PCA and t-SNE

2022-04-01

```r
# Load Libraries
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.3

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

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

## Warning: package 'ggplot2' was built under R version 4.1.3

## Warning: package 'dplyr' was built under R version 4.1.3

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

```r
library(moments)
library(heatmaply)
```

```
## Warning: package 'heatmaply' was built under R version 4.1.3

## Loading required package: plotly

##
## Attaching package: 'plotly'

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

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

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

```
## Loading required package: viridis

## Warning: package 'viridis' was built under R version 4.1.3

## Loading required package: viridisLite

##
## ======================
## Welcome to heatmaply version 1.3.0
##
## Type citation('heatmaply') for how to cite the package.
## Type ?heatmaply for the main documentation.
##
## The github page is: https://github.com/talgalili/heatmaply/
## Please submit your suggestions and bug-reports at: https://github.com/talgalili/heatmaply/issues
## You may ask questions at stackoverflow, use the r and heatmaply tags:
##   https://stackoverflow.com/questions/tagged/heatmaply
## ======================
```

```
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
library(FactoMineR)
```

```
## Warning: package 'FactoMineR' was built under R version 4.1.3
```

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.1.3

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

```
# Import the dataset
smarket <- read.csv("http://bit.ly/CarreFourDataset")
```

```
# Preview the first few rows
head(smarket)
```

```
##    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
```

```
# Preview the first few rows
tail(smarket)
```

```
##       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
```

```
# Check number of records and variables
dim(smarket)
```

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

```
# Check the datatypes of our dataset
glimpse(smarket)
```

```
## Rows: 1,000
## Columns: 16
## $ Invoice.ID              <chr> "750-67-8428", "226-31-3081", "631-41-3108", "~
## $ Branch                  <chr> "A", "C", "A", "A", "A", "C", "A", "C", "A", "~
## $ Customer.type           <chr> "Member", "Normal", "Normal", "Member", "Norma~
## $ Gender                  <chr> "Female", "Female", "Male", "Male", "Male", "M~
## $ Product.line            <chr> "Health and beauty", "Electronic accessories",~
## $ Unit.price              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 68.8~
```

```
## $ Quantity                 <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10, 10~
## $ Tax                      <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085, 29~
## $ Date                     <chr> "1/5/2019", "3/8/2019", "3/3/2019", "1/27/2019~
## $ Time                     <chr> "13:08", "10:29", "13:23", "20:33", "10:37", "~
## $ Payment                  <chr> "Ewallet", "Cash", "Credit card", "Ewallet", "~
## $ cogs                     <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597.73,~
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7619~
## $ gross.income             <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085, 29~
## $ Rating                   <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2, 5~
## $ Total                    <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634.378~
```

```r
# Check the summary of our dataset
summary(smarket)
```

```
##   Invoice.ID           Branch          Customer.type         Gender
##  Length:1000        Length:1000        Length:1000        Length:1000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  Product.line         Unit.price       Quantity          Tax
##  Length:1000        Min.   :10.08   Min.   : 1.00   Min.   : 0.5085
##  Class :character   1st Qu.:32.88   1st Qu.: 3.00   1st Qu.: 5.9249
##  Mode  :character   Median :55.23   Median : 5.00   Median :12.0880
##                     Mean   :55.67   Mean   : 5.51   Mean   :15.3794
##                     3rd Qu.:77.94   3rd Qu.: 8.00   3rd Qu.:22.4453
##                     Max.   :99.96   Max.   :10.00   Max.   :49.6500
##      Date               Time             Payment              cogs
##  Length:1000        Length:1000        Length:1000        Min.   : 10.17
##  Class :character   Class :character   Class :character   1st Qu.:118.50
##  Mode  :character   Mode  :character   Mode  :character   Median :241.76
##                                                           Mean   :307.59
##                                                           3rd Qu.:448.90
##                                                           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
```

```r
# Check the column names
names(smarket)
```

```
##  [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"
```

Data Cleaning

```r
# Let's check for missing values
colSums(is.na(smarket))
```

```
##            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
```

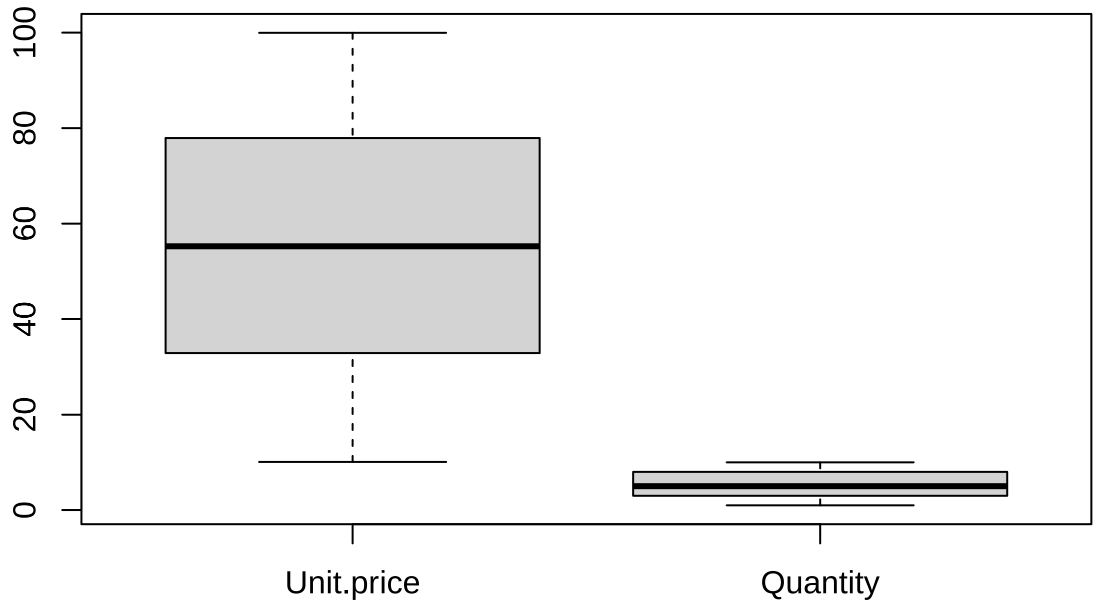
We have no missing values.

```r
# Check for duplicate values
sum(duplicated(smarket))
```

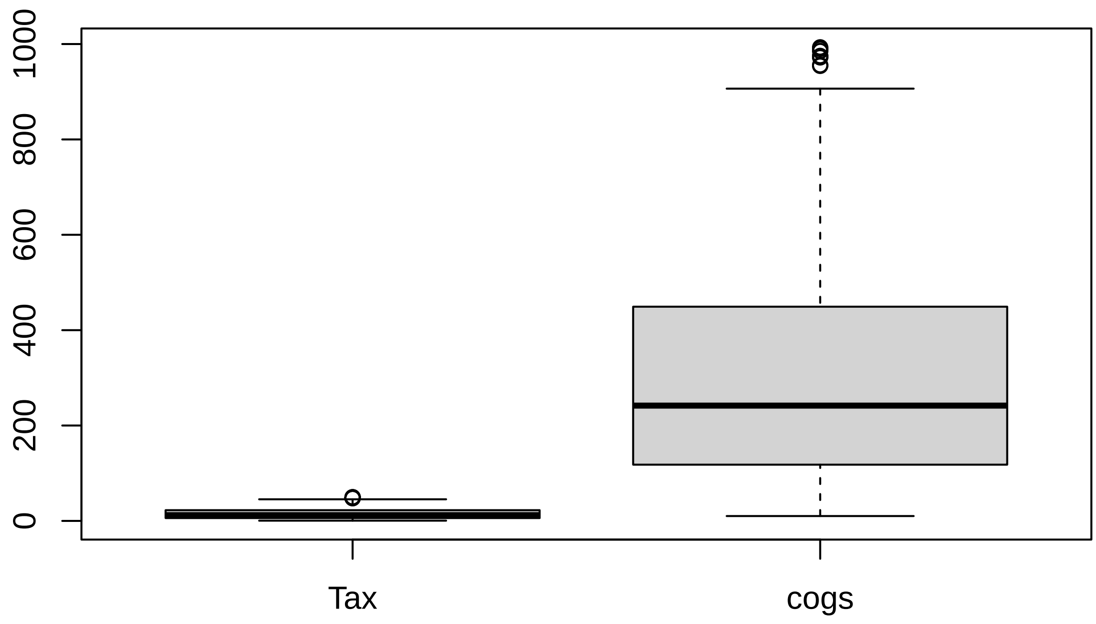
```
## [1] 0
```

We have no duplicate values

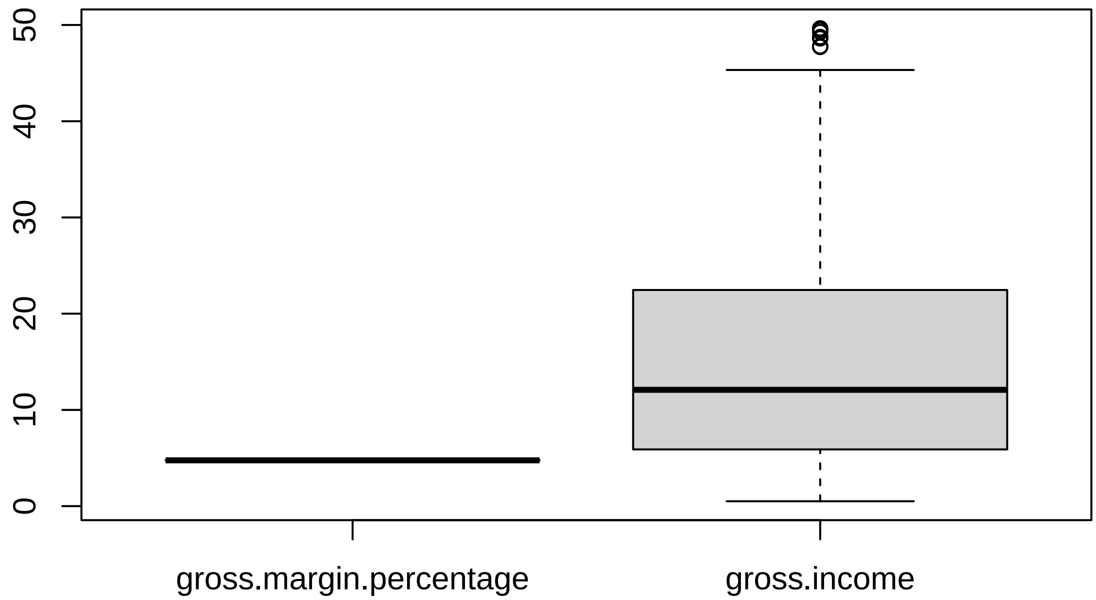
```r
# Checking for outliers in our numerical variables
boxplot(smarket[, c(6,7)])
```



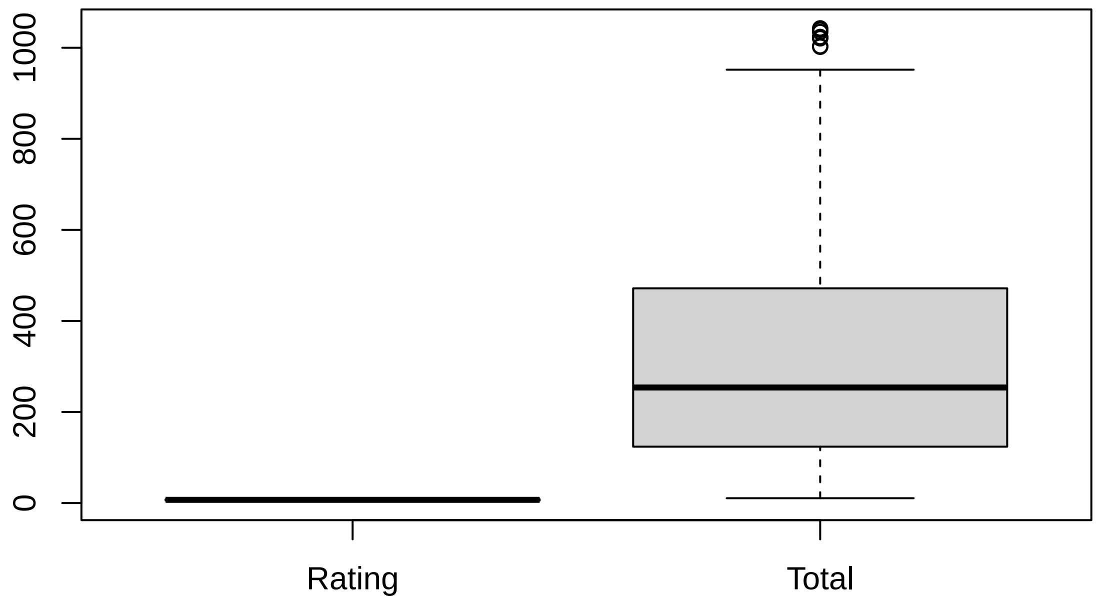

```r
# Checking for outliers in our numerical variables
boxplot(smarket[, c(8,12)])
```




```r
# Checking for outliers in our numerical variables
boxplot(smarket[, c(13,14)])
```




```r
# Checking for outliers in our numerical variables
boxplot(smarket[, c(15,16)])
```



There is presence of outliers in Tax, Cogs, Gross income and Total Variables

```
# Convert the date column from character datatype to date datatype
smarket$Date <- as.Date(smarket$Date, "%m/%d/%Y")
# Confirm changes made
glimpse(smarket)
```

```
## Rows: 1,000
## Columns: 16
## $ Invoice.ID              <chr> "750-67-8428", "226-31-3081", "631-41-3108", "~
## $ Branch                  <chr> "A", "C", "A", "A", "A", "C", "A", "C", "A", "~
## $ Customer.type           <chr> "Member", "Normal", "Normal", "Member", "Norma~
## $ Gender                  <chr> "Female", "Female", "Male", "Male", "Male", "M~
## $ Product.line            <chr> "Health and beauty", "Electronic accessories",~
## $ Unit.price              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 68.8~
## $ Quantity                <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10, 10~
## $ Tax                     <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085, 29~
## $ Date                    <date> 2019-01-05, 2019-03-08, 2019-03-03, 2019-01-2~
## $ Time                    <chr> "13:08", "10:29", "13:23", "20:33", "10:37", "~
## $ Payment                 <chr> "Ewallet", "Cash", "Credit card", "Ewallet", "~
## $ cogs                    <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597.73,~
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7619~
## $ gross.income            <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085, 29~
## $ Rating                  <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2, 5~
## $ Total                   <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634.378~
```

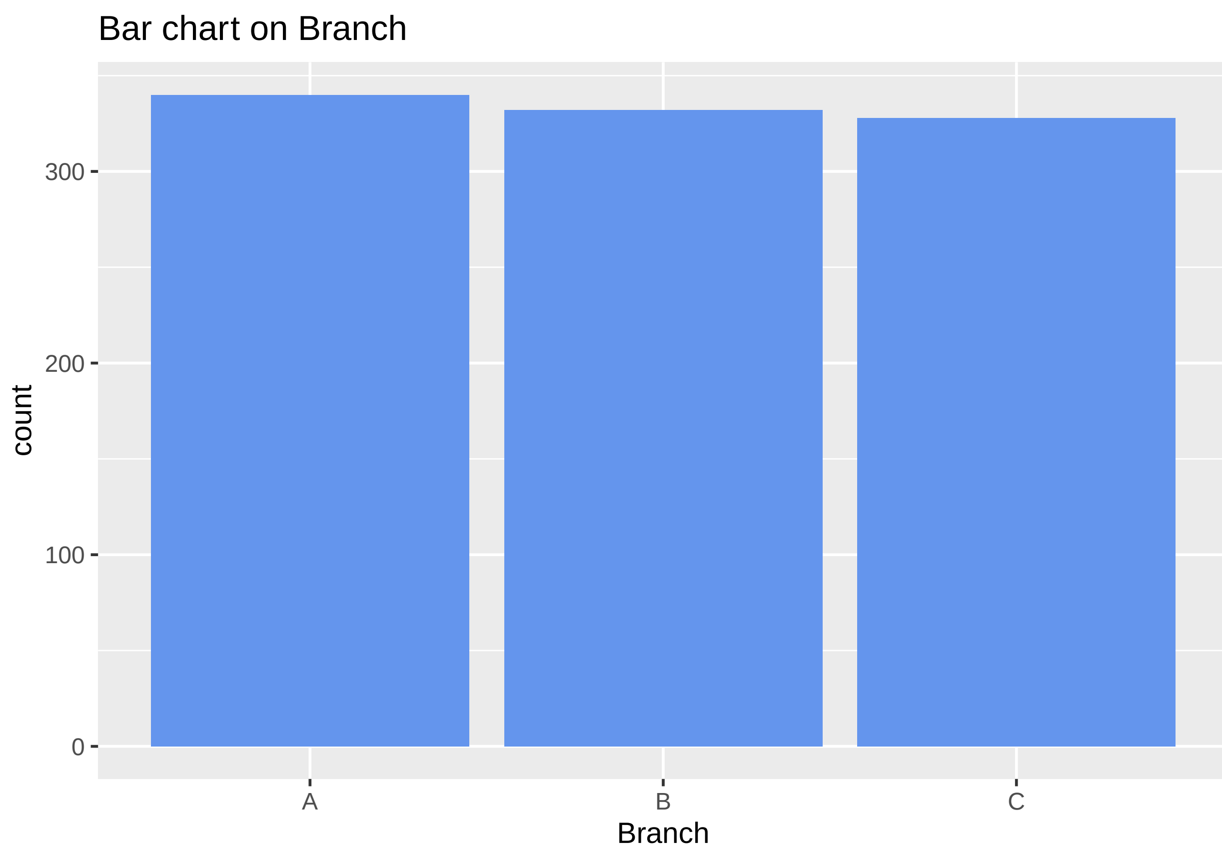
```r
# We will extract month from the date column
smarket$Month<- format(smarket$Date, "%m")
head(smarket)
```

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

```r
# Replace the numbers in months with the name of the month
smarket$Month[smarket$Month == "01"] <- "January"
smarket$Month[smarket$Month == "02"] <- "February"
smarket$Month[smarket$Month == "03"] <- "March"
```

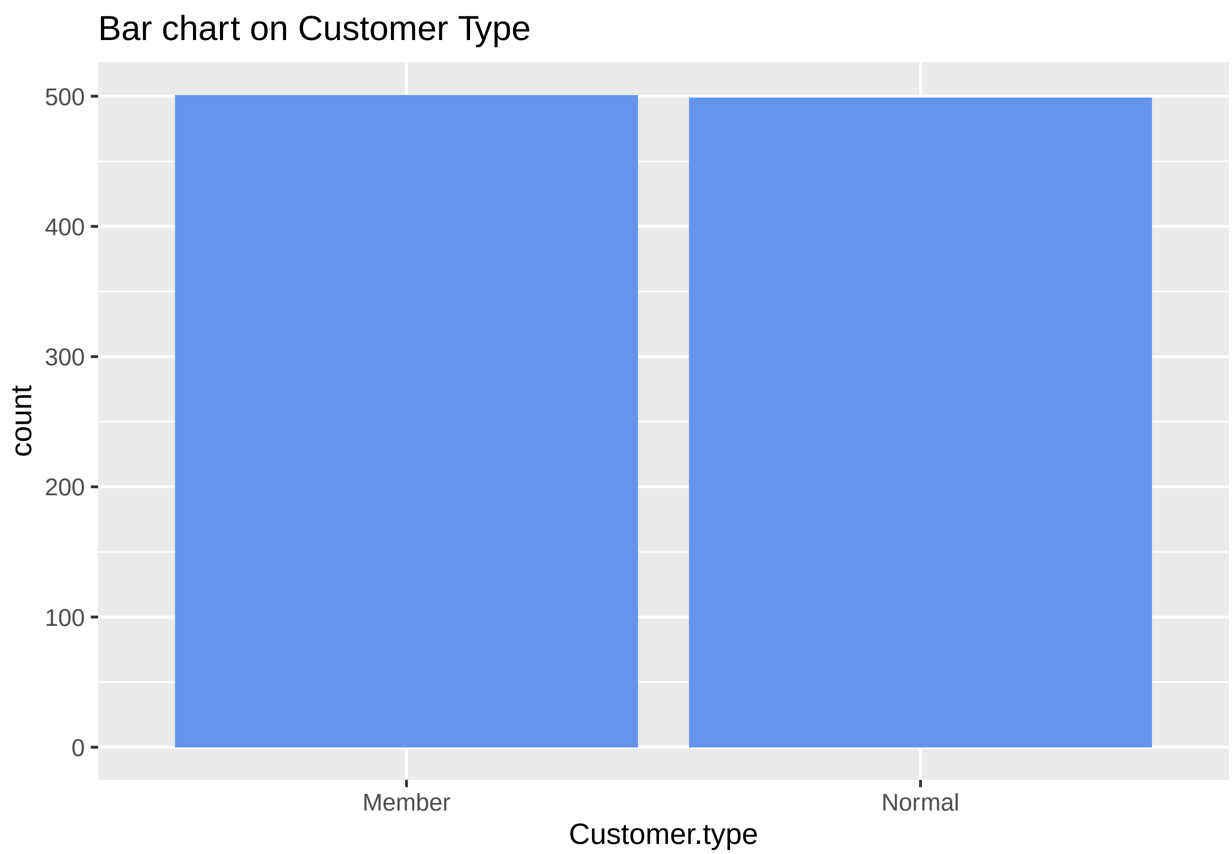
Univariate Analysis

```r
# Bar chart on revenue
gg.1 <- ggplot (data = smarket, aes(x= Branch)) +
  geom_bar(fill = "cornflowerblue")
gg.1 + ggtitle("Bar chart on Branch")
```



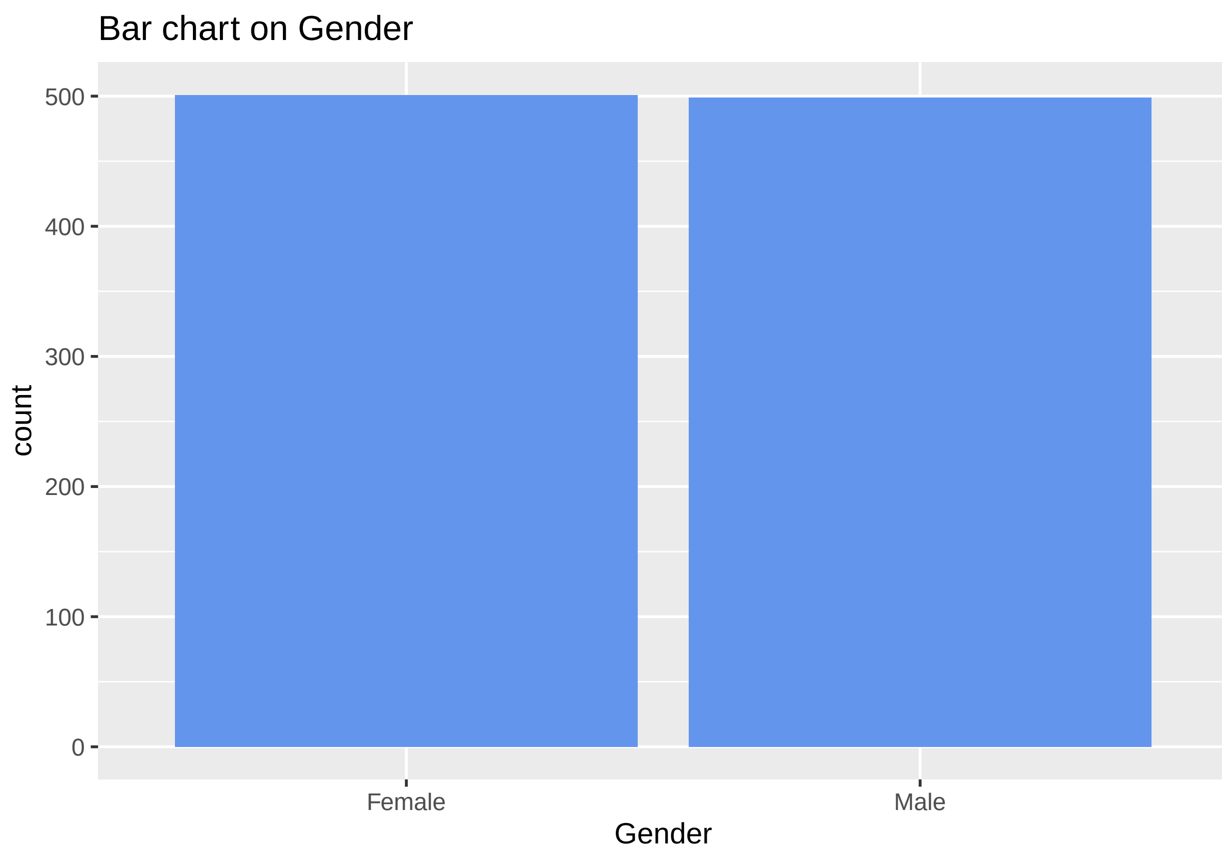
The number of branches in our data are not too different from each other

```
# Bar Chart on Customer Type
gg.2 <- ggplot (data = smarket, aes(x= Customer.type)) +
  geom_bar(fill = "cornflowerblue")
gg.2 + ggtitle("Bar chart on Customer Type")
```



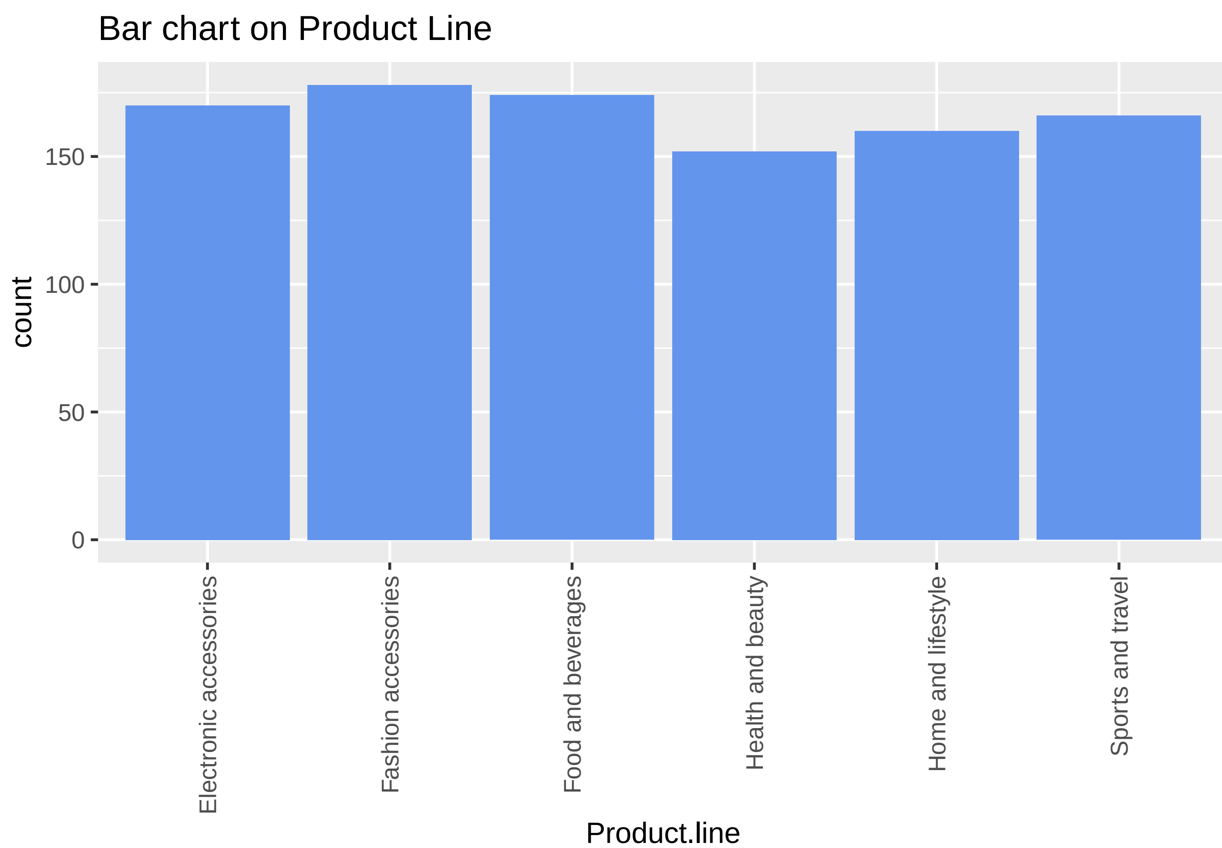
There isn't a big difference in the number of member and normal customers

```
# Bar Chart on Gender
gg.3 <- ggplot (data = smarket, aes(x= Gender)) +
  geom_bar(fill = "cornflowerblue")
gg.3 + ggtitle("Bar chart on Gender")
```



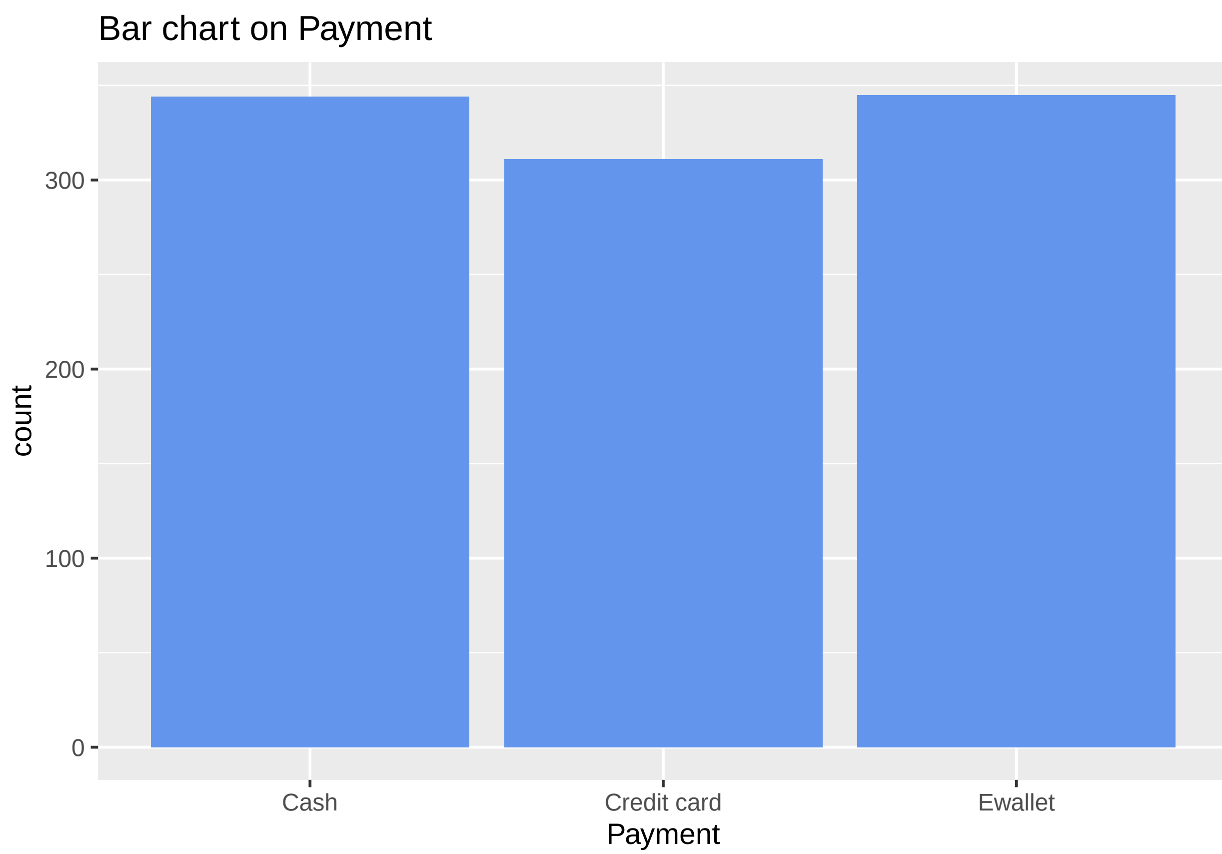
There isn't a big difference in the number of male and female customers

```
# Bar Chart on Product Line
gg.4 <- ggplot (data = smarket, aes(x= Product.line)) +
  geom_bar(fill = "cornflowerblue") + theme(axis.text.x = element_text(
    angle = 90, vjust =.5, hjust = 1
  ))
gg.4 + ggtitle("Bar chart on Product Line")
```



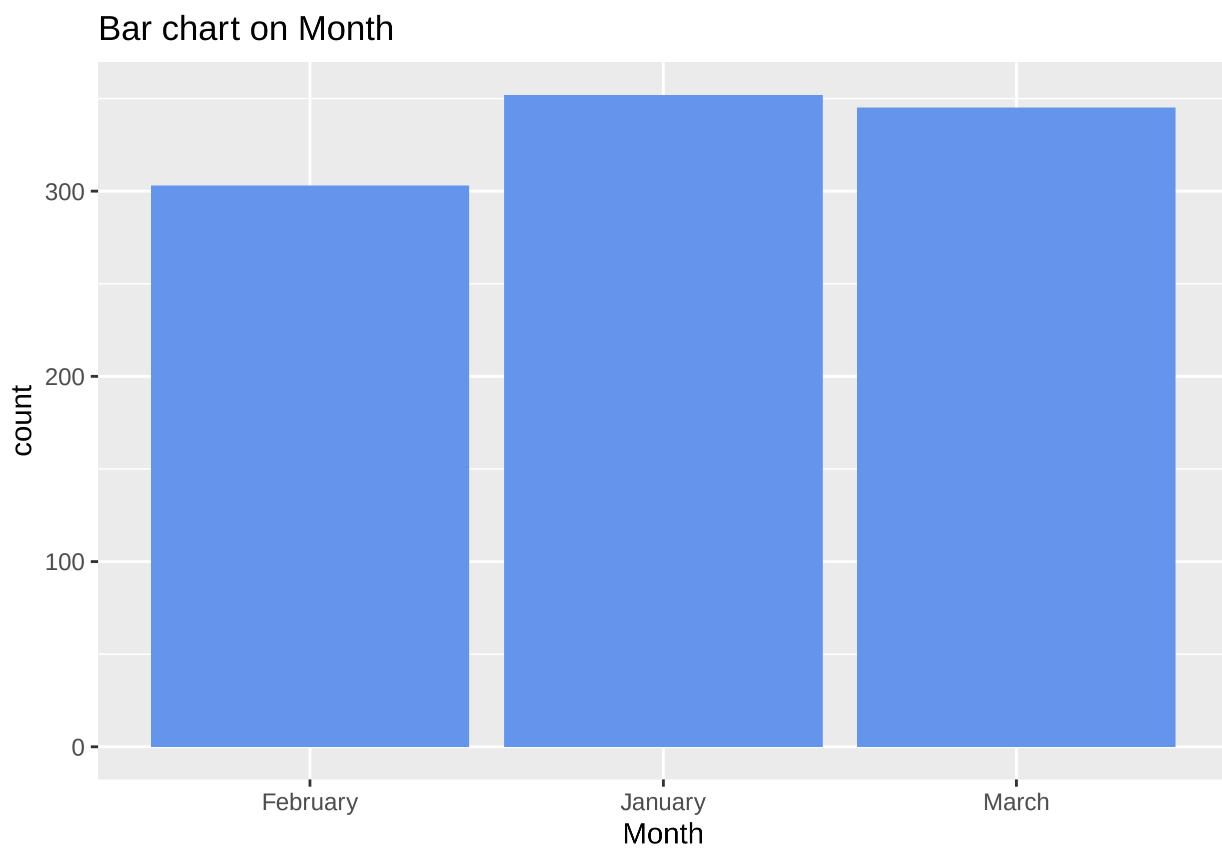
Fashion accessories is the most popular product followed closely by food and beverages

```
# Bar Chart on Payment
gg.5 <- ggplot (data = smarket, aes(x= Payment)) +
  geom_bar(fill = "cornflowerblue")
gg.5 + ggtitle("Bar chart on Payment")
```



There is no big difference in the number of customers who paid via Ewallet and cash

```
# Bar Chart on Month
gg.6<- ggplot (data = smarket, aes(x= Month)) +
  geom_bar(fill = "cornflowerblue")
gg.6 + ggtitle("Bar chart on Month")
```



January had the most transactions followed closely by March

```
# Frequency chart on Unit price
freq.1 <- table(smarket$Unit.price)
head(sort(freq.1, decreasing = T))
```

```
##
## 83.77  15.5  15.8 18.08 19.15 20.01
##     3     2     2     2     2     2
```

```
# Frequency chart on Quantity
freq.2 <- table(smarket$Quantity)
head(sort(freq.2, decreasing = T))
```

```
##
##  10   1   4   5   7   6
## 119 112 109 102 102  98
```

```
# Frequency chart on Tax
freq.3 <- table(smarket$Tax)
head(sort(freq.3, decreasing = T),n=15)
```

```
##
##   4.154   4.464   8.377  9.0045  10.326 10.3635   12.57  13.188  22.428   39.48
##       2       2       2       2       2       2       2       2       2       2
```

```
## 0.5085  0.6045   0.627   0.639   0.699
##      1       1       1       1       1
```

```r
# Frequency chart on cogs
freq.4 <- table(smarket$cogs)
head(sort(freq.4, decreasing = T), n=15)
```

```
##
##  83.08  89.28 167.54 180.09 206.52 207.27  251.4 263.76 448.56  789.6  10.17
##      2      2      2      2      2      2      2      2      2      2      1
##  12.09  12.54  12.78  13.98
##      1      1      1      1
```

```r
# Frequency chart on gross margin percentage
freq.5 <- table(smarket$gross.margin.percentage)
head(sort(freq.5, decreasing = T))
```

```
## 4.761904762
##        1000
```

```r
# Frequency chart on Rating
freq.5 <- table(smarket$Rating)
head(sort(freq.5, decreasing = T))
```

```
##
##   6 6.6 4.2 9.5   5 5.1
##  26  24  22  22  21  21
```

```r
# Frequency chart on Gross Income
freq.5 <- table(smarket$gross.income)
head(sort(freq.5, decreasing = T), n=10)
```

```
##
##   4.154   4.464   8.377  9.0045  10.326 10.3635   12.57  13.188  22.428   39.48
##       2       2       2       2       2       2       2       2       2       2
```

```r
# Frequency chart on Total
freq.5 <- table(smarket$Total)
head(sort(freq.5, decreasing = T), n=15)
```

```
##
##   87.234   93.744  175.917 189.0945  216.846 217.6335   263.97  276.948
##        2        2        2        2        2        2        2        2
##  470.988   829.08  10.6785  12.6945   13.167   13.419   14.679
##        2        2        1        1        1        1        1
```

```r
# Identify the mean, median, min, max and quantile of our numerical variables
summary(smarket[,c(6:8,12:16)])
```

```
##    Unit.price       Quantity          Tax                 cogs
##  Min.   :10.08   Min.   : 1.00   Min.   : 0.5085   Min.   : 10.17
##  1st Qu.:32.88   1st Qu.: 3.00   1st Qu.: 5.9249   1st Qu.:118.50
##  Median :55.23   Median : 5.00   Median :12.0880   Median :241.76
##  Mean   :55.67   Mean   : 5.51   Mean   :15.3794   Mean   :307.59
##  3rd Qu.:77.94   3rd Qu.: 8.00   3rd Qu.:22.4453   3rd Qu.:448.90
##  Max.   :99.96   Max.   :10.00   Max.   :49.6500   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
```

```r
# We'll also check the mode of our numerical variables
mode <- function(x) {
  u <- unique(x)
  tab <- tabulate(match(x, u))
  u[tab == max(tab)]
}

print("Mode of unit price is:")
```

```
## [1] "Mode of unit price is:"
```

```r
mode(smarket$Unit.price)
```

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

```r
print("Mode of tax is:")
```

```
## [1] "Mode of tax is:"
```

```r
mode(smarket$Tax)
```

```
##  [1] 39.4800  9.0045 10.3260 12.5700 10.3635 13.1880  4.1540  8.3770 22.4280
## [10]  4.4640
```

```r
print("Mode of COGS is:")
```

```
## [1] "Mode of COGS is:"
```

```r
mode(smarket$cogs)
```

```
##  [1] 789.60 180.09 206.52 251.40 207.27 263.76  83.08 167.54 448.56  89.28
```

```r
print("Mode of gross margin percentage is:")
```

```
## [1] "Mode of gross margin percentage is:"
```

```r
mode(smarket$gross.margin.percentage)
```

```
## [1] 4.761905
```

```r
print("Mode of Gross income:")
```

```
## [1] "Mode of Gross income:"
```

```r
mode(smarket$gross.income)
```

```
##  [1] 39.4800  9.0045 10.3260 12.5700 10.3635 13.1880  4.1540  8.3770 22.4280
## [10]  4.4640
```

```r
print("Mode of Rating is:")
```

```
## [1] "Mode of Rating is:"
```

```r
mode(smarket$Rating)
```

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

```r
print("Mode of Total is:")
```

```
## [1] "Mode of Total is:"
```

```r
mode(smarket$Total)
```

```
##  [1] 829.0800 189.0945 216.8460 263.9700 217.6335 276.9480  87.2340 175.9170
##  [9] 470.9880  93.7440
```

```r
# Check the variance of our numerical variables
print("Variance of unit price is:")
```

```
## [1] "Variance of unit price is:"
```

```r
var(smarket$Unit.price)
```

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

```r
print("Variance of tax is:")
```

```
## [1] "Variance of tax is:"
```

```r
var(smarket$Tax)
```

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

```r
print("Variance of COGS is:")
```

```
## [1] "Variance of COGS is:"
```

```r
var(smarket$cogs)
```

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

```r
print("Variance of gross margin percentage is:")
```

```
## [1] "Variance of gross margin percentage is:"
```

```r
var(smarket$gross.margin.percentage)
```

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

```r
print("Variance of Gross income:")
```

```
## [1] "Variance of Gross income:"
```

```r
var(smarket$gross.income)
```

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

```r
print("Variance of Rating is:")
```

```
## [1] "Variance of Rating is:"
```

```r
var(smarket$Rating)
```

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

```r
print("Variance of Total is:")
```

```
## [1] "Variance of Total is:"
```

```
var(smarket$Total)
```

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

```
# Check the skewness of our variables
print("The skewness of our variables:")
```

```
## [1] "The skewness of our variables:"
```

```
sapply(smarket[,c(6:8,12:16)],skewness)
```

```
##             Unit.price                Quantity                     Tax
##            0.007066827             0.012921628             0.891230392
##                   cogs gross.margin.percentage            gross.income
##            0.891230392                     NaN             0.891230392
##                 Rating                   Total
##            0.008996129             0.891230392
```

```
# Check the kurtosis of our variables
print("The kurtosis of our variables:")
```

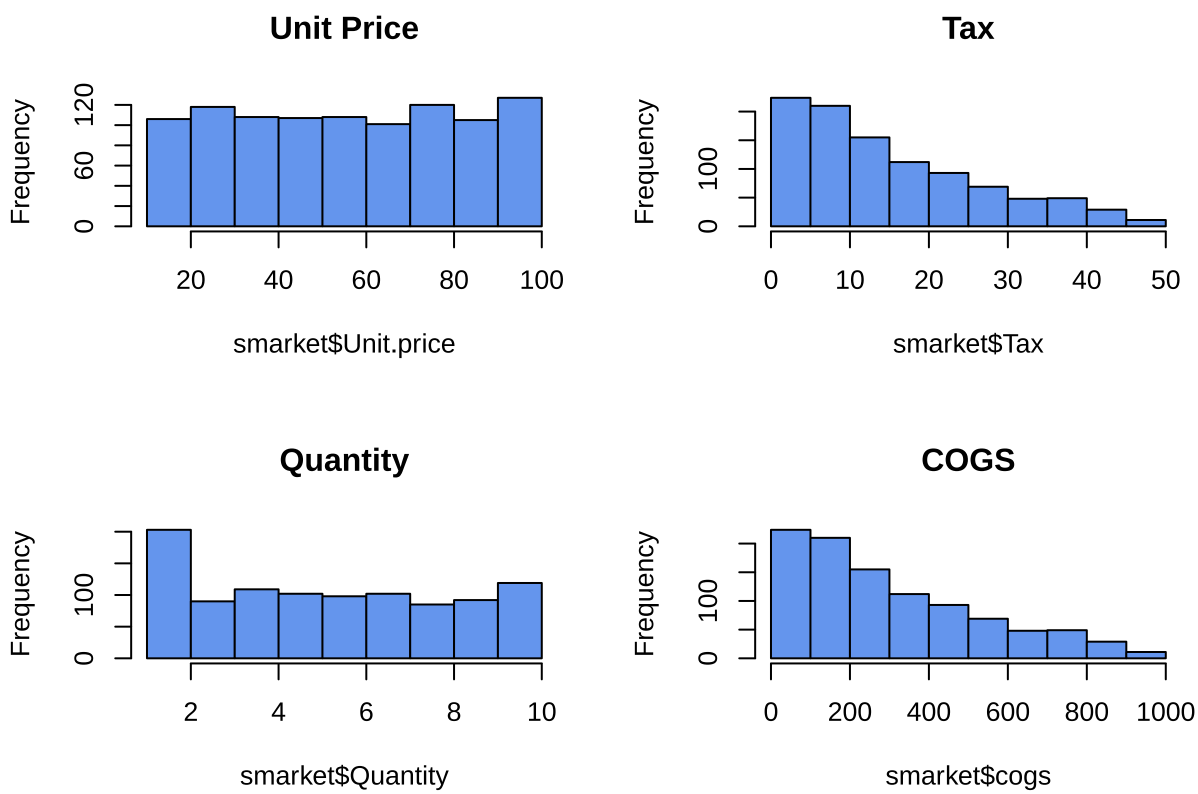
```
## [1] "The kurtosis of our variables:"
```

```
sapply(smarket[,c(6:8,12:16)],kurtosis)
```

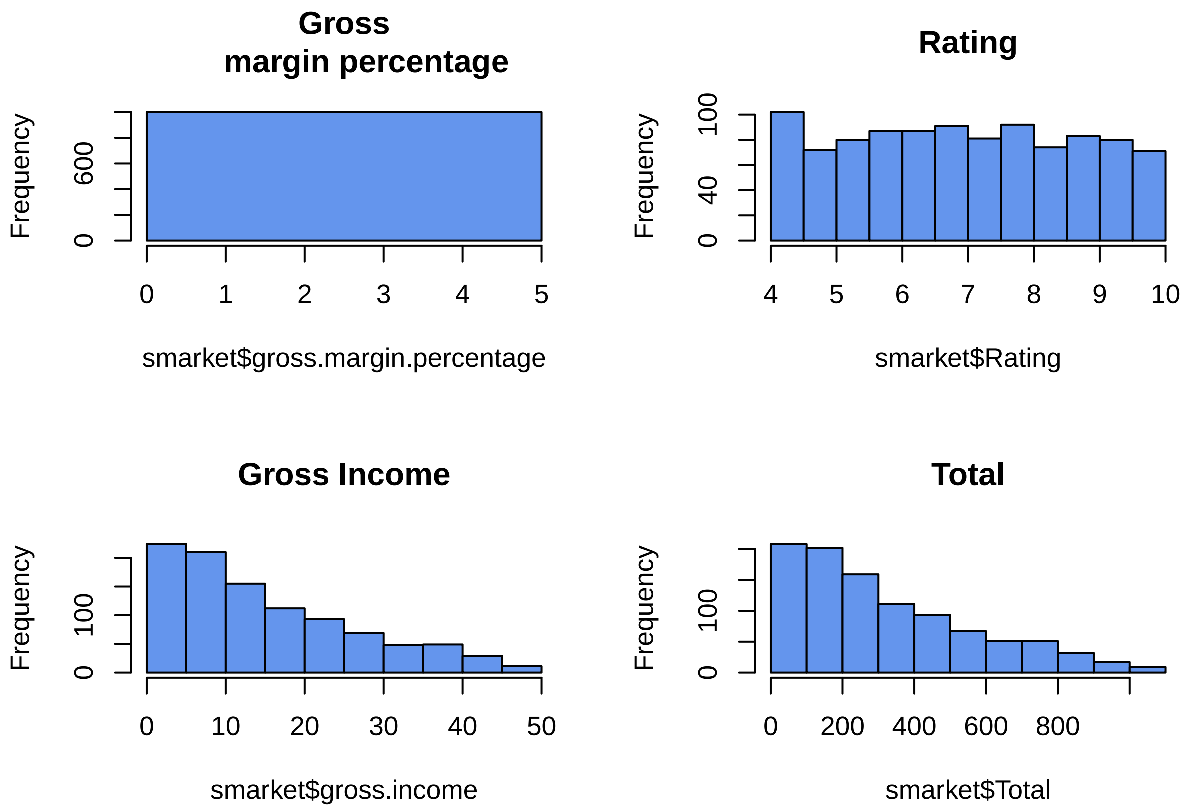
```
##             Unit.price                Quantity                     Tax
##               1.781499                1.784528                2.912530
##                   cogs gross.margin.percentage            gross.income
##               2.912530                     NaN                2.912530
##                 Rating                   Total
##               1.848169                2.912530
```

Our numerical variables do not have heavy tails

```
# Plot histograms of our numeric variables
par(mfcol=c(2,2))
hist(smarket$Unit.price, col = "cornflowerblue", main = "Unit Price")
hist(smarket$Quantity, col = "cornflowerblue",
     main = "Quantity")
hist(smarket$Tax, col = "cornflowerblue", main="Tax")
hist(smarket$cogs, col = "cornflowerblue",
     main ="COGS")
```




```r
# Plot histograms of our numeric variables
par(mfcol=c(2,2))
hist(smarket$gross.margin.percentage, col = "cornflowerblue", main = "Gross
     margin percentage")
hist(smarket$gross.income, col = "cornflowerblue",
     main = "Gross Income")
hist(smarket$Rating, col = "cornflowerblue", main="Rating")
hist(smarket$Total, col = "cornflowerblue",
     main ="Total")
```



Unit price quantity and Rating are symmetrical and the rest are moderately skewed.

Bivariate Analysis

```r
# Assess the correlation of our numerical variables
cor(smarket[,c(6:8,12:16)])
```

```
## Warning in cor(smarket[, c(6:8, 12:16)]): the standard deviation is zero
```

```
##                          Unit.price    Quantity        Tax       cogs
## Unit.price              1.000000000  0.01077756  0.6339621  0.6339621
## Quantity                0.010777564  1.00000000  0.7055102  0.7055102
## Tax                     0.633962089  0.70551019  1.0000000  1.0000000
## cogs                    0.633962089  0.70551019  1.0000000  1.0000000
## gross.margin.percentage          NA          NA         NA         NA
## gross.income            0.633962089  0.70551019  1.0000000  1.0000000
## Rating                 -0.008777507 -0.01581490 -0.0364417 -0.0364417
## Total                   0.633962089  0.70551019  1.0000000  1.0000000
##                        gross.margin.percentage gross.income       Rating
## Unit.price                                  NA    0.6339621 -0.008777507
## Quantity                                    NA    0.7055102 -0.015814905
## Tax                                         NA    1.0000000 -0.036441705
## cogs                                        NA    1.0000000 -0.036441705
## gross.margin.percentage                      1           NA           NA
```

```
## gross.income                             NA    1.0000000 -0.036441705
## Rating                                   NA   -0.0364417  1.000000000
## Total                                    NA    1.0000000 -0.036441705
##                               Total
## Unit.price                0.6339621
## Quantity                  0.7055102
## Tax                       1.0000000
## cogs                      1.0000000
## gross.margin.percentage          NA
## gross.income              1.0000000
## Rating                   -0.0364417
## Total                     1.0000000
```

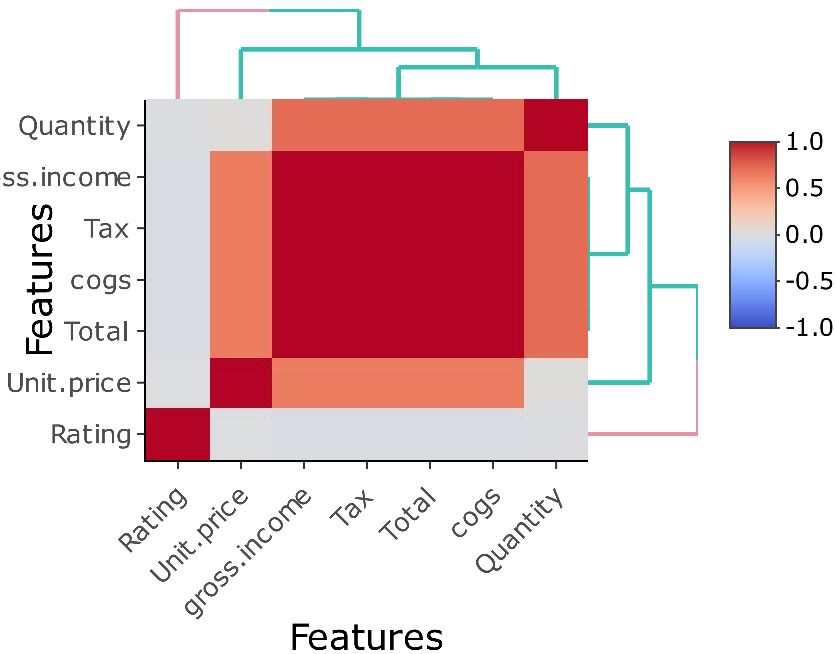
```
# Check for correlation without the gross.margin.percentage
cor(smarket[,c(6:8,12,14,15,16)])
```

```
##                Unit.price    Quantity        Tax       cogs gross.income
## Unit.price    1.000000000  0.01077756  0.6339621  0.6339621    0.6339621
## Quantity      0.010777564  1.00000000  0.7055102  0.7055102    0.7055102
## Tax           0.633962089  0.70551019  1.0000000  1.0000000    1.0000000
## cogs          0.633962089  0.70551019  1.0000000  1.0000000    1.0000000
## gross.income  0.633962089  0.70551019  1.0000000  1.0000000    1.0000000
## Rating       -0.008777507 -0.01581490 -0.0364417 -0.0364417   -0.0364417
## Total         0.633962089  0.70551019  1.0000000  1.0000000    1.0000000
##                    Rating      Total
## Unit.price   -0.008777507  0.6339621
## Quantity     -0.015814905  0.7055102
## Tax          -0.036441705  1.0000000
## cogs         -0.036441705  1.0000000
## gross.income -0.036441705  1.0000000
## Rating        1.000000000 -0.0364417
## Total        -0.036441705  1.0000000
```

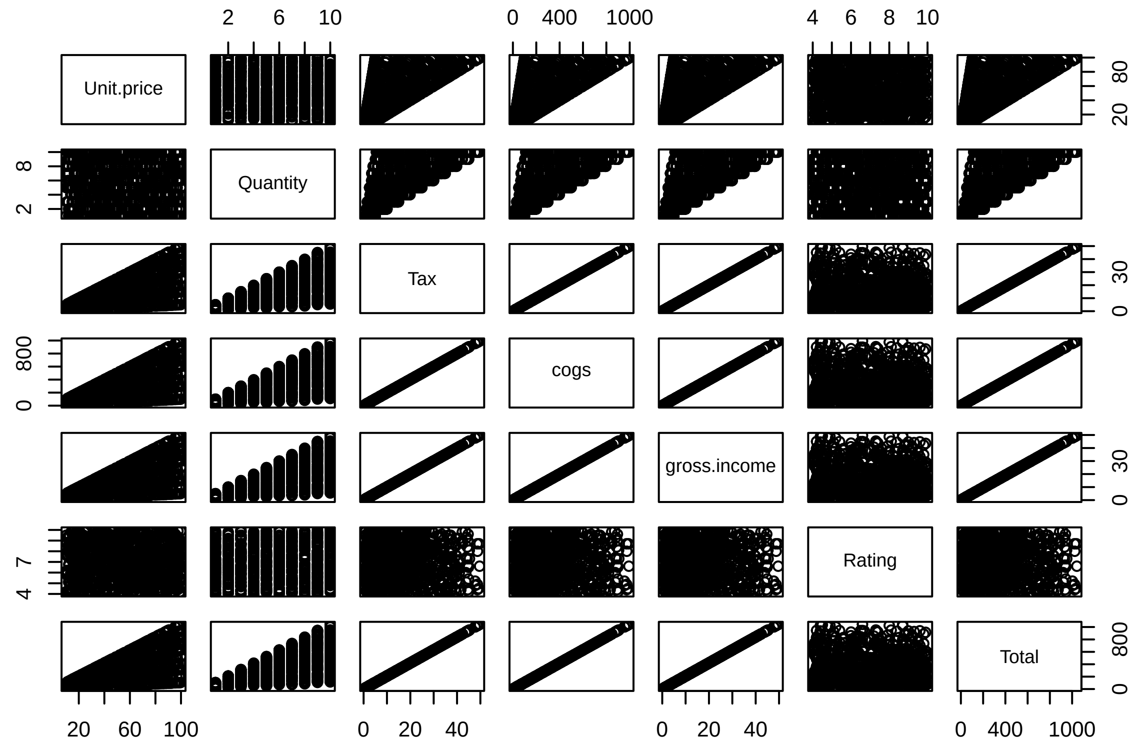
There is a strong positive correlation between quantity and Tax ,cog, Gross income and total Which makes sense because Quantity is used in the calculation of these variables

There's a weak negative correlation between Rating and Tax ,cog, Gross income and total

```
# Plotting corr heatmap
heatmaply_cor(x = cor(smarket[,c(6:8,12,14:16)]), xlab = "Features",
              ylab = "Features", k_col = 2, k_row = 2)
```



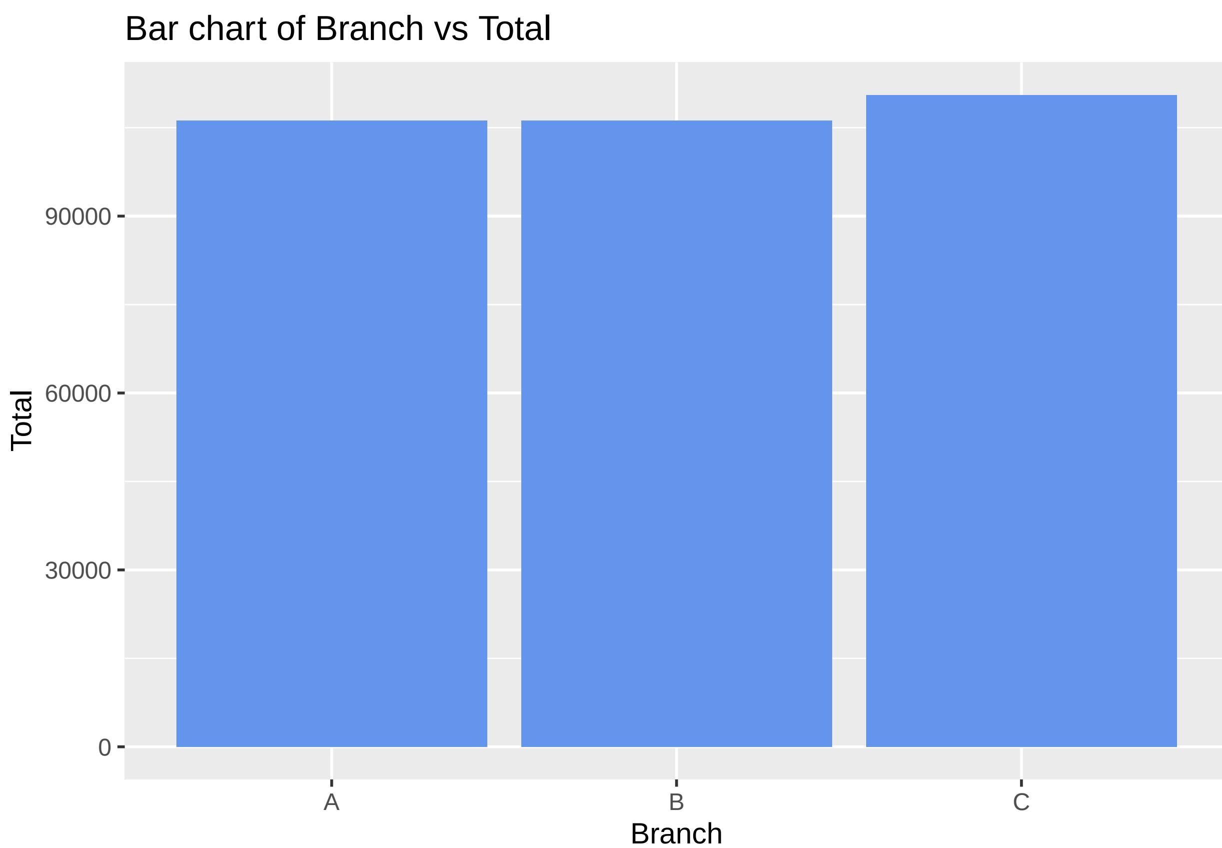

```
# Plot a pairplot
pairs(smarket[,c(6:8,12,14:16)])
```




```r
# Bar Chart of Branch vs Total
agg_1 <- aggregate(smarket$Total,list(smarket$Branch),sum)
agg_1
```

```
##   Group.1        x
## 1       A 106200.4
## 2       B 106197.7
## 3       C 110568.7
```

```r
ggplot(data = agg_1 , aes(x= Group.1, y =x)) + geom_col(fill ="cornflowerblue") +
  labs(y="Total", x="Branch")+ggtitle("Bar chart of Branch vs Total")
```

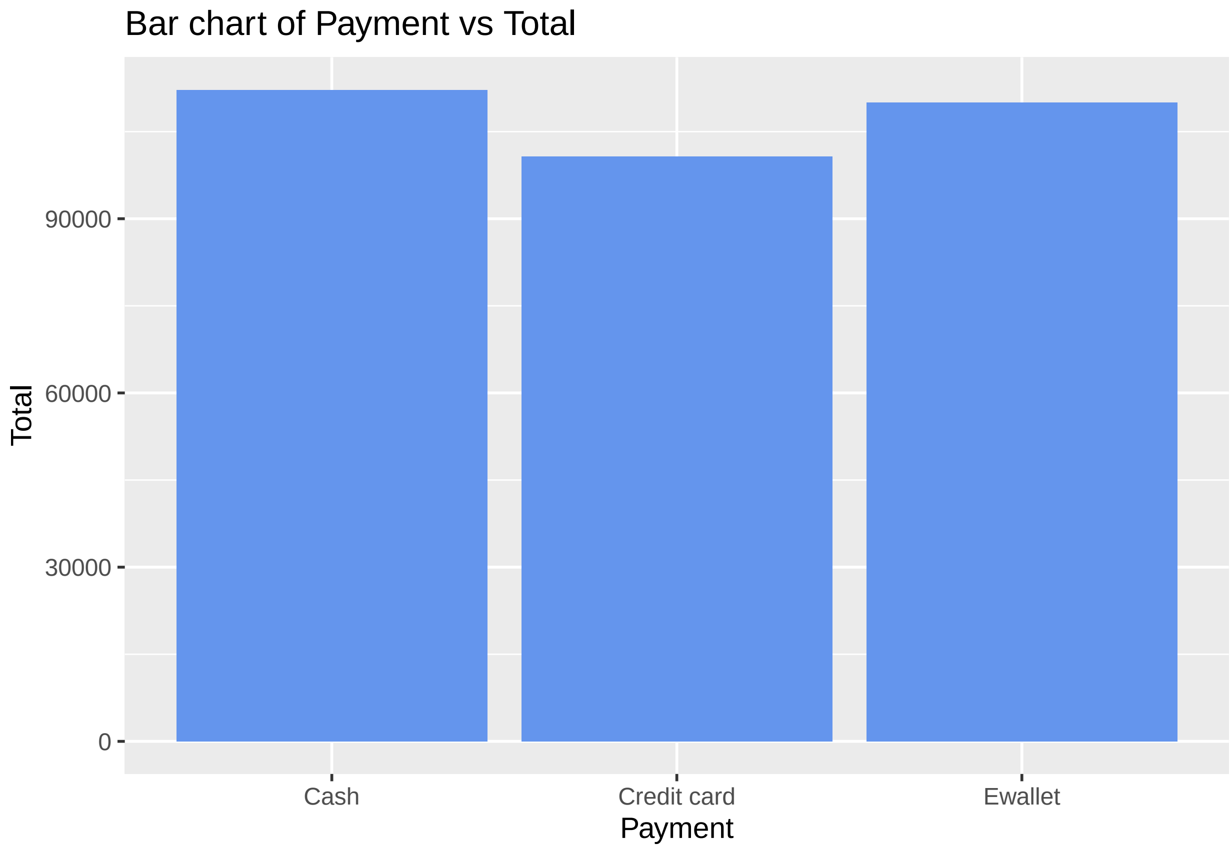


The branch with the highest number of sales is c

```
# Bar chart of Payment vs Total
agg_2 <- aggregate(smarket$Total,list(smarket$Payment),sum)
agg_2
```

```
##      Group.1        x
## 1       Cash 112206.6
## 2 Credit card 100767.1
## 3    Ewallet 109993.1
```

```
ggplot(data = agg_2 , aes(x= Group.1, y =x)) + geom_col(fill ="cornflowerblue") +
  labs(y="Total", x="Payment")+ggtitle("Bar chart of Payment vs Total")
```

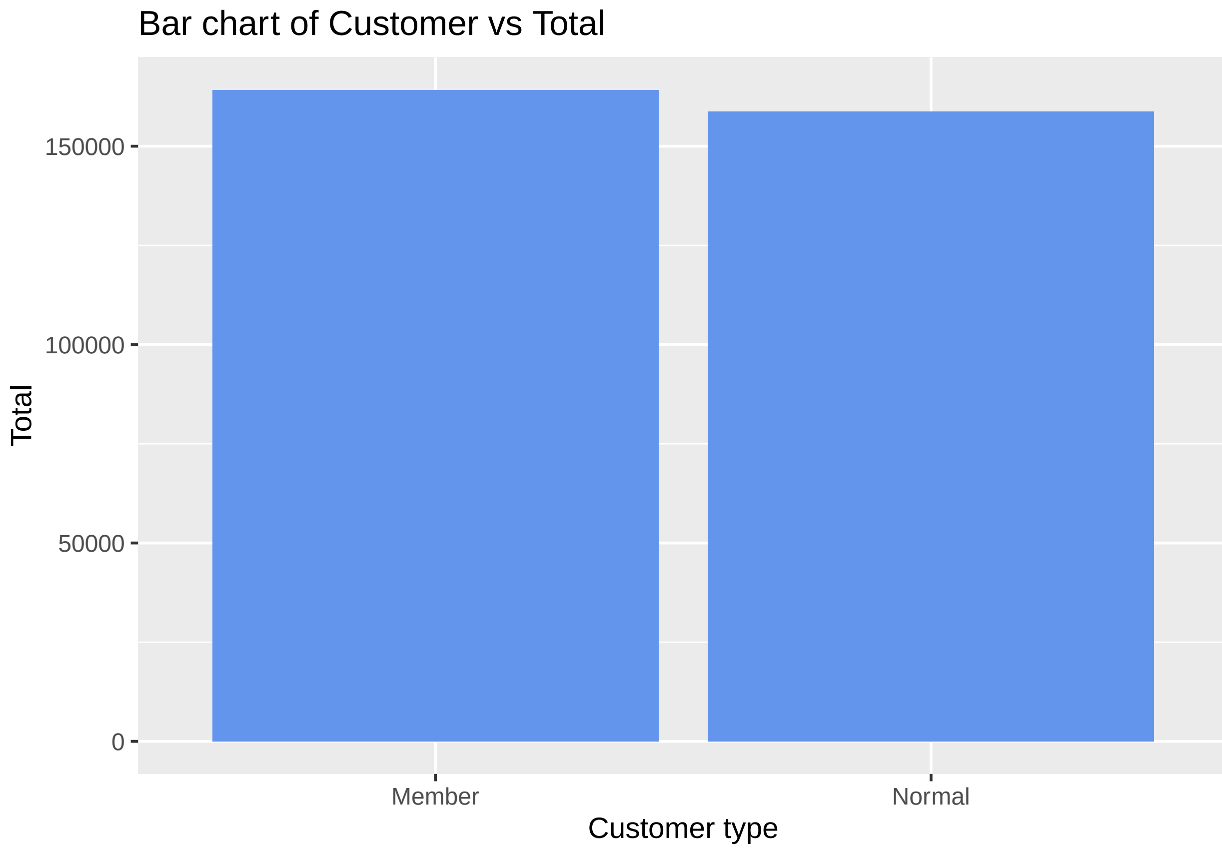


The Payment method with the highest number of sales is cash

```
# Bar chart of Customer type vs Total
agg_3 <- aggregate(smarket$Total,list(smarket$Customer.type),sum)
agg_3
```

```
##   Group.1        x
## 1  Member 164223.4
## 2  Normal 158743.3
```

```
ggplot(data = agg_3 , aes(x= Group.1, y =x)) + geom_col(fill ="cornflowerblue") +
  labs(y="Total", x="Customer type")+ggtitle("Bar chart of Customer vs Total")
```

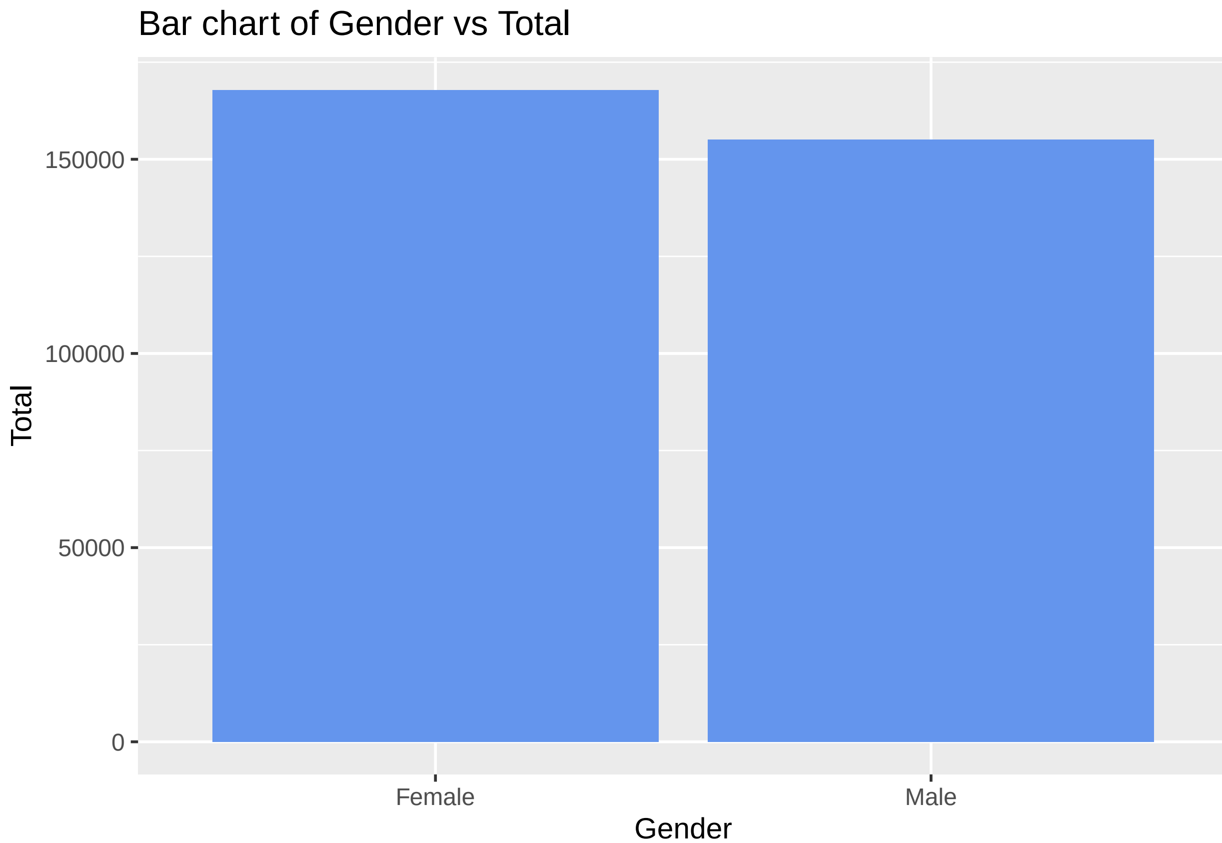


Member customer type have the highest number of sales

```
# Bar chart of Gender vs Total
agg_4 <- aggregate(smarket$Total,list(smarket$Gender),sum)
agg_4
```

```
##   Group.1        x
## 1  Female 167882.9
## 2    Male 155083.8
```

```
ggplot(data = agg_4 , aes(x= Group.1, y =x)) + geom_col(fill ="cornflowerblue") +
  labs(y="Total", x="Gender")+ggtitle("Bar chart of Gender vs Total")
```

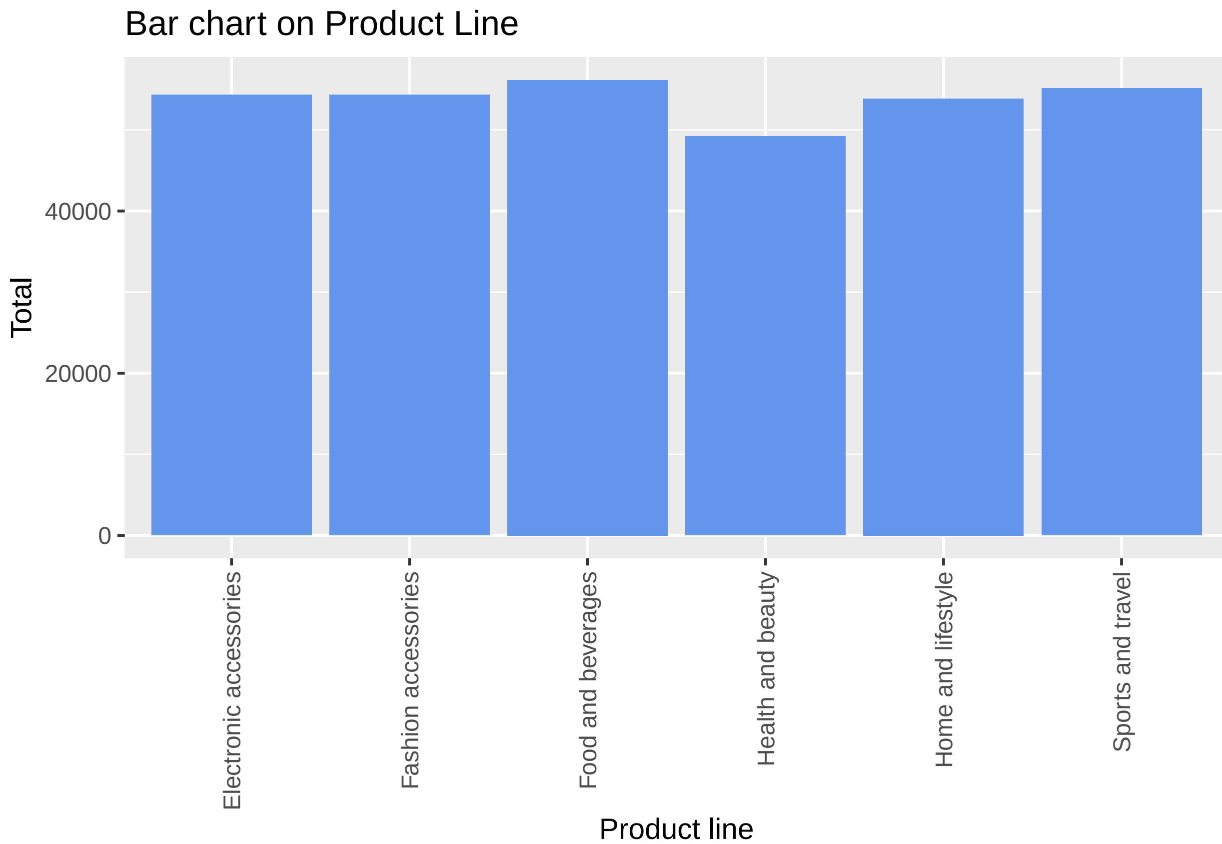


Female customers have highest number of sales

```
# Bar chart of Product line vs Total
agg_5 <- aggregate(smarket$Total,list(smarket$Product.line),sum)
agg_5
```

```
##                 Group.1        x
## 1 Electronic accessories 54337.53
## 2    Fashion accessories 54305.89
## 3     Food and beverages 56144.84
## 4      Health and beauty 49193.74
## 5     Home and lifestyle 53861.91
## 6      Sports and travel 55122.83
```

```
gg.agg <- ggplot (data = agg_5, aes(x= Group.1, y =x)) +
  geom_col(fill = "cornflowerblue") + theme(axis.text.x = element_text(
    angle = 90, vjust =.5, hjust = 1
  )) + labs(y="Total", x="Product line")+ggtitle("Bar chart of Product line vs Total")
gg.agg + ggtitle("Bar chart on Product Line")
```

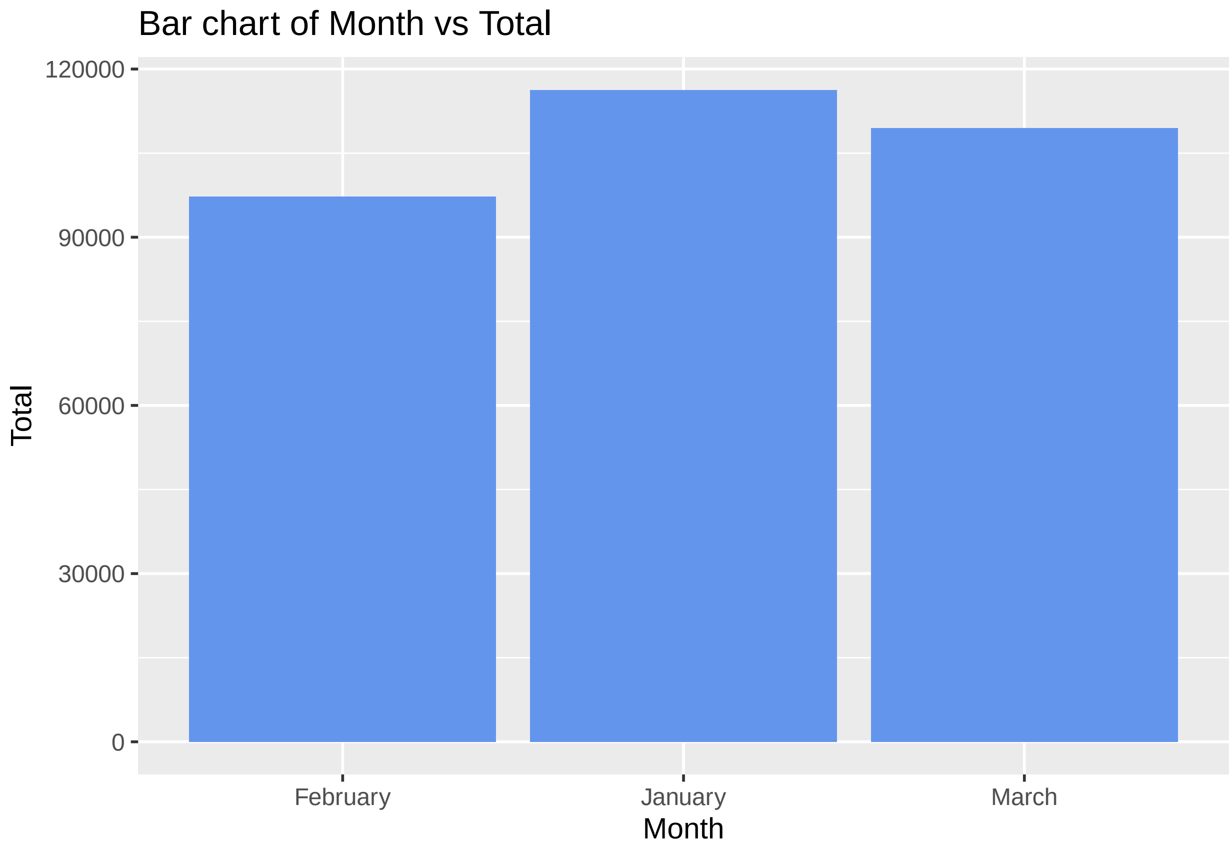


Food and beverages have the highest number of sales

```
# Bar chart of Month vs Total
agg_6 <- aggregate(smarket$Total,list(smarket$Month),sum)
agg_6
```

```
##    Group.1         x
## 1 February  97219.37
## 2  January 116291.87
## 3    March 109455.51
```

```
ggplot(data = agg_6 , aes(x= Group.1, y =x)) + geom_col(fill ="cornflowerblue") +
  labs(y="Total", x="Month")+ggtitle("Bar chart of Month vs Total")
```



January is the month with the highest number of sales

Data Preprocessing

```
# We will start with converting our categorical data to numerical data type
# Convert our variables to factor datatype
final.market <- smarket
final.market$Branch<- as.factor(final.market$Branch)
final.market$Product.line <- as.factor(final.market$Product.line)
final.market$Month <- as.factor(final.market$Month)
final.market$Customer.type <- as.factor(final.market$Customer.type)
final.market$Gender <- as.factor(final.market$Gender)
final.market$Payment<- as.factor(final.market$Payment)
```

```
# Convert our variables from factor to numeric datatype
final.market$Branch<- as.numeric(final.market$Branch)
final.market$Product.line <- as.numeric(final.market$Product.line)
final.market$Month <- as.numeric(final.market$Month)
final.market$Customer.type <- as.numeric(final.market$Customer.type)
final.market$Gender <- as.numeric(final.market$Gender)
final.market$Payment<- as.numeric(final.market$Payment)
```

```r
# Confirm changes made
head(final.market)
```

```
##    Invoice.ID Branch Customer.type Gender Product.line Unit.price Quantity
## 1 750-67-8428      1             1      1            4      74.69        7
## 2 226-31-3081      3             2      1            1      15.28        5
## 3 631-41-3108      1             2      2            5      46.33        7
## 4 123-19-1176      1             1      2            4      58.22        8
## 5 373-73-7910      1             2      2            6      86.31        7
## 6 699-14-3026      3             2      2            1      85.39        7
##       Tax       Date  Time Payment   cogs gross.margin.percentage gross.income
## 1 26.1415 2019-01-05 13:08       3 522.83                4.761905      26.1415
## 2  3.8200 2019-03-08 10:29       1  76.40                4.761905       3.8200
## 3 16.2155 2019-03-03 13:23       2 324.31                4.761905      16.2155
## 4 23.2880 2019-01-27 20:33       3 465.76                4.761905      23.2880
## 5 30.2085 2019-02-08 10:37       3 604.17                4.761905      30.2085
## 6 29.8865 2019-03-25 18:30       3 597.73                4.761905      29.8865
##   Rating    Total Month
## 1    9.1 548.9715     2
## 2    9.6  80.2200     3
## 3    7.4 340.5255     3
## 4    8.4 489.0480     2
## 5    5.3 634.3785     1
## 6    4.1 627.6165     3
```

```r
# We will carry out our analysis without invoive variable because it's a
# unique variables, gross.margin.percentage varible only has one value hence
# 0 variance, instead of using the date and time variable we will use month
final <- final.market[,c(-1,-9,-10,-13)]
# View the new dataset
head(final)
```

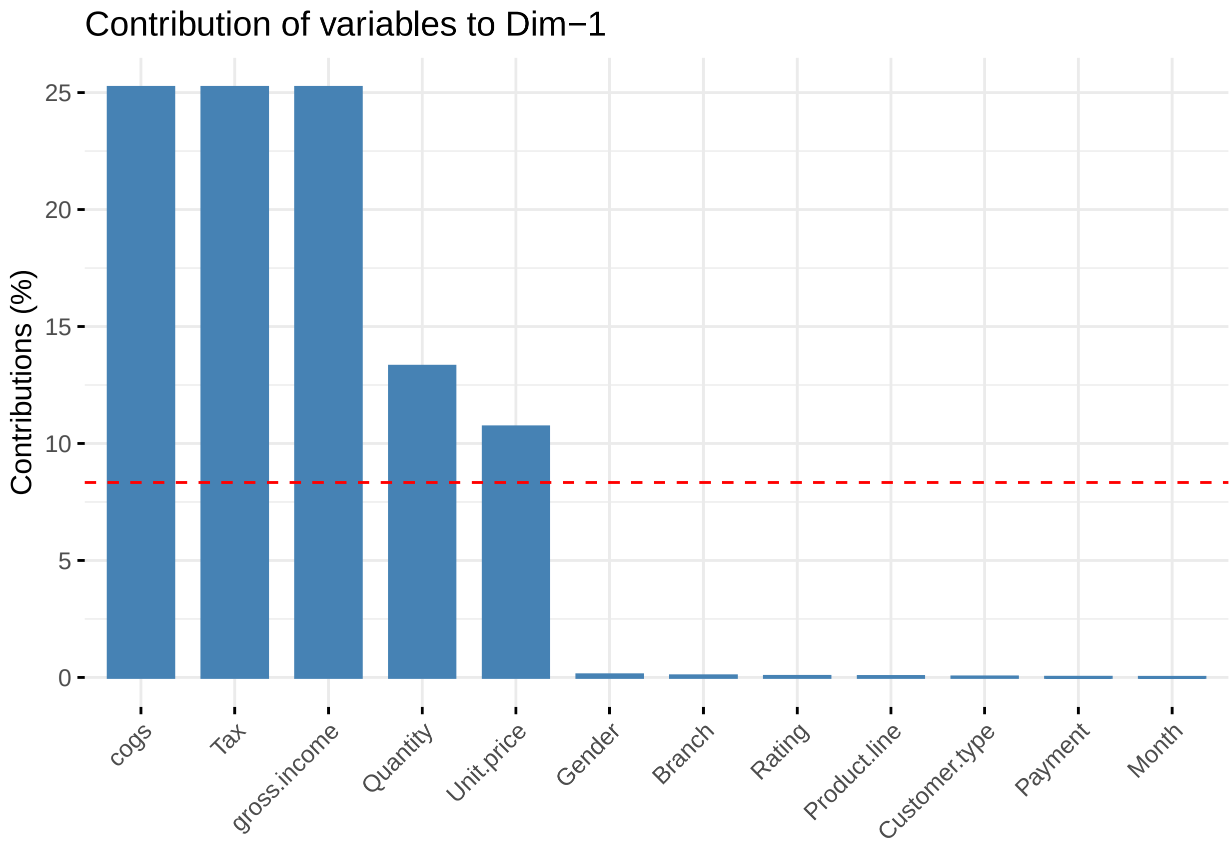
```
##   Branch Customer.type Gender Product.line Unit.price Quantity     Tax Payment
## 1      1             1      1            4      74.69        7 26.1415       3
## 2      3             2      1            1      15.28        5  3.8200       1
## 3      1             2      2            5      46.33        7 16.2155       2
## 4      1             1      2            4      58.22        8 23.2880       3
## 5      1             2      2            6      86.31        7 30.2085       3
## 6      3             2      2            1      85.39        7 29.8865       3
##     cogs gross.income Rating    Total Month
## 1 522.83      26.1415    9.1 548.9715     2
## 2  76.40       3.8200    9.6  80.2200     3
## 3 324.31      16.2155    7.4 340.5255     3
## 4 465.76      23.2880    8.4 489.0480     2
## 5 604.17      30.2085    5.3 634.3785     1
## 6 597.73      29.8865    4.1 627.6165     3
```

Performing PCA

```r
# We will drop our target variable - Total
pc <- prcomp(final[,-12],
             center = TRUE,
            scale. = TRUE)
print(pc)
```

```
## Standard deviations (1, .., p=12):
##  [1] 1.983660e+00 1.090093e+00 1.032622e+00 1.015016e+00 1.002936e+00
##  [6] 9.781046e-01 9.692678e-01 9.587693e-01 9.352167e-01 2.903498e-01
## [11] 2.765730e-16 1.102259e-16
##
## Rotation (n x k) = (12 x 12):
##                        PC1          PC2          PC3          PC4          PC5
## Branch          0.026774168 -0.456915788  0.068198717 -0.122355947  0.342867825
## Customer.type -0.015653235  0.210938688  0.497527438 -0.527332781 -0.003188021
## Gender        -0.033921782  0.432326114  0.451928013  0.255674367  0.165085840
## Product.line   0.020484525  0.317065786 -0.453401637  0.369809260 -0.187350387
## Unit.price     0.327318905  0.015696627  0.313427148  0.473556921  0.243654837
## Quantity       0.364739308 -0.003631654 -0.247499214 -0.415892161 -0.249483694
## Tax            0.502219382  0.015581910  0.017448193 -0.005177041 -0.003152543
## Payment       -0.009578868  0.407874734  0.009007581 -0.011879593 -0.354185239
## cogs           0.502219382  0.015581910  0.017448193 -0.005177041 -0.003152543
## gross.income   0.502219382  0.015581910  0.017448193 -0.005177041 -0.003152543
## Rating        -0.021696121 -0.194800583  0.406076613  0.089763920 -0.696333778
## Month         -0.007745303  0.503983357 -0.112331213 -0.315149727  0.296834072
##                        PC6          PC7          PC8          PC9         PC10
## Branch          0.290516084 -0.704528997 -0.002625143  0.269196241  0.009823824
## Customer.type -0.244734649 -0.028949152 -0.579934108  0.180378169  0.006581350
## Gender        -0.007070214 -0.003314004  0.481188348  0.532413886 -0.003749010
## Product.line  -0.351060899 -0.428454215 -0.342893139  0.311798733  0.002578331
## Unit.price     0.021188780 -0.034313634 -0.265371817 -0.316894873  0.581622373
## Quantity      -0.024426212  0.049372101  0.259403179  0.283448533  0.647935537
## Tax           -0.000627059  0.005272838  0.008167088 -0.001355379 -0.283598864
## Payment        0.811348866 -0.117723572 -0.182603915 -0.048843359  0.001147339
## cogs          -0.000627059  0.005272838  0.008167088 -0.001355379 -0.283598864
## gross.income  -0.000627059  0.005272838  0.008167088 -0.001355379 -0.283598864
## Rating        -0.224047826 -0.373733521  0.239599700 -0.237049387 -0.016459111
## Month         -0.151262748 -0.402470893  0.293064990 -0.524371124  0.013314096
##                       PC11          PC12
## Branch        -1.363486e-18 -5.513776e-18
## Customer.type  8.586149e-17  1.510125e-16
## Gender        -1.224434e-16 -1.004989e-16
## Product.line  -4.724803e-17 -9.802207e-18
## Unit.price     3.385859e-16 -3.728213e-16
## Quantity       4.162725e-16 -2.963019e-16
## Tax           -6.898069e-01  4.368444e-01
## Payment        6.174747e-17  9.294328e-18
## cogs           7.232218e-01  3.789681e-01
## gross.income  -3.341492e-02 -8.158125e-01
## Rating        -1.255332e-16  1.411814e-17
## Month         -1.111792e-16 -5.816152e-18
```

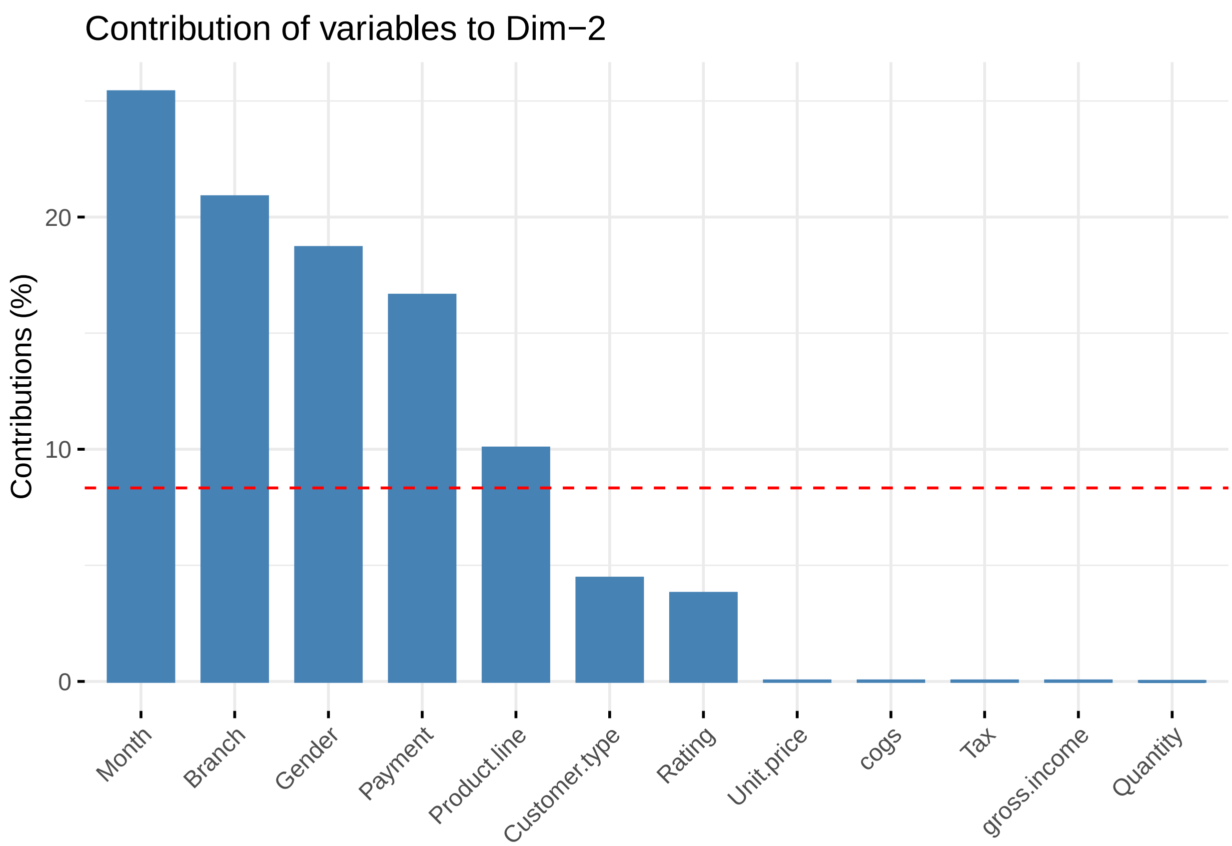
```r
# Contribution of variables of PC1
fviz_contrib(pc, choice = "var",axes = 1, Top = 10)
```



The larger the value of the contribution, the more the variable contributes to the component.

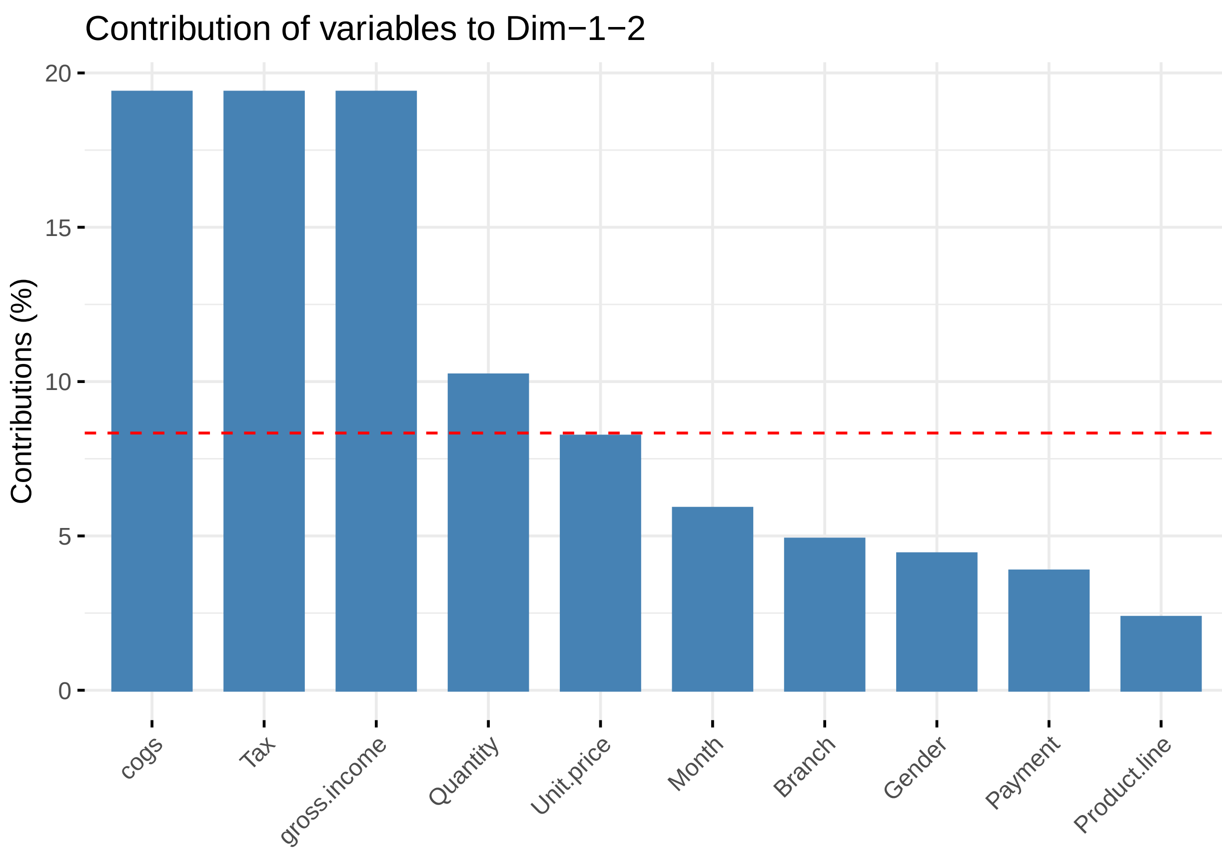
cogs, Tax and gross income contribute the most to PC1

```r
# Contribution of variables of PC2
fviz_contrib(pc, choice = "var",axes = 2, Top = 10)
```



Month contributes the most to PC2

```
# Contribution of variables to both PC1 and PC2
fviz_contrib(pc,choice = "var",axes = 1:2 , top = 10)
```

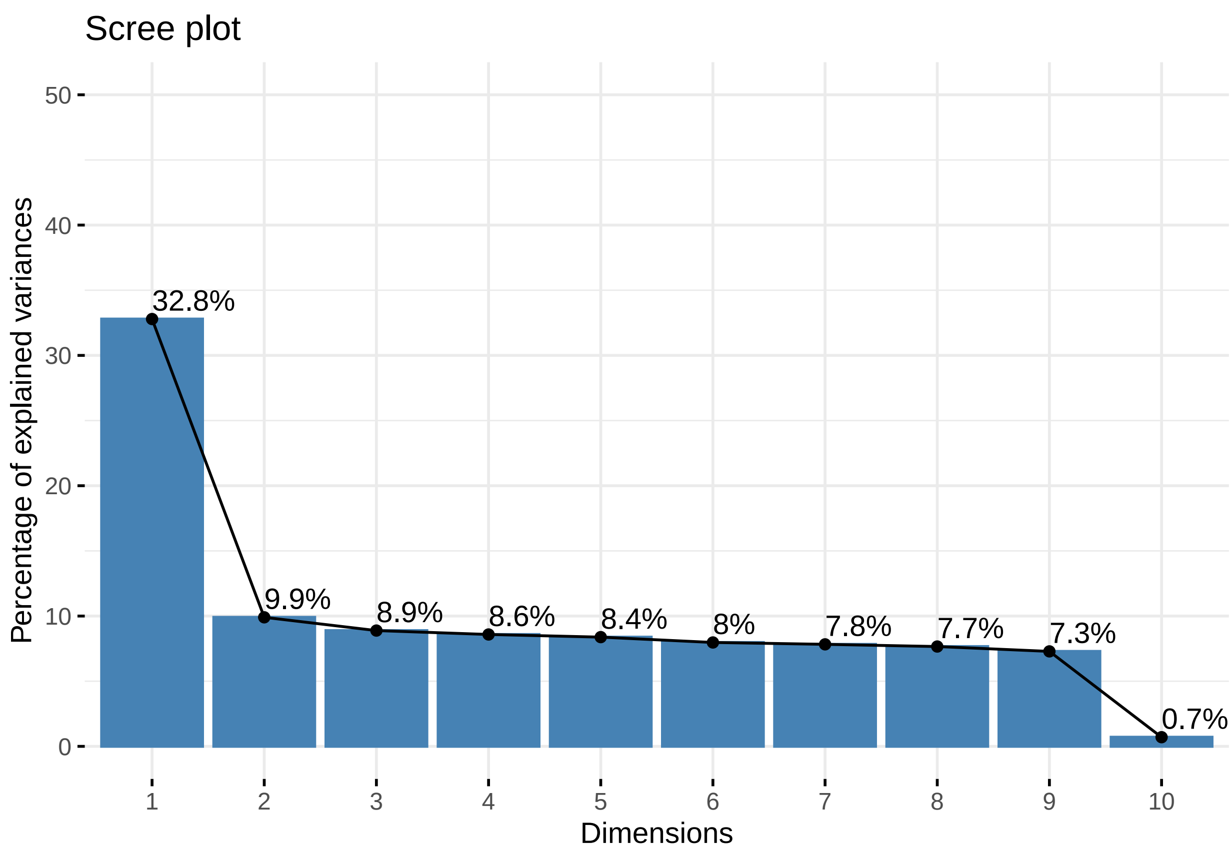


cogs, tax and income contribute the most to PC1 and PC2

```
# Summary of our PCA
summary(pc)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.9837 1.09009 1.03262 1.01502 1.00294 0.97810 0.96927
## Proportion of Variance 0.3279 0.09903 0.08886 0.08585 0.08382 0.07972 0.07829
## Cumulative Proportion  0.3279 0.42693 0.51579 0.60165 0.68547 0.76520 0.84349
##                           PC8     PC9    PC10      PC11      PC12
## Standard deviation     0.9588 0.93522 0.29035 2.766e-16 1.102e-16
## Proportion of Variance 0.0766 0.07289 0.00703 0.000e+00 0.000e+00
## Cumulative Proportion  0.9201 0.99297 1.00000 1.000e+00 1.000e+00
```

```
# Plot a scree plot
# It displays the total variance explained by each principal component
fviz_eig(pc, addlabels = T, ylim = c(0,50))
```



We might want to stop at the 8th principal component because 92.01% of the information contained in the data are retained by the first 8 principal components

```
# Plotting a PCA plot
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:plotly':
##
##     arrange, mutate, rename, summarise

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

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

```
## Loading required package: scales
```

```
##
## Attaching package: 'scales'
```

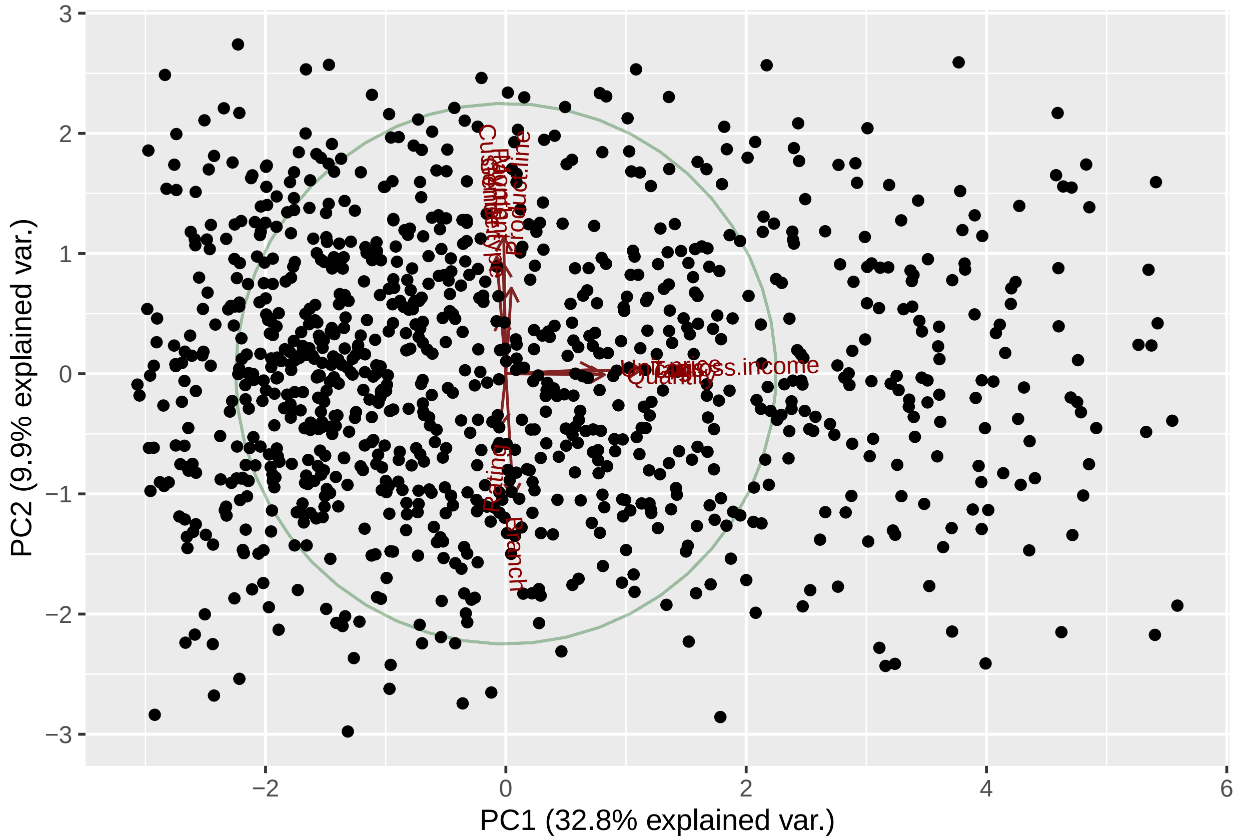
```
## The following object is masked from 'package:viridis':
##
##     viridis_pal
```

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

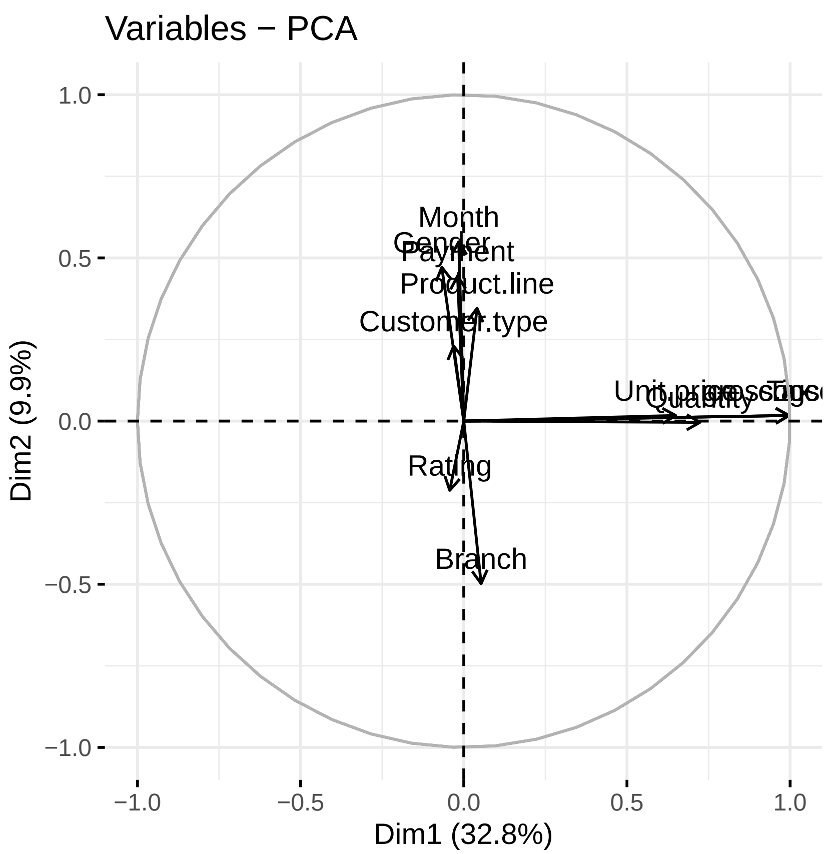
```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
## Loading required package: grid
```

```
gg <- ggbiplot(pcobj = pc,
               choices = c(1,2),
               circle = TRUE,
               ellipse = TRUE,
               scale = 0
              )
print(gg)
```




```r
# Another visualization for better visibility
fviz_pca_var(pc, col.var = "black")
```



Positively correlated variables are grouped together and negatively correlated variables are positioned on opposite sides of the plot.

t-SNE

```
# Load the t-SNE library
library(Rtsne)
```

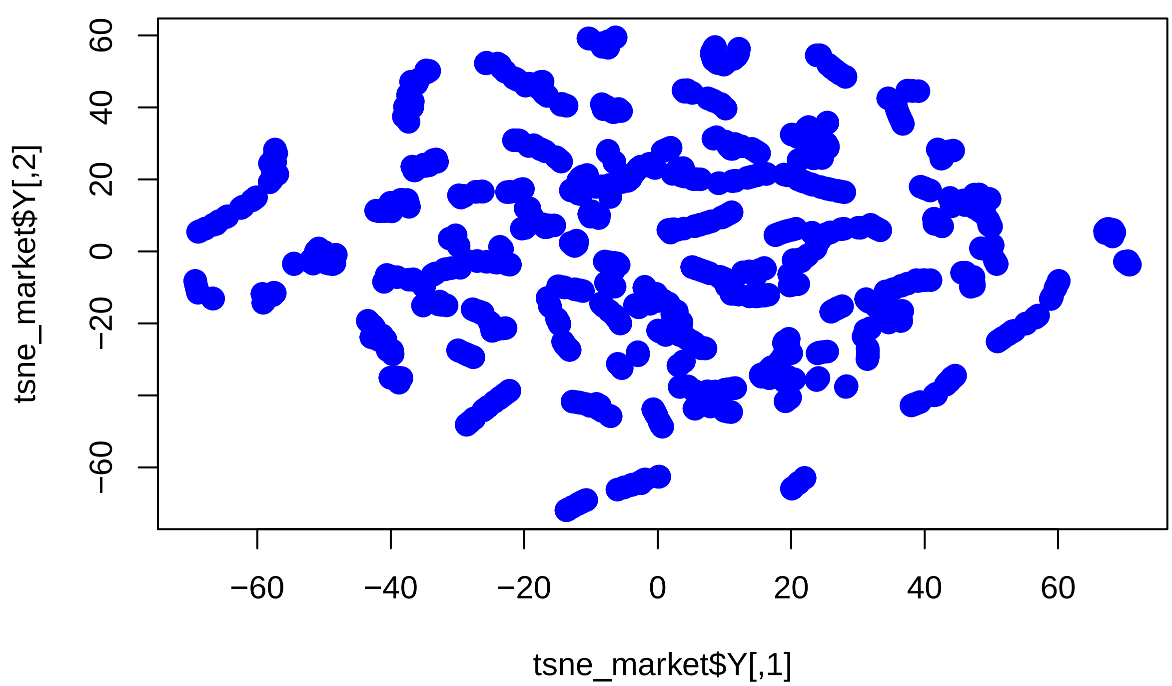
```
## Warning: package 'Rtsne' was built under R version 4.1.3
```

```
# Run t-SNE algorithm
tsne_market <- Rtsne(final[,-12], perplexity=5,verbose=TRUE, max_iter = 1000)
```

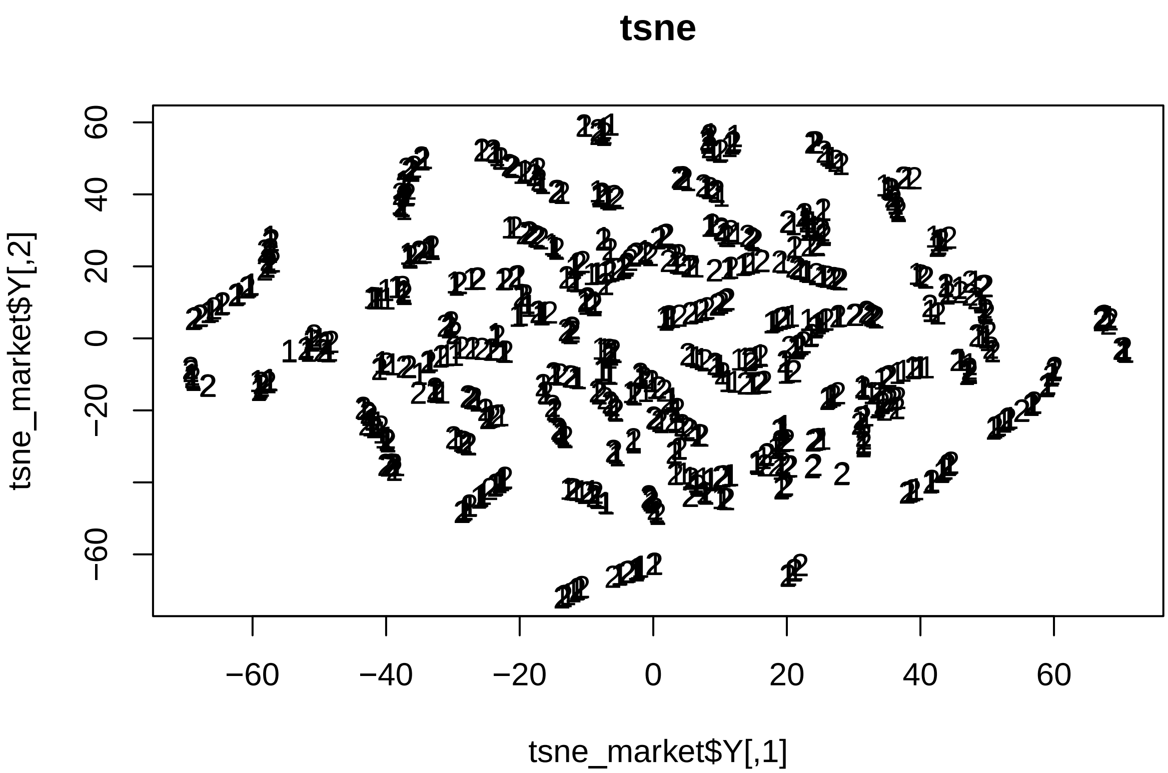
```
## Performing PCA
## Read the 1000 x 12 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 5.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.07 seconds (sparsity = 0.017276)!
## Learning embedding...
## Iteration 50: error is 83.786814 (50 iterations in 0.25 seconds)
## Iteration 100: error is 69.158822 (50 iterations in 0.16 seconds)
```

```
## Iteration 150: error is 64.568296 (50 iterations in 0.17 seconds)
## Iteration 200: error is 62.120352 (50 iterations in 0.17 seconds)
## Iteration 250: error is 60.534927 (50 iterations in 0.18 seconds)
## Iteration 300: error is 1.302719 (50 iterations in 0.23 seconds)
## Iteration 350: error is 0.828179 (50 iterations in 0.20 seconds)
## Iteration 400: error is 0.663733 (50 iterations in 0.20 seconds)
## Iteration 450: error is 0.592105 (50 iterations in 0.19 seconds)
## Iteration 500: error is 0.555627 (50 iterations in 0.55 seconds)
## Iteration 550: error is 0.534034 (50 iterations in 0.30 seconds)
## Iteration 600: error is 0.516290 (50 iterations in 0.18 seconds)
## Iteration 650: error is 0.500967 (50 iterations in 0.19 seconds)
## Iteration 700: error is 0.490888 (50 iterations in 0.18 seconds)
## Iteration 750: error is 0.481636 (50 iterations in 0.18 seconds)
## Iteration 800: error is 0.475822 (50 iterations in 0.18 seconds)
## Iteration 850: error is 0.470482 (50 iterations in 0.18 seconds)
## Iteration 900: error is 0.466741 (50 iterations in 0.17 seconds)
## Iteration 950: error is 0.463328 (50 iterations in 0.17 seconds)
## Iteration 1000: error is 0.459364 (50 iterations in 0.17 seconds)
## Fitting performed in 4.21 seconds.
```

```
# Generate the t_SNE plot
plot(tsne_market$Y, col = "blue", pch = 19, cex = 1.5)
```




```
# Generate the t_SNE plot with Customer type as our label
plot(tsne_market$Y, t='n',main = 'tsne')
text(tsne_market$Y,labels= final$Customer.type)
```



t-SNE is able to find patterns where PCA is unable to.