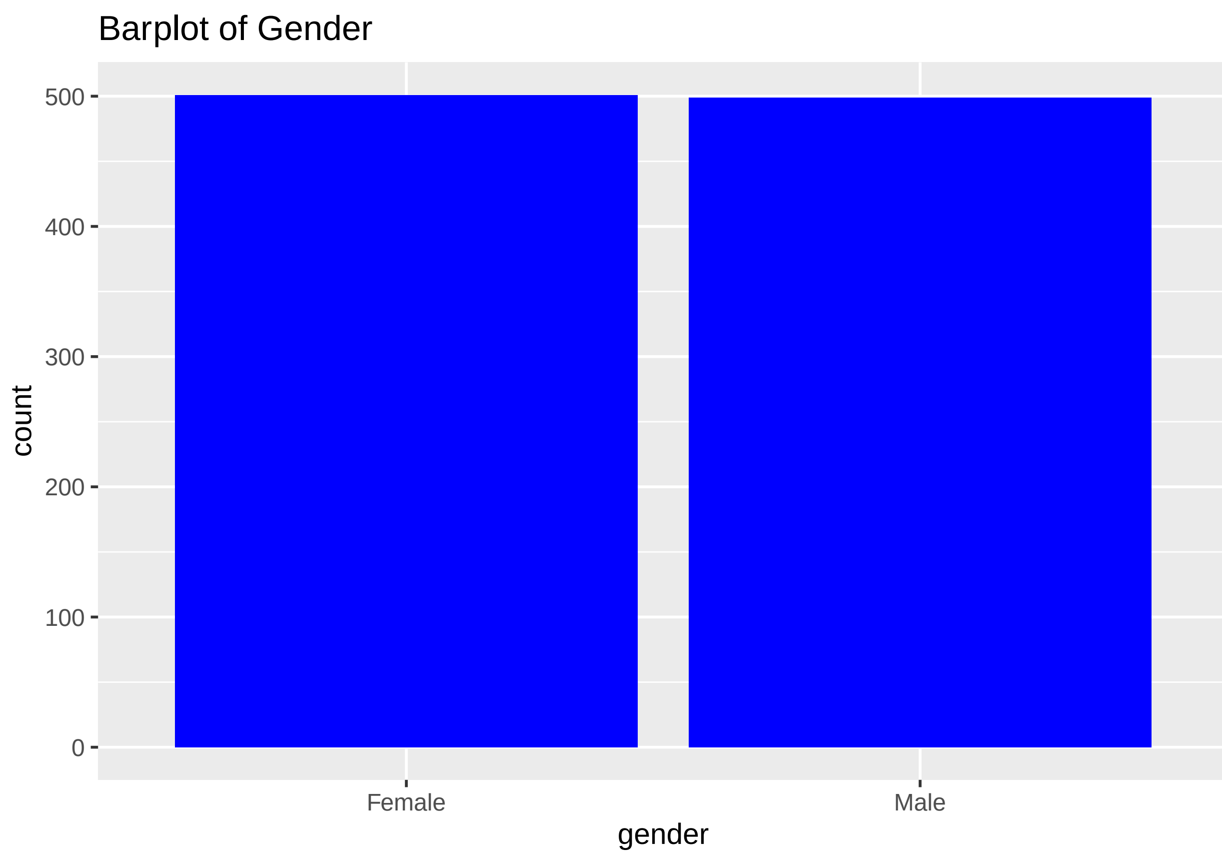
Categorical Variables
The data collected on Branches A is slightly more than branch B and C .

```
# Frequency of Customer Type column

ggplot(df, aes(x = customer_type)) +
  geom_bar(fill="blue") +  ggtitle('Barplot of Customer Type')
```



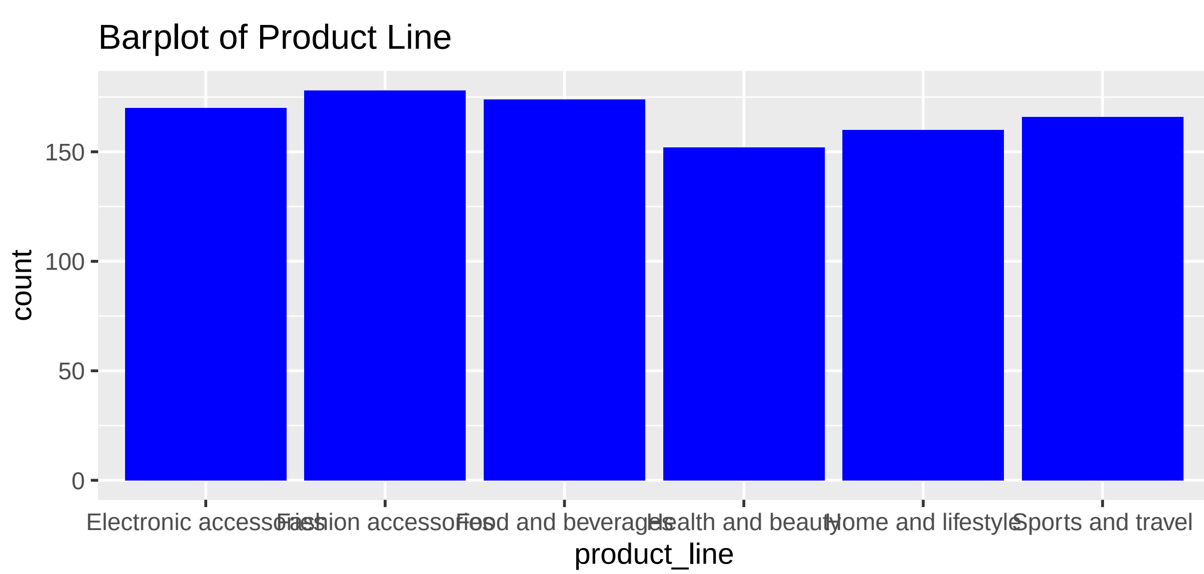
The information collected was half from the members and half from the normal customers.

```
# Frequency of Gender column

ggplot(df, aes(x = gender)) +
  geom_bar(fill="blue") +  ggtitle('Barplot of Gender')
```



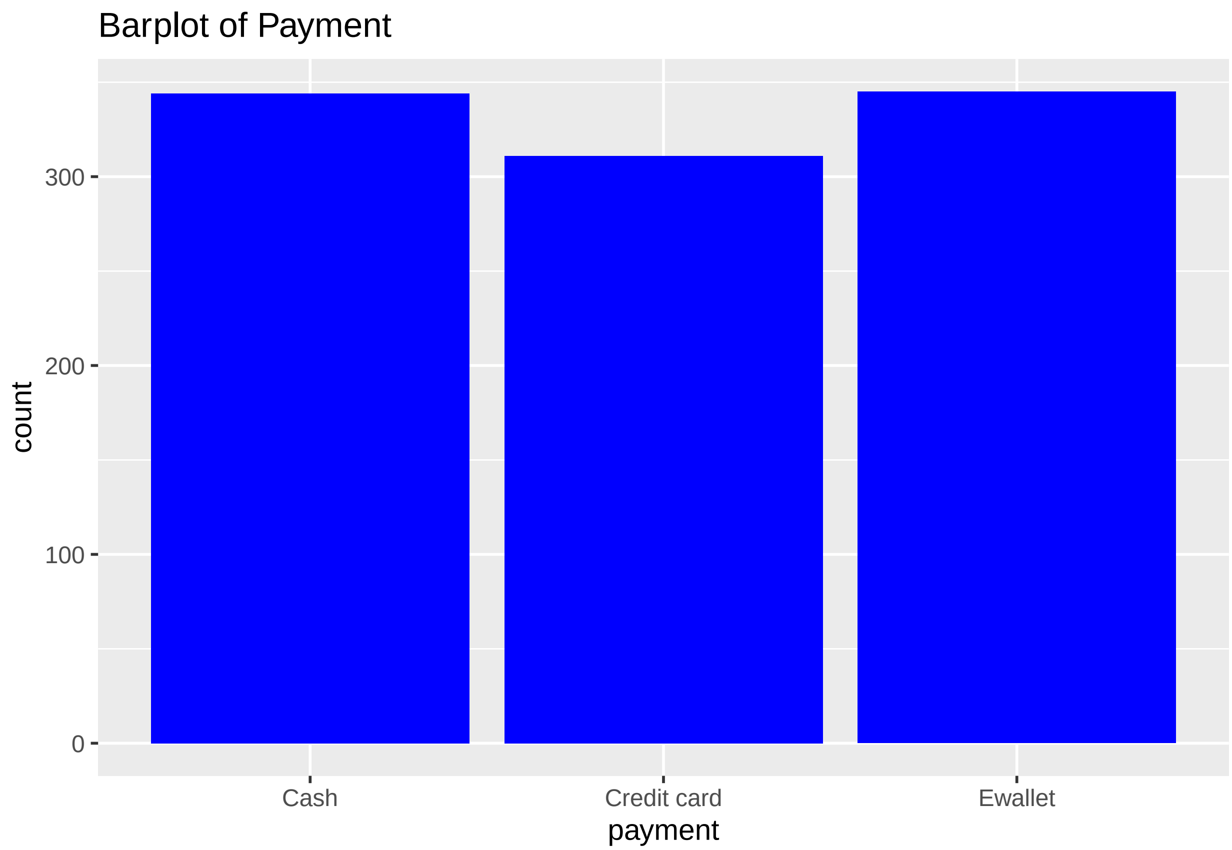
The data from male and female persons is equal

```
# Frequency of Product Line column

ggplot(df, aes(x = product_line)) +
  geom_bar(fill="blue") +  ggtitle('Barplot of Product Line')
```



The most popular product line is Fashion accessories followed by food and beverages

```r
# Frequency of Payment column

ggplot(df, aes(x = payment)) +
  geom_bar(fill="blue") +  ggtitle('Barplot of Payment')
```



Slightly More people paid their bills with E wallet and cash rather than Credit card

```r
# numerical columns.
num_col <- unlist(lapply(df, is.numeric))
df_num <- subset(df, select = num_col)
head (df_num)
```

Numerical Variables

```
##    unit_price quantity     tax     time   cogs gross_income rating    total
## 1:      74.69        7 26.1415 13:08:00 522.83      26.1415    9.1 548.9715
## 2:      15.28        5  3.8200 10:29:00  76.40       3.8200    9.6  80.2200
## 3:      46.33        7 16.2155 13:23:00 324.31      16.2155    7.4 340.5255
## 4:      58.22        8 23.2880 20:33:00 465.76      23.2880    8.4 489.0480
## 5:      86.31        7 30.2085 10:37:00 604.17      30.2085    5.3 634.3785
## 6:      85.39        7 29.8865 18:30:00 597.73      29.8865    4.1 627.6165
```

```r
#Getting the measures of dispersion in the numerical columns.
summary_stats <- data.frame(
  Mean = apply(df_num, 2, mean),
  Median = apply(df_num, 2, median),
  Min = apply(df_num, 2, min),
  Max = apply(df_num, 2, max))
summary_stats
```

```
##                     Mean    Median       Min      Max
## unit_price      55.67213    55.230   10.0800    99.96
## quantity         5.51000     5.000    1.0000    10.00
## tax             15.37937    12.088    0.5085    49.65
## time         55481.88000 55140.000 36000.0000 75540.00
## cogs           307.58738   241.760   10.1700   993.00
## gross_income    15.37937    12.088    0.5085    49.65
## rating           6.97270     7.000    4.0000    10.00
## total          322.96675   253.848   10.6785  1042.65
```

```r
# compute the measures of cenral tendancy and the measures of dispersion of the numerical variables and
library(moments)
```

```
##
## Attaching package: 'moments'

## The following objects are masked from 'package:e1071':
##
##     kurtosis, moment, skewness
```

```r
statistics <- data.frame(
Variance= apply(df_num, 2, var),
Std = apply(df_num, 2, sd),
Skewness = apply(df_num, 2, skewness),
Kurtosis = apply(df_num, 2, kurtosis))
# round off the values to 2 decimal places and display the data1frame
statistics <- round(statistics, 2)
statistics
```

```
##                  Variance      Std Skewness Kurtosis
## unit_price         701.97    26.49     0.01     1.78
## quantity             8.55     2.92     0.01     1.78
## tax                137.10    11.71     0.89     2.91
## time         132058417.28 11491.67     0.02     1.77
## cogs             54838.64   234.18     0.89     2.91
## gross_income       137.10    11.71     0.89     2.91
## rating               2.95     1.72     0.01     1.85
## total            60459.60   245.89     0.89     2.91
```

```r
# Define the function
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
```

```
}

# Mode
mode.unit_price <- getmode(df$unit_price)
mode.unit_price
```

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

```
mode.quantity <- getmode(df$quantity)
mode.quantity
```

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

```
mode.tax <- getmode(df$tax)
mode.tax
```

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

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

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

```
mode.gross_income <- getmode(df$gross_income)
mode.gross_income
```

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

```
mode.rating <- getmode(df$rating)
mode.rating
```

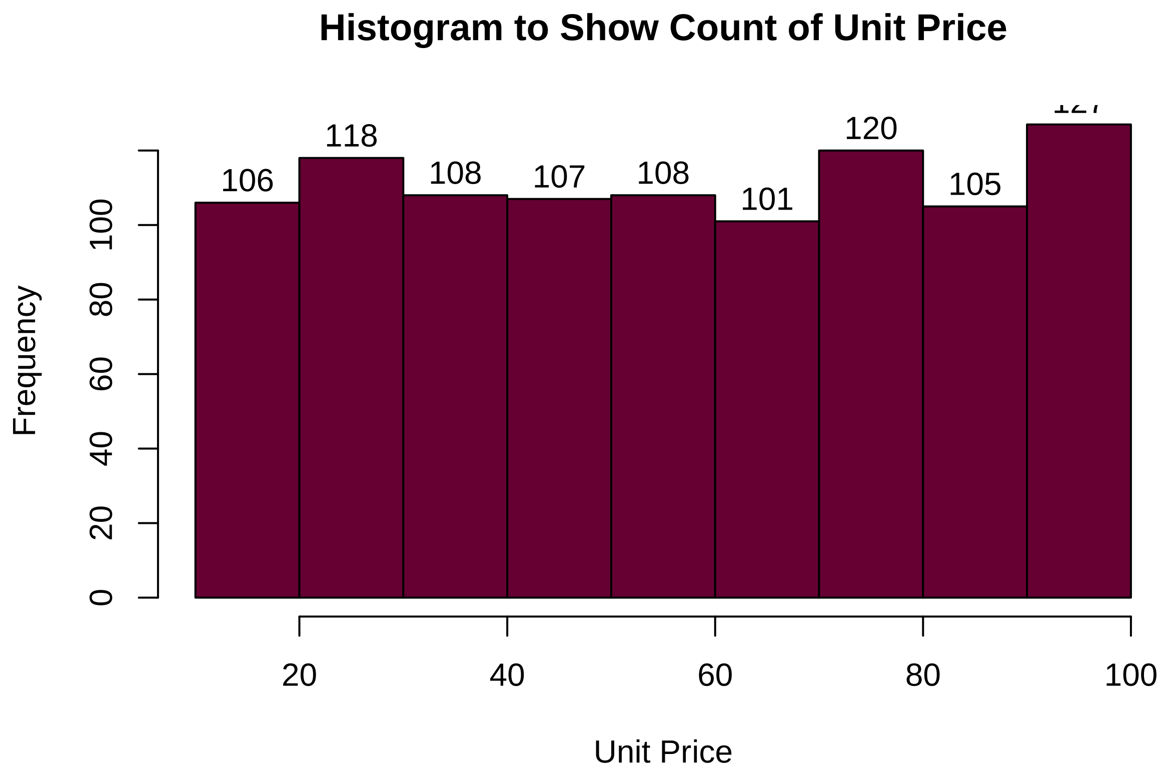
```
## [1] 6
```

```
mode.total  <- getmode(df$total)
mode.total
```

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

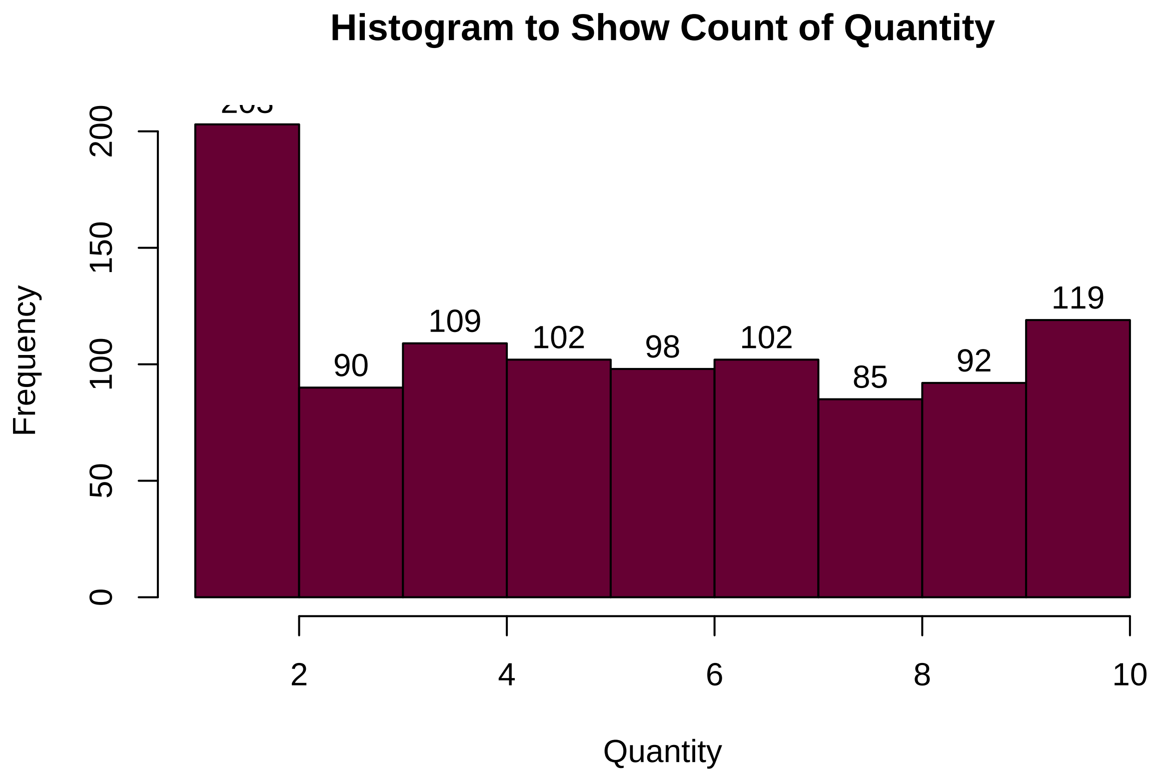
Histograms for Numerical Variables

```
# plot a histogram to visualize the distribution of values in 'Unit Price' column
hist(df$unit_price,
    col="#660033",
    main="Histogram to Show Count of Unit Price",
    xlab="Unit Price",
    ylab="Frequency",
    labels=TRUE)
```



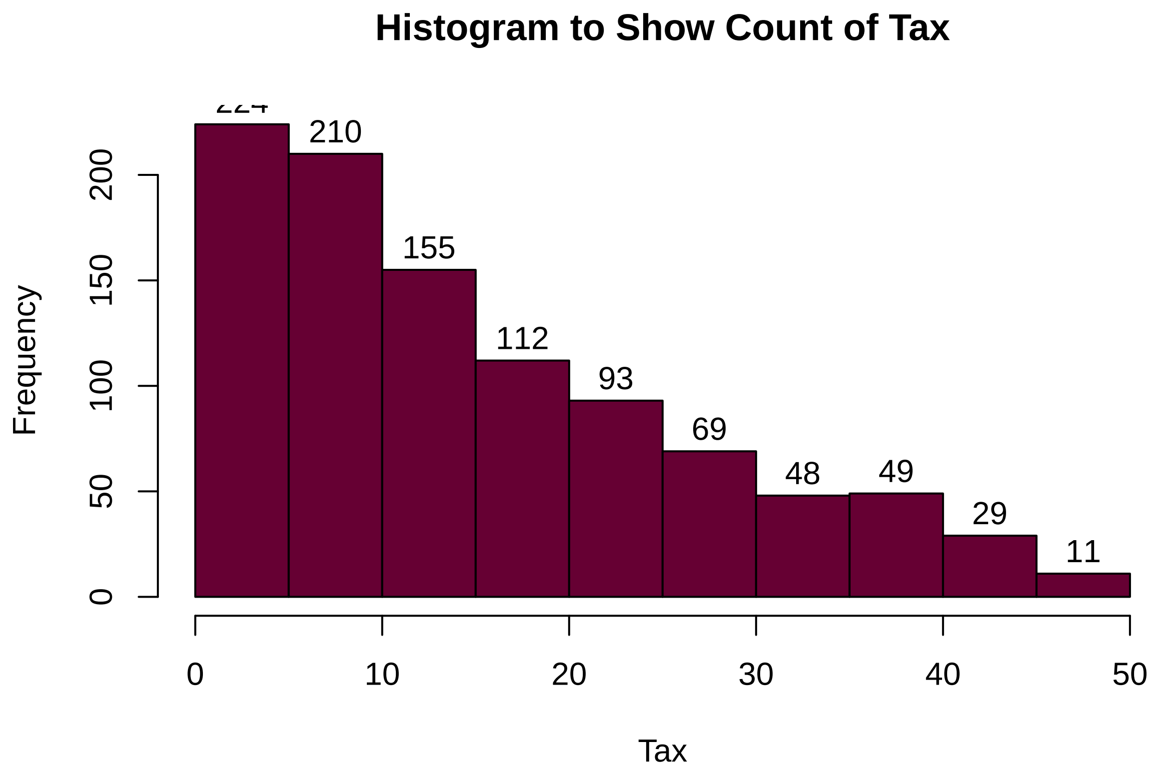
More items have a unit price of 90 - 100

```
# plot a histogram to visualize the distribution of values in 'Quantity' column
hist(df$quantity,
    col="#660033",
    main="Histogram to Show Count of Quantity",
    xlab="Quantity",
    ylab="Frequency",
    labels=TRUE)
```



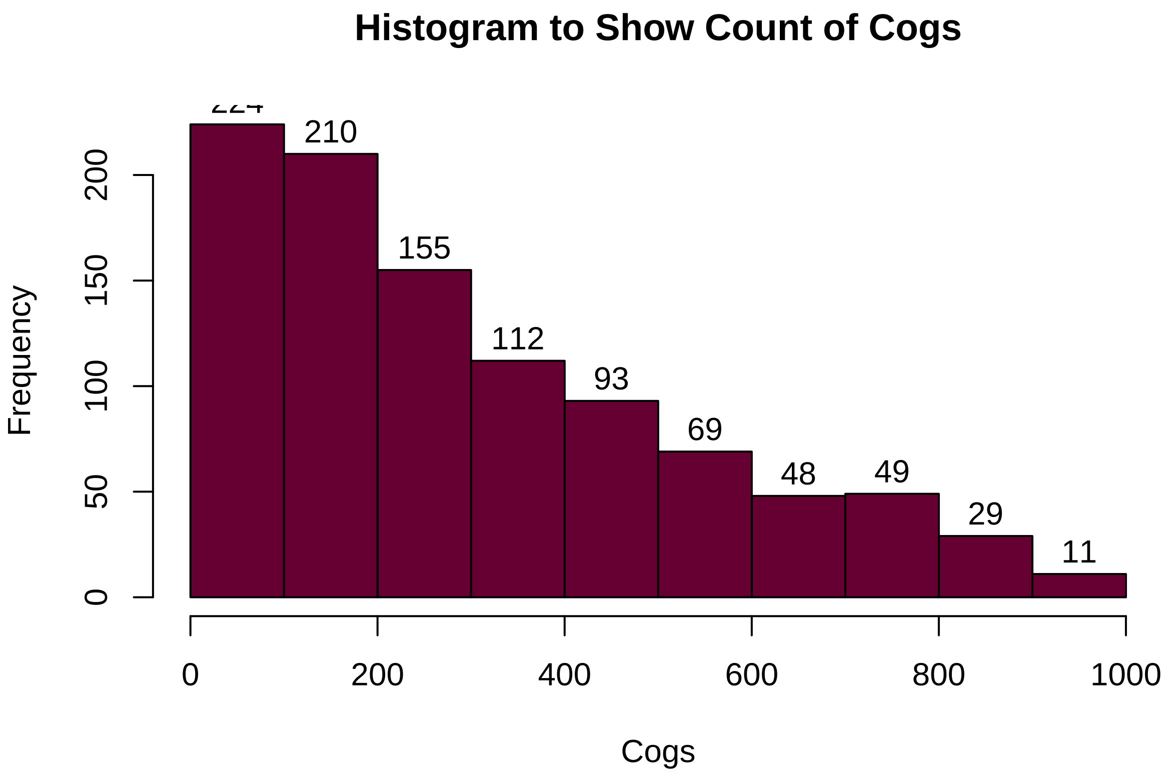
Most customers bought 1 item at a time

```
# plot a histogram to visualize the distribution of values in 'Tax' column
hist(df$tax,
     col="#660033",
     main="Histogram to Show Count of Tax",
     xlab="Tax",
     ylab="Frequency",
     labels=TRUE)
```



The tax bracket 0 - 5 had a higher number of items

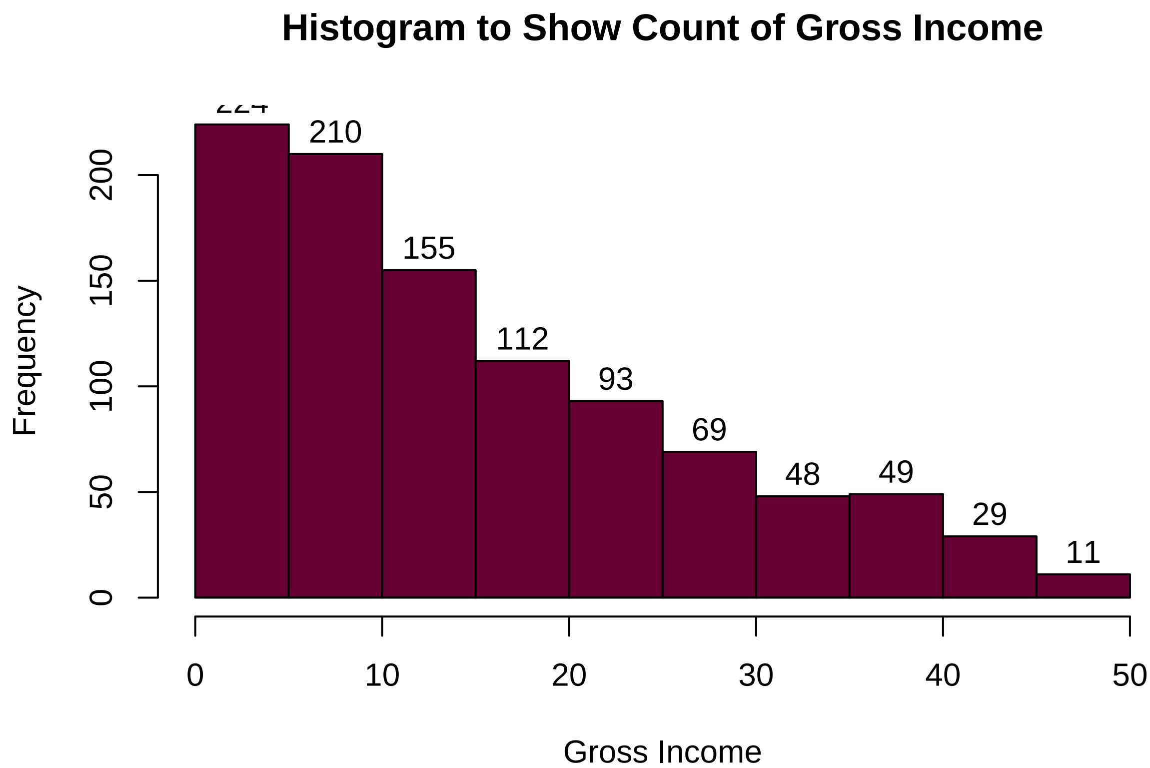
```
# plot a histogram to visualize the distribution of values in 'Cogs' column
hist(df$cogs,
     col="#660033",
     main="Histogram to Show Count of Cogs",
     xlab="Cogs",
     ylab="Frequency",
     labels=TRUE)
```



Cogs (Cost of goods sold) The items cost bracket of 0 - 100 has the higher amount of items

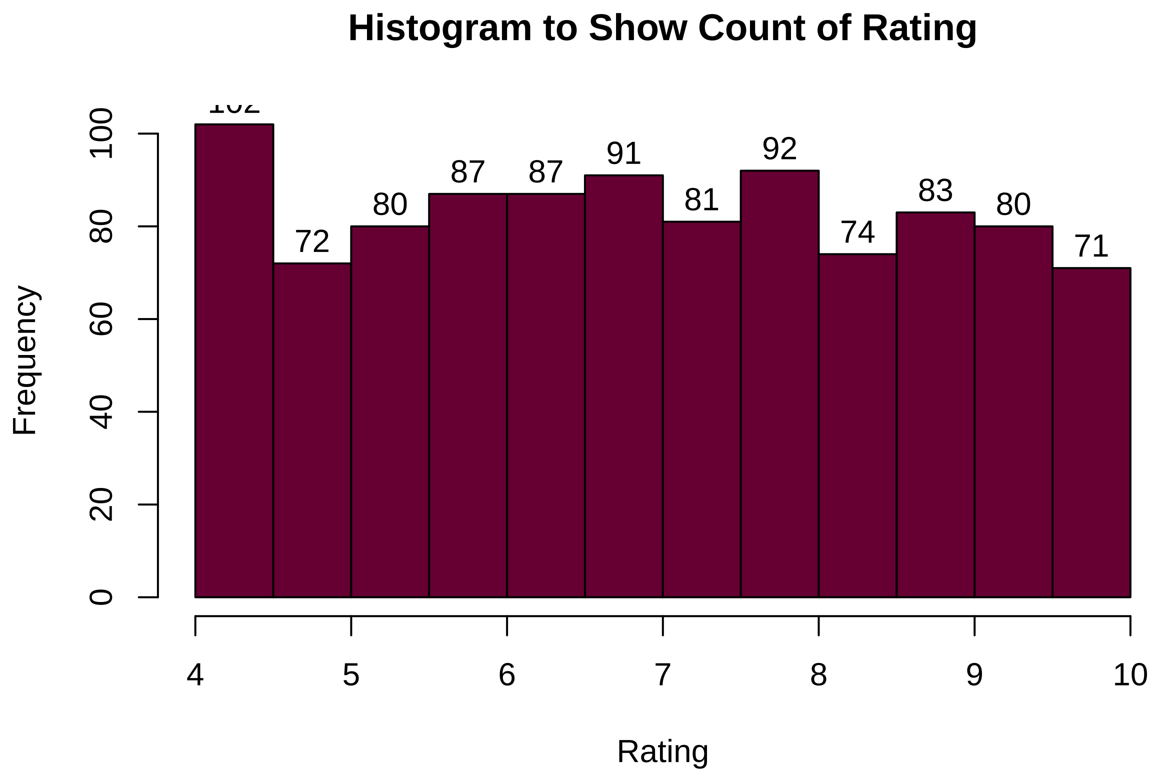
Tax and Cogs have a similar histogram

```
# plot a histogram to visualize the distribution of values in 'Gross Income' column

hist(df$gross_income,
    col="#660033",
    main="Histogram to Show Count of Gross Income",
    xlab="Gross Income",
    ylab="Frequency",
    labels=TRUE)
```



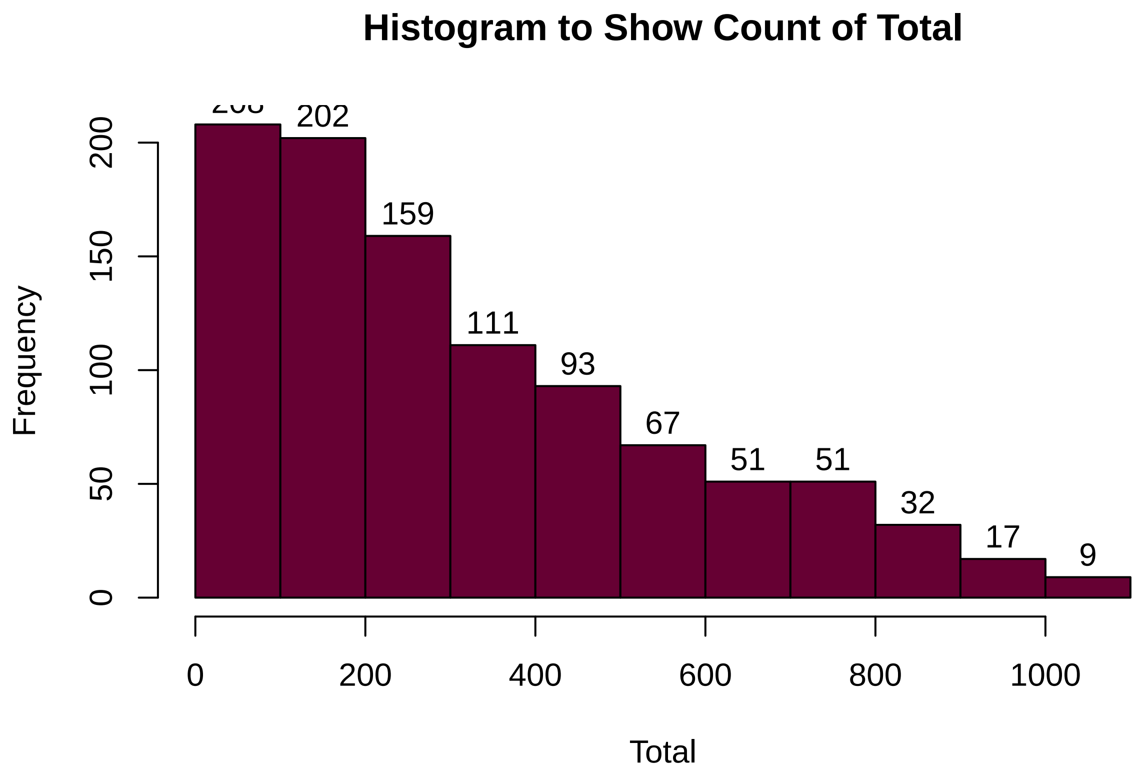
Gross Income, Tax and Cogs have a similar histogram

```
# plot a histogram to visualize the distribution of values in 'Rating' column

hist(df$rating,
     col="#660033",
     main="Histogram to Show Count of Rating",
     xlab="Rating",
     ylab="Frequency",
     labels=TRUE)
```



The rating 4 - 4.5 had higher amount of items than other rating brackets

```
# plot a histogram to visualize the distribution of values in 'total' column

hist(df$total,
     col="#660033",
     main="Histogram to Show Count of Total",
     xlab="Total",
     ylab="Frequency",
     labels=TRUE)
```

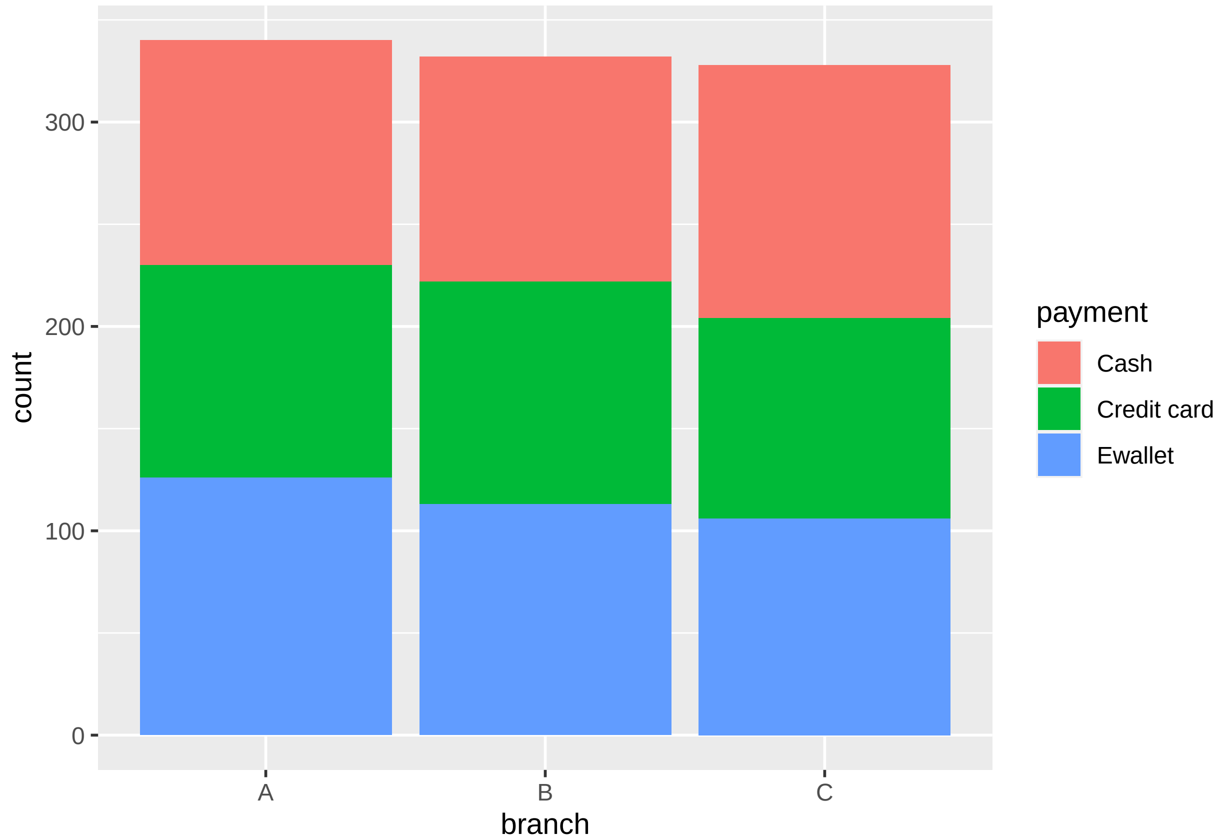


A higher number of items fell into the total price bracket 0 - 100

Bivariate Analysis

Payment method frequency in every branch

```
# Create barplot

ggplot(df, aes(fill=payment, x=branch)) +
    geom_bar(position="stack")
```



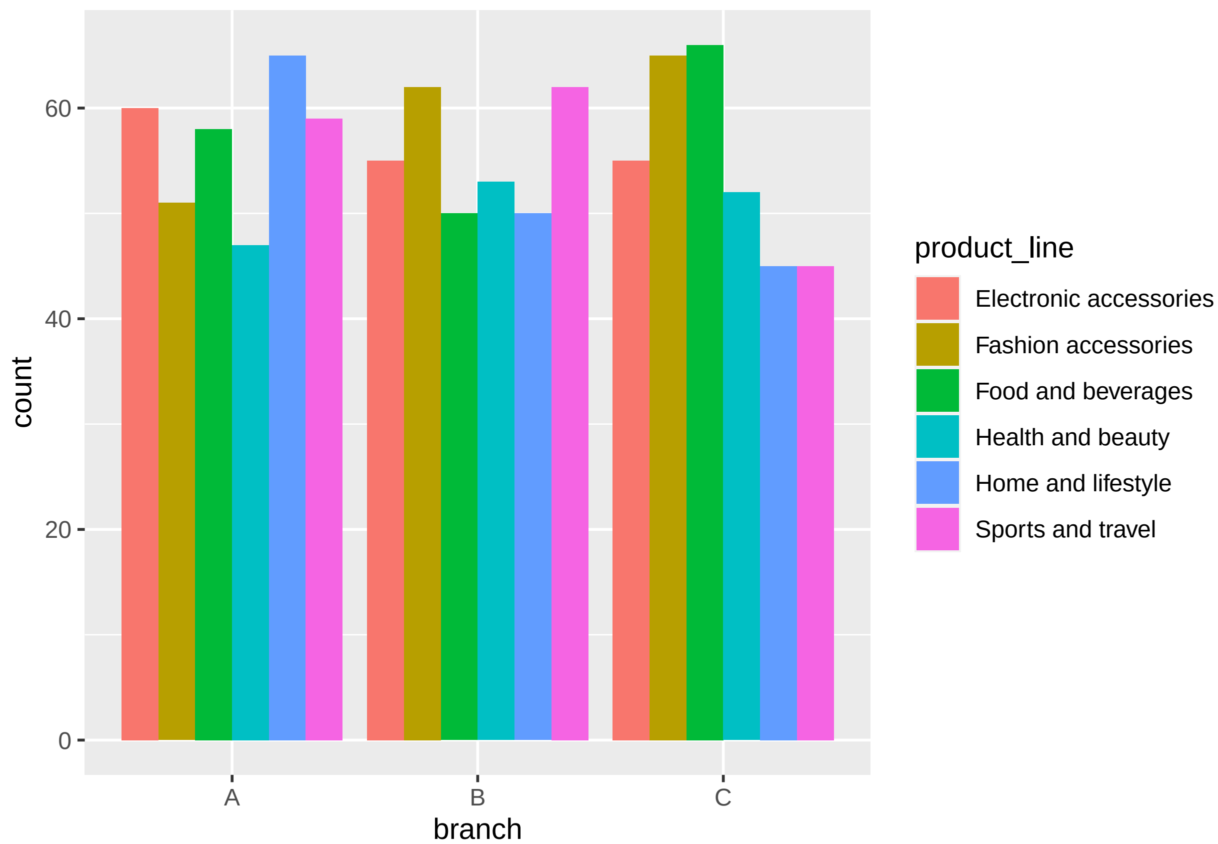
Most popular method of payment in Branch A is E-wallet

Most popular method of payment in Branch B is E-wallet but the other 2 modes of payment are also popular

Most popular method of payment in Branch B is Cash

Product line Frequency in every branch

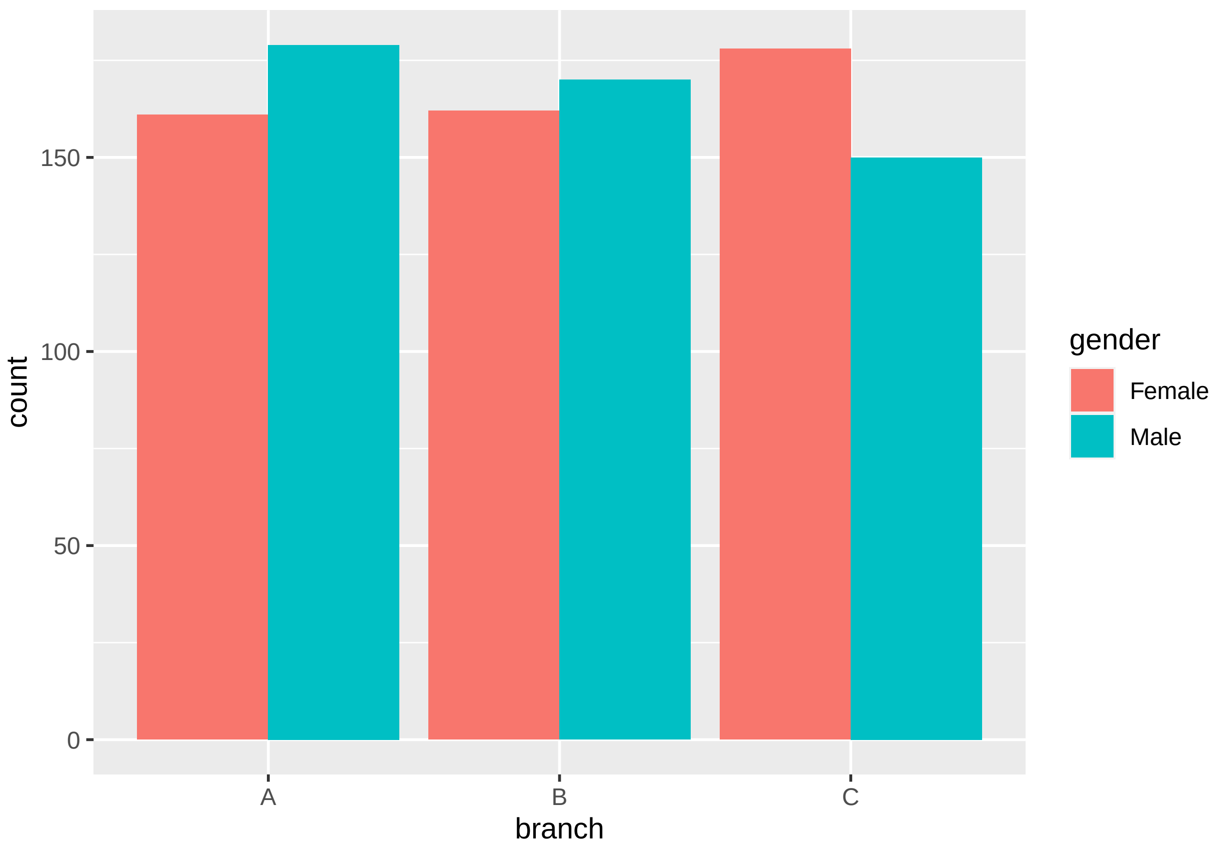
```
# Create Barplot

ggplot(df, aes(fill=product_line, x=branch)) +
    geom_bar(position="dodge")
```



From the plot, Branch B sells more sports & travel and Health & Beauty goods than the other branches. Branch A sells more home and lifestyle goods than the other branches. Branch c sells more Food & Beverages, Fashion Accessories and Electronic accessories than the other branches Therefore, the marketing team should stack these branches with the product with which they sell more.

Gender Frequency in every branch

```
ggplot(df, aes(fill=gender, x=branch)) +
    geom_bar(position="dodge")
```



There are more males in the Carrefour branches A and B than the females. This is not what many people assume as many people erroneously think that there are usually more females doing shopping. In branch C, there are more females shopping than males

```
head(df)
```

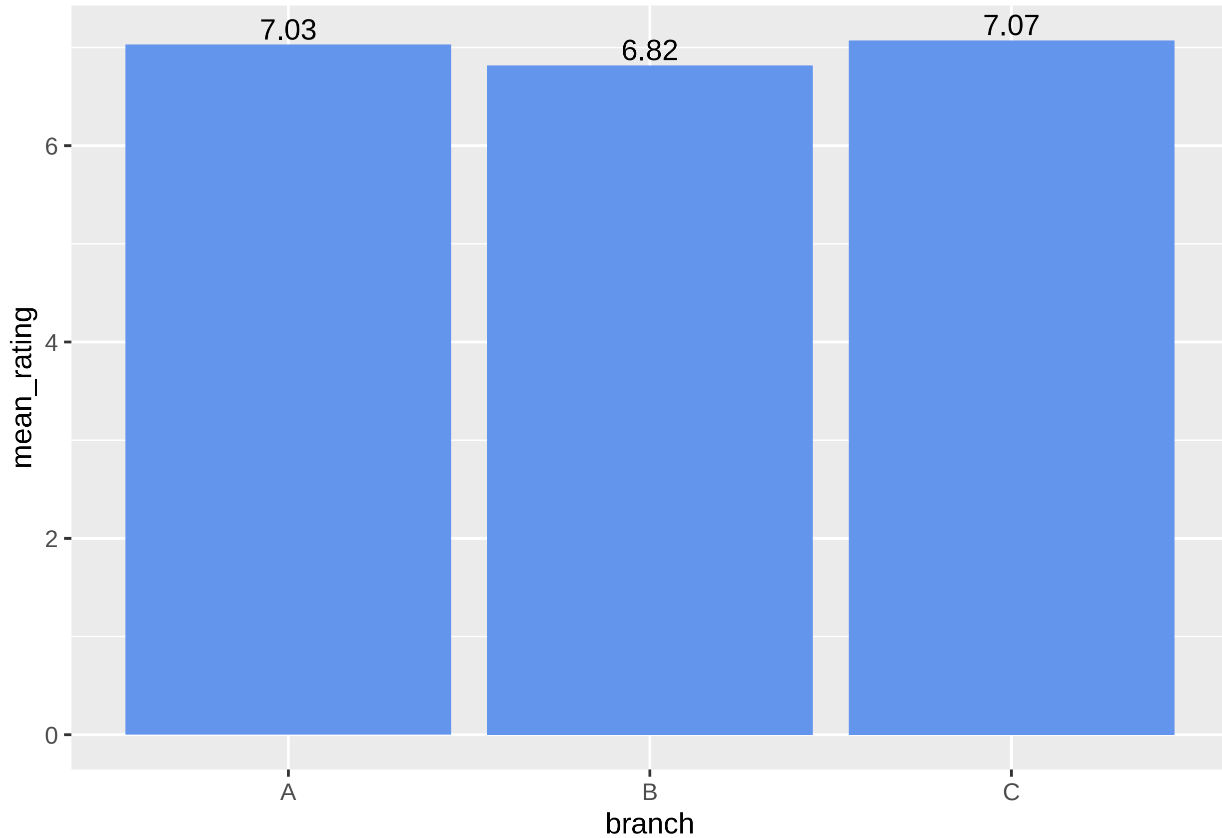
```
##     invoice_id branch customer_type gender           product_line unit_price
## 1: 750-67-8428      A        Member Female      Health and beauty      74.69
## 2: 226-31-3081      C        Normal Female Electronic accessories      15.28
## 3: 631-41-3108      A        Normal   Male     Home and 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_income rating
## 1:        7 26.1415 2019-01-05 13:08:00     Ewallet 522.83      26.1415    9.1
## 2:        5  3.8200 2019-03-08 10:29:00        Cash  76.40       3.8200    9.6
## 3:        7 16.2155 2019-03-03 13:23:00 Credit card 324.31      16.2155    7.4
## 4:        8 23.2880 2019-01-27 20:33:00     Ewallet 465.76      23.2880    8.4
## 5:        7 30.2085 2019-02-08 10:37:00     Ewallet 604.17      30.2085    5.3
## 6:        7 29.8865 2019-03-25 18:30:00     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
```

Mean Rating for items every branch

```
# calculate mean rating for each branch
library(dplyr)
```

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

## 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
```

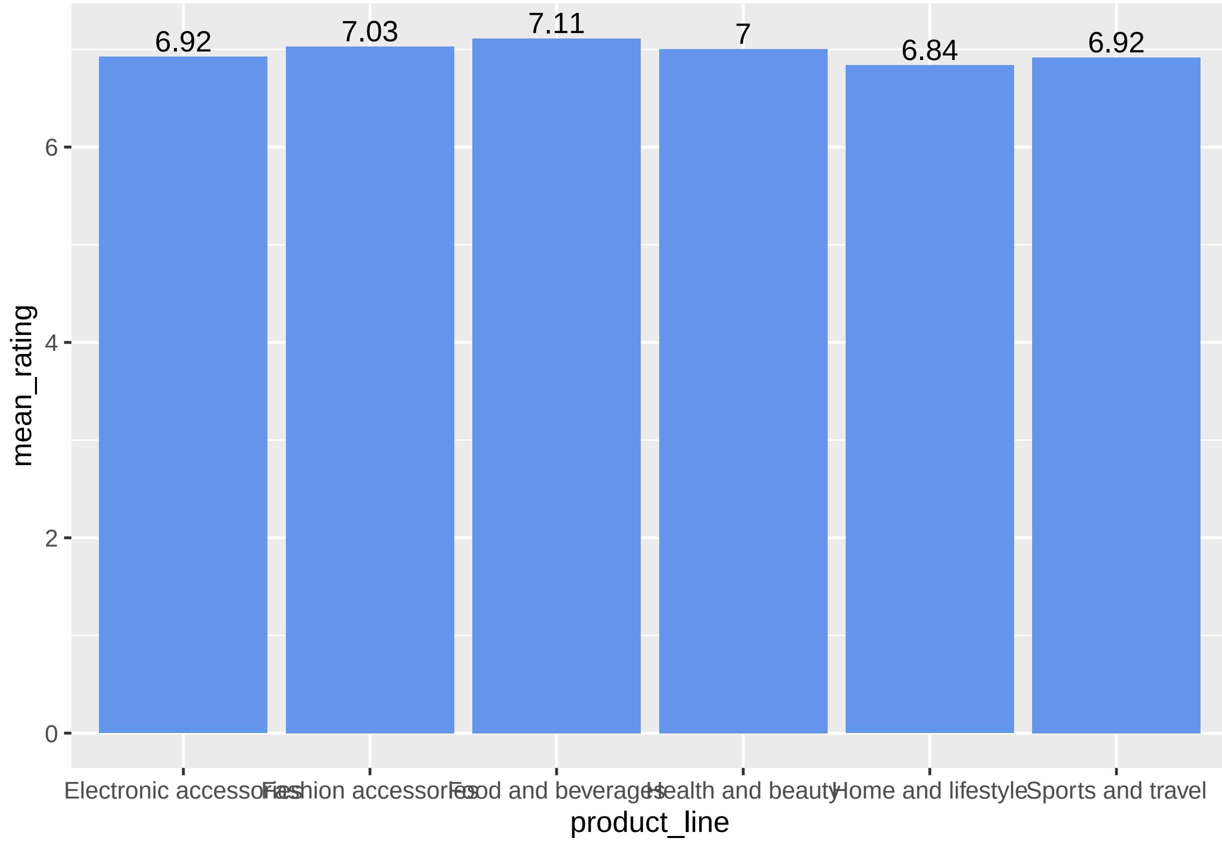
```
plotdata <- df %>%
  group_by(branch) %>%
  summarize(mean_rating = mean(rating))

# plot mean salaries
ggplot(plotdata, aes(x = branch, y = mean_rating)) +
  geom_bar(stat = "identity", fill = "cornflowerblue") +
  geom_text(aes(label = round(mean_rating,2)),
            vjust = -0.25)
```



Branch C has a higher mean rating of the product line items than the other branches

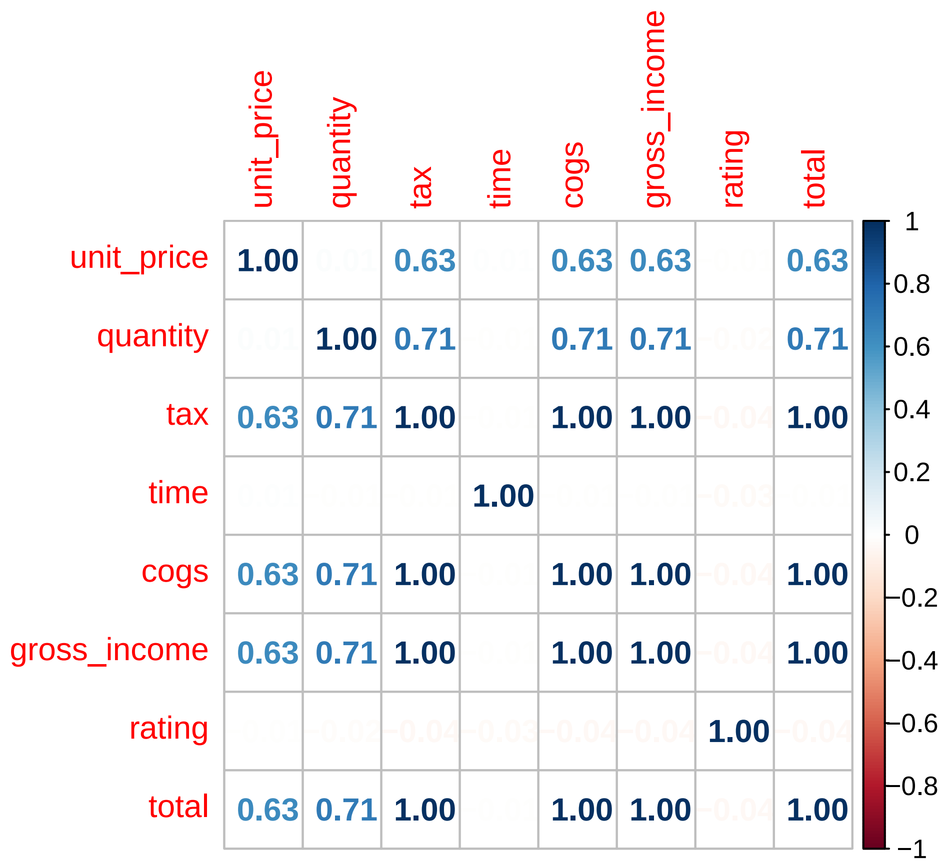
Mean Rating for items in every product line

```
# calculate mean rating for each product line
library(dplyr)
plotdata1 <- df %>%
  group_by(product_line) %>%
  summarize(mean_rating = mean(rating))

# plot mean salaries
ggplot(plotdata1, aes(x = product_line, y = mean_rating)) +
  geom_bar(stat = "identity", fill = "cornflowerblue") +
  geom_text(aes(label = round(mean_rating,2)),
            vjust = -0.25)
```



Food and Beverages has the highest rating and Home & Lifestyle has the least rating

```
# calculate correlations
correlations <- cor(df_num)
# create correlation plot
corrplot(correlations, method="number")
```



Gross income, tax, cogs and total have a correlation of 1 because they are calculated from from the same starting point (Cogs) and with the same fractions for tax, gross income and total.

```
# Make a copy of the df
df_copy <- df

# Label Encoder
#Branch , customer_type, Gender, productline , payment
lbl <- LabelEncoder$new()

lbl$fit(df$branch)
df$branch <- lbl$fit_transform(df$branch)
lbl$fit(df$customer_type)
df$customer_type <- lbl$fit_transform(df$customer_type)
lbl$fit(df$gender)
df$gender <- lbl$fit_transform(df$gender)
lbl$fit(df$product_line)
df$product_line <- lbl$fit_transform(df$product_line)
lbl$fit(df$payment)
df$payment <- lbl$fit_transform(df$payment)
```

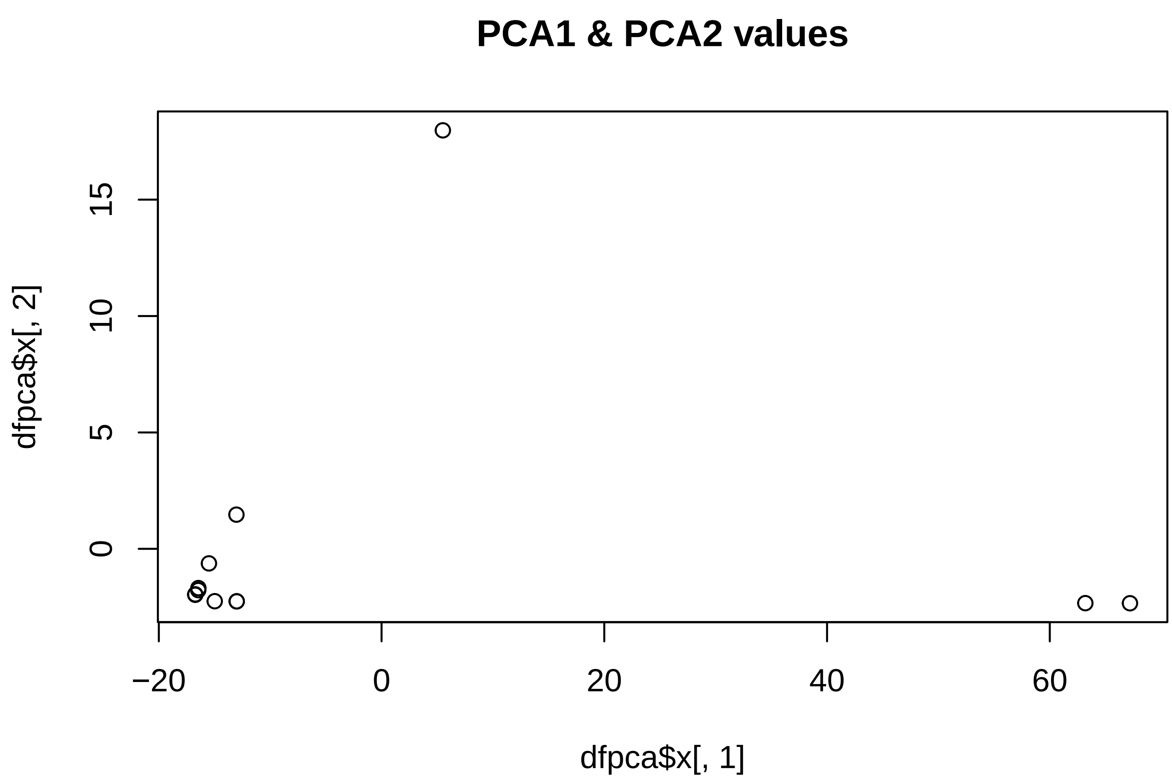
```
# Drop the categorcal columns
df$invoice_id <- NULL
df$date <- NULL
df$time <- NULL
```

```r
str(df)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  12 variables:
##  $ branch       : num  0 1 0 0 0 1 0 1 0 2 ...
##  $ customer_type: num  0 1 1 0 1 1 0 1 0 0 ...
##  $ gender       : num  0 0 1 1 1 1 0 0 0 0 ...
##  $ product_line : num  0 1 2 0 3 1 1 2 0 4 ...
##  $ unit_price   : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ quantity     : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ tax          : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ payment      : num  0 1 2 0 0 0 0 0 2 2 ...
##  $ cogs         : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ gross_income : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ rating       : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
##  $ total        : num  549 80.2 340.5 489 634.4 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Performing the PCA

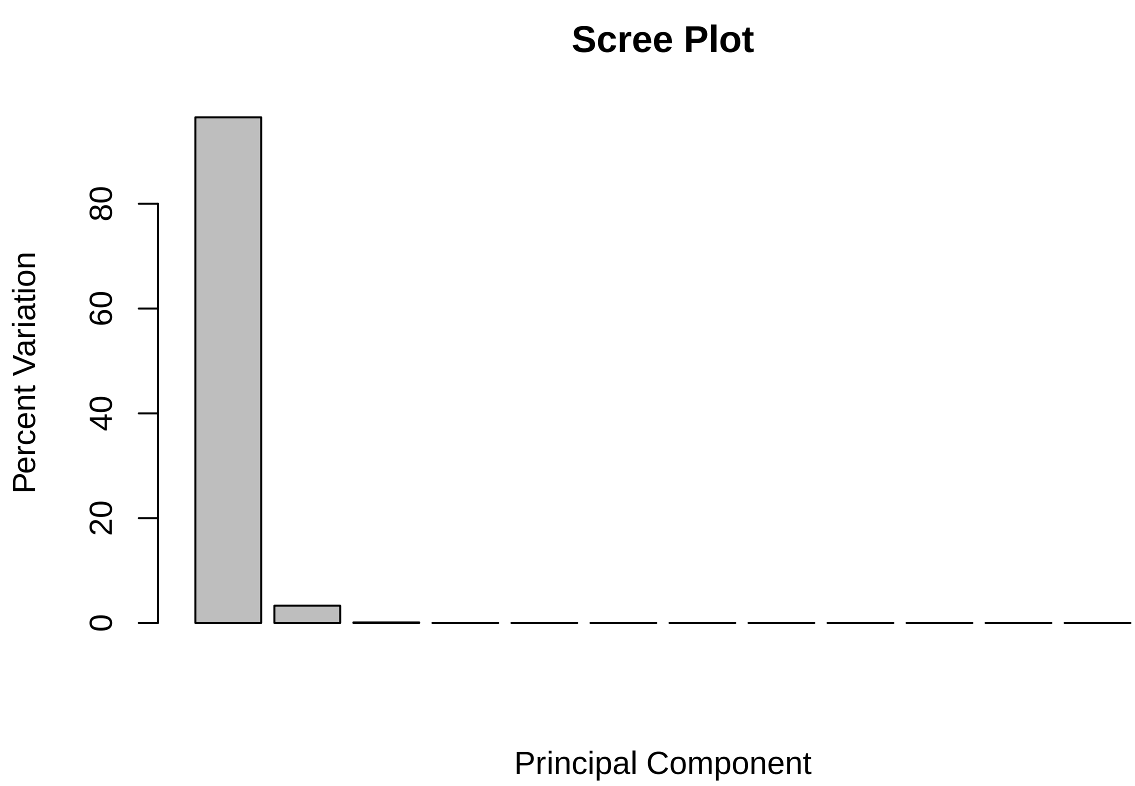
```r
# # Run the PCA on the df
dfpca <- prcomp(t(df),center = TRUE, scale=TRUE)
## plot pc1 and pc2
plot(dfpca$x[,1], dfpca$x[,2], main = "PCA1 & PCA2 values")
```



```r
# Lets get a summary of the pca
summary (dfpca)
```

```
## Importance of components:
##                            PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     31.0616 5.76498 1.21319 0.50237 0.29831 0.23451 0.20497
## Proportion of Variance  0.9648 0.03323 0.00147 0.00025 0.00009 0.00005 0.00004
## Cumulative Proportion   0.9648 0.99806 0.99953 0.99978 0.99987 0.99993 0.99997
##                            PC8     PC9      PC10      PC11      PC12
## Standard deviation     0.14119 0.09579 2.638e-14 1.965e-15 6.211e-17
## Proportion of Variance 0.00002 0.00001 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  0.99999 1.00000 1.000e+00 1.000e+00 1.000e+00
```

```r
## make a scree plot
pca.var <- dfpca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
barplot(pca.var.per, main="Scree Plot", xlab="Principal Component", ylab="Percent Variation")
```

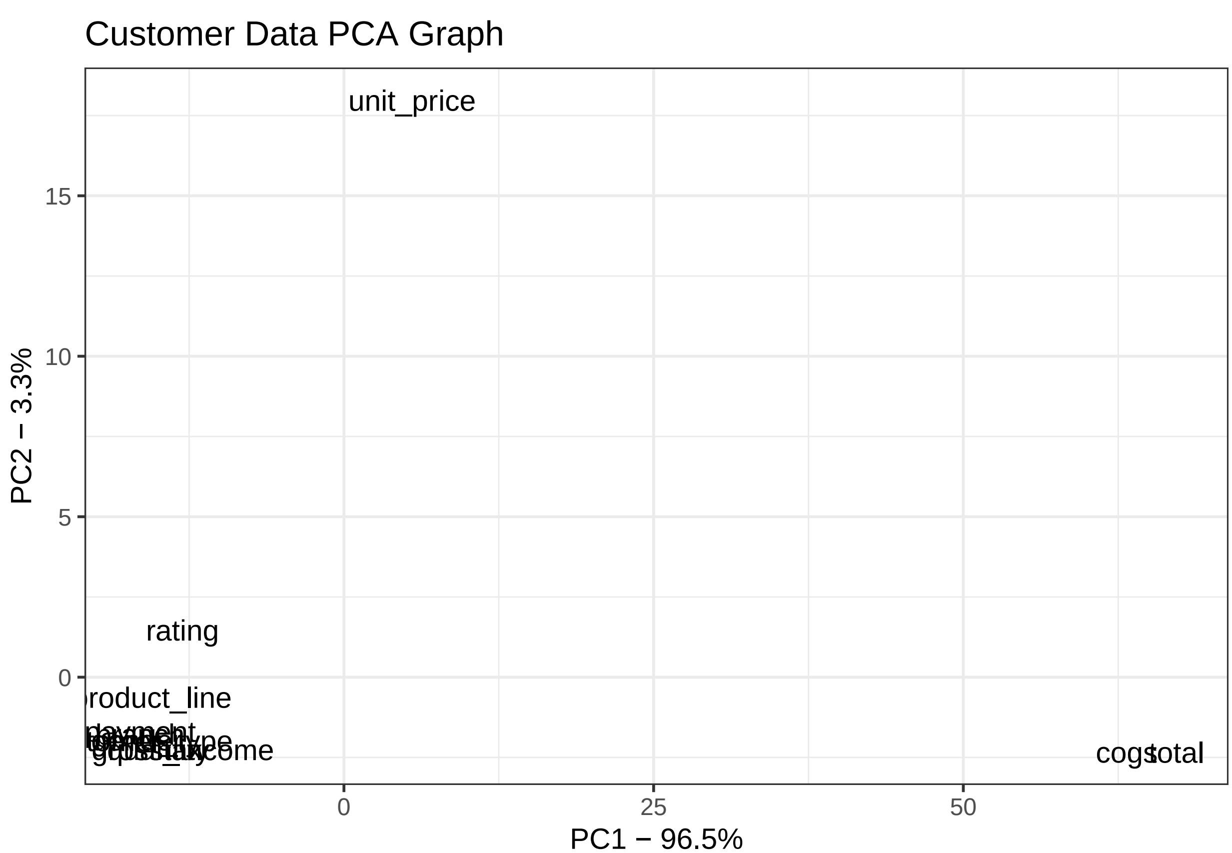



```r
## plot that shows the PCs and the variation:
pca.data <- data.frame(Sample=rownames(dfpca$x),
                       X=dfpca$x[,1],
                       Y=dfpca$x[,2])
pca.data
```

```
##                         Sample          X          Y
```

```
## branch                 branch -16.460925 -1.774218
## customer_type customer_type -16.728657 -1.974943
## gender                 gender -16.727800 -1.955175
## product_line     product_line -15.501089 -0.625429
## unit_price         unit_price   5.501295 17.977265
## quantity             quantity -14.979897 -2.249242
## tax                       tax -13.006234 -2.255524
## payment               payment -16.446861 -1.686001
## cogs                     cogs  63.189817 -2.333115
## gross_income     gross_income -13.006234 -2.255524
## rating                 rating -13.033551  1.469104
## total                   total  67.200135 -2.337199
```

```r
ggplot(data=pca.data, aes(x=X, y=Y, label=Sample)) +
  geom_text() +
  xlab(paste("PC1 - ", pca.var.per[1], "%", sep="")) +
  ylab(paste("PC2 - ", pca.var.per[2], "%", sep="")) +
  theme_bw() +
  ggtitle("Customer Data PCA Graph")
```



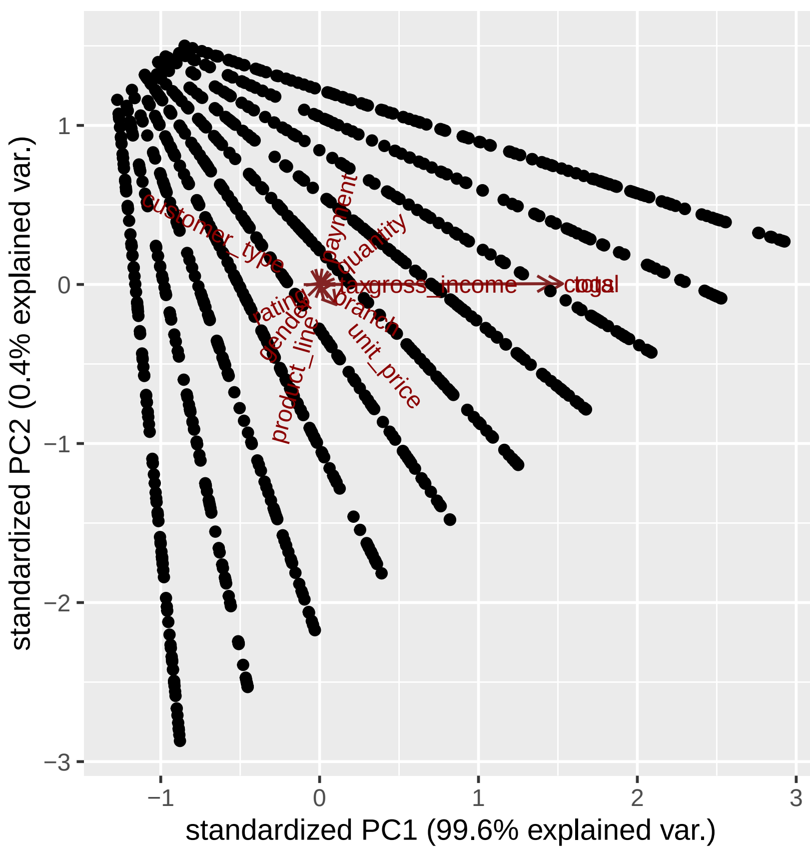
PC1 explains 96.5% of the total variance, which means that nearly 96% of the information in the dataset (11 variables) can be encapsulated by just that one Principal Component. PC2 explains 3.3% of the variance. etc

```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

## Loading required package: scales

## Loading required package: grid
```

```
ggbiplot (prcomp(df))
```



Part 2: Feature Selection

using the filter method.

```
# Installing and loading our caret package
suppressWarnings(
        suppressMessages(if
                         (!require(caret, quietly=TRUE))
                install.packages("caret")))
library(caret)
```

```
# Installing and loading the corrplot package for plotting
# ---
#
suppressWarnings(
        suppressMessages(if
                         (!require(corrplot, quietly=TRUE))
                install.packages("corrplot")))
library(corrplot)
```

```
# Calculating the correlation matrix
correlationMatrix <- cor(df)
# Find attributes that are highly correlated
# ---
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75)
highlyCorrelated
```

```
## [1]  7  9 10
```

```
correlationMatrix
```

```
##                    branch customer_type       gender product_line   unit_price
## branch         1.000000000  -0.004899261 -0.012218875   0.01257525  0.013763477
## customer_type -0.004899261   1.000000000  0.039996160  -0.02510945 -0.020237875
## gender        -0.012218875   0.039996160  1.000000000  -0.06612647  0.015444630
## product_line   0.012575246  -0.025109450 -0.066126475   1.00000000  0.038427649
## unit_price     0.013763477  -0.020237875  0.015444630   0.03842765  1.000000000
## quantity       0.002120920  -0.016762706 -0.074258307  -0.06251471  0.010777564
## tax            0.012811933  -0.019670283 -0.049450989  -0.01854396  0.633962089
## payment        0.026725563  -0.069286242 -0.049514182   0.01051098 -0.019637884
## cogs           0.012811933  -0.019670283 -0.049450989  -0.01854396  0.633962089
## gross_income   0.012811933  -0.019670283 -0.049450989  -0.01854396  0.633962089
## rating        -0.049585348   0.018888672  0.004800208   0.02339096 -0.008777507
## total          0.012811933  -0.019670283 -0.049450989  -0.01854396  0.633962089
##                   quantity          tax      payment         cogs gross_income
## branch         0.002120920  0.012811933  0.026725563  0.012811933  0.012811933
## customer_type -0.016762706 -0.019670283 -0.069286242 -0.019670283 -0.019670283
## gender        -0.074258307 -0.049450989 -0.049514182 -0.049450989 -0.049450989
## product_line  -0.062514713 -0.018543956  0.010510982 -0.018543956 -0.018543956
## unit_price     0.010777564  0.633962089 -0.019637884  0.633962089  0.633962089
## quantity       1.000000000  0.705510186  0.007333388  0.705510186  0.705510186
```

```
## tax           0.705510186  1.000000000  0.008823723  1.000000000  1.000000000
## payment       0.007333388  0.008823723  1.000000000  0.008823723  0.008823723
## cogs          0.705510186  1.000000000  0.008823723  1.000000000  1.000000000
## gross_income  0.705510186  1.000000000  0.008823723  1.000000000  1.000000000
## rating       -0.015814905 -0.036441705  0.013001094 -0.036441705 -0.036441705
## total         0.705510186  1.000000000  0.008823723  1.000000000  1.000000000
##                    rating        total
## branch       -0.049585348  0.012811933
## customer_type  0.018888672 -0.019670283
## gender        0.004800208 -0.049450989
## product_line  0.023390962 -0.018543956
## unit_price   -0.008777507  0.633962089
## quantity     -0.015814905  0.705510186
## tax          -0.036441705  1.000000000
## payment       0.013001094  0.008823723
## cogs         -0.036441705  1.000000000
## gross_income -0.036441705  1.000000000
## rating        1.000000000 -0.036441705
## total        -0.036441705  1.000000000
```

```
# Names of highly correlations
names (df[, 7])
```

```
## [1] "tax"
```

```
names (df[, 9])
```

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

```
names (df[, 11])
```

```
## [1] "rating"
```

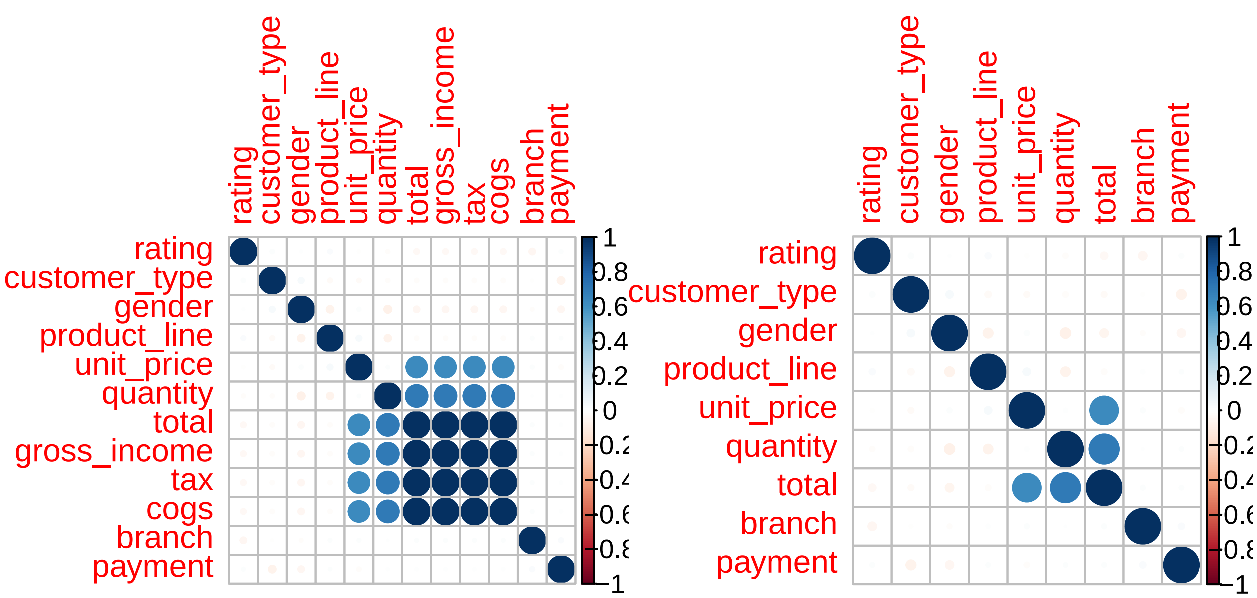
```
# Next step is removing the variables with high correlation
df_low <- df[-highlyCorrelated]
df_low$tax <- NULL
df_low$cogs <- NULL
df_low$gross_income <- NULL
```

```
cor2 <- cor(df_low)
cor2
```

```
##                    branch customer_type       gender product_line   unit_price
## branch        1.000000000  -0.006113857 -0.013460802  0.008640181  0.013551891
## customer_type -0.006113857   1.000000000  0.037110365 -0.026797451 -0.020544234
## gender       -0.013460802   0.037110365  1.000000000 -0.067954892  0.015205909
## product_line  0.008640181  -0.026797451 -0.067954892  1.000000000  0.037893893
## unit_price    0.013551891  -0.020544234  0.015205909  0.037893893  1.000000000
## quantity      0.001930628  -0.018705894 -0.076351656 -0.063649293  0.009800802
## payment       0.025373513  -0.068185247 -0.048336870  0.010315646 -0.018116773
## rating       -0.049616876   0.017746989  0.003631188  0.023536164 -0.008367916
```

```
## total          0.012931022  -0.020884334 -0.050733456 -0.019186236  0.633734080
##                   quantity     payment        rating        total
## branch         0.001930628  0.02537351 -0.049616876  0.01293102
## customer_type -0.018705894 -0.06818525  0.017746989 -0.02088433
## gender        -0.076351656 -0.04833687  0.003631188 -0.05073346
## product_line  -0.063649293  0.01031565  0.023536164 -0.01918624
## unit_price     0.009800802 -0.01811677 -0.008367916  0.63373408
## quantity       1.000000000  0.01020392 -0.016105001  0.70504027
## payment        0.010203918  1.00000000  0.012852398  0.01146344
## rating        -0.016105001  0.01285240  1.000000000 -0.03642915
## total          0.705040267  0.01146344 -0.036429151  1.00000000
```

```r
# Performing our graphical comparison
# ---
#
library(stats)
par(mfrow = c(1, 2))
corrplot(correlationMatrix, order = "hclust")
corrplot(cor(df_low), order = "hclust")
```



From the filter method, There are a few columns that have been eliminated because of high such a high correlation: - Tax - Cogs _ Gross Income

We should try another method and see what other features we will remain with

wrapper method

```r
# Installing and loading our clustvarsel package
suppressWarnings(
        suppressMessages(if
                         (!require(clustvarsel, quietly=TRUE))
                install.packages("clustvarsel")))

library(clustvarsel)
# Installing and loading our mclust package
suppressWarnings(
        suppressMessages(if
                         (!require(mclust, quietly=TRUE))
                install.packages("mclust")))
library(mclust)
```

```r
# Sequential forward greedy search (default)
#
out = clustvarsel(df_low, G = 1:5)
out
```

```
## ------------------------------------------------------
## Variable selection for Gaussian model-based clustering
## Stepwise (forward/backward) greedy search
## ------------------------------------------------------
##
##  Variable proposed Type of step  BICclust Model G    BICdiff Decision
##              total          Add -13434.37     V 4   385.9196 Accepted
##         unit_price          Add -21507.86   VEV 5   800.0361 Accepted
##           quantity          Add -22352.30   VVV 5  2462.4005 Accepted
##           quantity       Remove -21507.86   VEV 5  2462.4005 Rejected
##             rating          Add -24954.28   VEV 5  1322.0645 Accepted
##             rating       Remove -22352.30   VVV 5  1322.0645 Rejected
##       product_line          Add -30232.02   EVV 5 -1369.5858 Rejected
##             rating       Remove -22352.30   VVV 5  1322.0645 Rejected
##
## Selected subset: total, unit_price, quantity, rating
```

For the wrapper method only a few columns have been selected for modelling. these are: - Total - Quantity - Unit Price

Embended methods

```r
suppressWarnings(
        suppressMessages(if
                         (!require(wskm, quietly=TRUE))
                install.packages("wskm")))
library(wskm)
set.seed(2)
model <- ewkm(df_low, 3, lambda=2, maxiter=1000)
```

```r
suppressWarnings(
        suppressMessages(if
                         (!require(cluster, quietly=TRUE))
                install.packages("cluster")))
library("cluster")
clusplot(df_low, model$cluster, color=TRUE, cor = TRUE, shade=TRUE,
         labels=2, lines=1,main='Cluster Analysis for df')
```

```
## Warning in plot.window(...): "cor" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "cor" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "cor" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "cor" is not a
## graphical parameter
```

```
## Warning in box(...): "cor" is not a graphical parameter
```

```
## Warning in title(...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

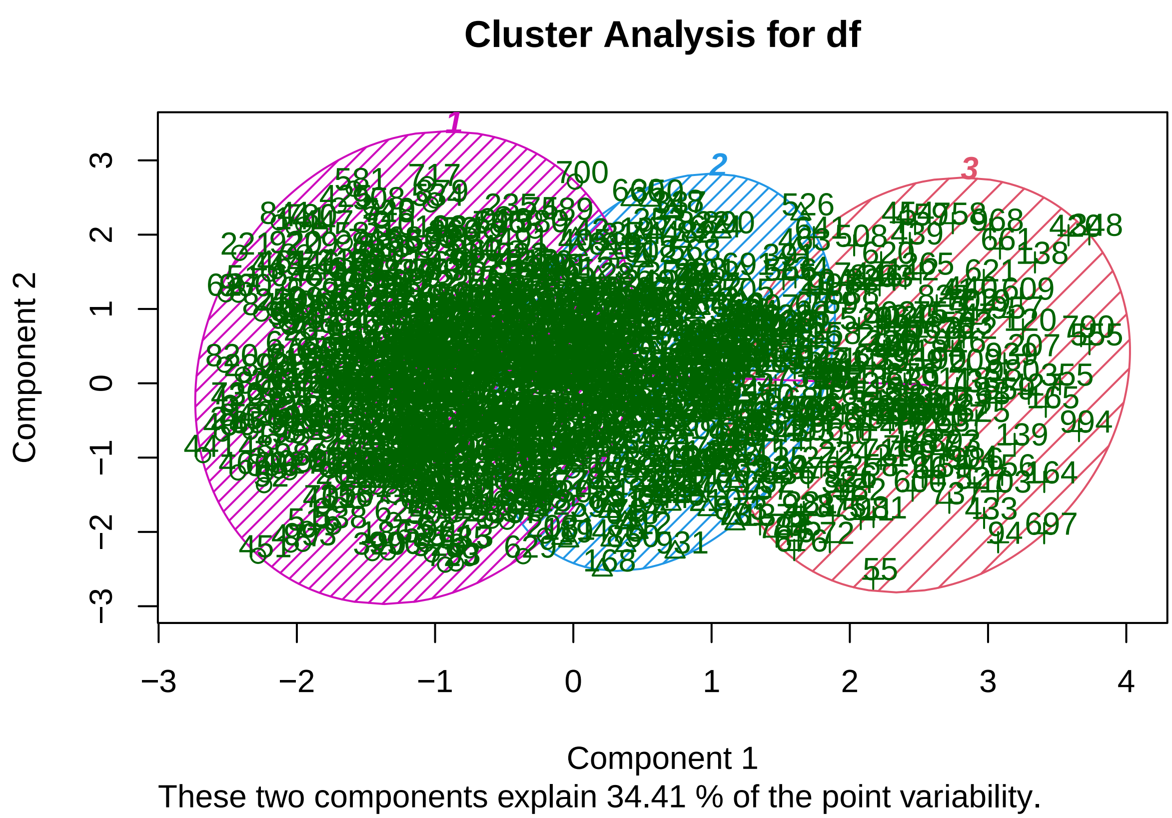
```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in polygon(z[[k]], density = if (shade) density[k] else 0, col =
## col.clus[jInd[i]], : "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in polygon(z[[k]], density = if (shade) density[k] else 0, col =
## col.clus[jInd[i]], : "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter
```

```
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter

## Warning in polygon(z[[k]], density = if (shade) density[k] else 0, col =
## col.clus[jInd[i]], : "cor" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "cor" is not a graphical
## parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "cor" is not a graphical
## parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "cor" is not a graphical
```

```
## parameter

## Warning in segments(loc[i, 1], loc[i, 2], loc[j, 1], loc[j, 2], col = 6, : "cor"
## is not a graphical parameter

## Warning in text.default(xy, labels = labs, ...): "cor" is not a graphical
## parameter

## Warning in text.default(xy, labels = labs, ...): "cor" is not a graphical
## parameter
```




```r
# Weights are calculated for each variable and cluster.
# They are a measure of the relative importance of each variable
# with regards to the membership of the observations to that cluster.
# The weights are incorporated into the distance function,
# typically reducing the distance for more important variables.
# Weights remain stored in the model and we can check them as follows:
#
round(model$weights*100,2)
```

```
##   branch customer_type gender product_line unit_price quantity payment rating
## 1      0         45.15  54.84            0          0        0       0      0
## 2      0         43.39  56.60            0          0        0       0      0
## 3      0         50.00  50.00            0          0        0       0      0
```

```
##   total
## 1     0
## 2     0
## 3     0
```