R Notebook

##Introduction

a) Specifying the Question

The main objective of the study is to use the data provided to aid Carrefour's marketing team in formulating strategies to boost sales.

b) Defining the Metrics for Success

• Determining and visualising the descriptive statistics of the variables in the datasets provided.

• Carrying out principal component analysis.

• Carrying out feature selection.

• Performing association analysis.

• Identifying possibly fraudelent transactions.

c) Understanding the context

Sales and Marketing teams aim to maximise a business' profit. Data-driven insights allow for the planning of more targeted and effective campaigns.

d) Recording the Experimental Design

• Determine the main objectives.

• Load and preview the datasets.

• Understand the data.

• Prepare the datasets - Identify outliers, anomalies, duplicates, missing values, and determine how deal with them, drop unnecessary columns etc.

• Analyse the data using univariate, bivariate, and multivariate analysis techniques.

• Carry out dimesnionality reduction, feature selection, associative analysis and anomaly detection on the respective datasets

• Conclusion and recommendations

e) Data Relevance

The datasets provided were relevant to the research question as they had relevant details on the sales at Carrefour.

##Loading the dataset

```
#loading some required libraries
library(readr)
library(data.table)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(psych)

##
## Attaching package: 'psych'

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

library(Rtsne)
library(tidyverse)

## ── Attaching packages
## ───────────────────────────────────────
## tidyverse 1.3.2 ──

## ✓ tibble  3.1.7     ✓ dplyr   1.0.9
## ✓ tidyr   1.2.0     ✓ stringr 1.4.0
## ✓ purrr   0.3.4     ✓ forcats 0.5.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## ✖ psych::%+%()       masks ggplot2::%+%()
## ✖ psych::alpha()     masks ggplot2::alpha()
## ✖ dplyr::between()   masks data.table::between()
## ✖ dplyr::filter()    masks stats::filter()
## ✖ dplyr::first()     masks data.table::first()
## ✖ dplyr::lag()       masks stats::lag()
## ✖ dplyr::last()      masks data.table::last()
## ✖ purrr::lift()      masks caret::lift()
## ✖ purrr::transpose() masks data.table::transpose()

library(factoextra)

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

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, th
en dplyr:
## library(plyr); library(dplyr)
## -------------------------------------------------------------------------
----
##
## Attaching package: 'plyr'
##
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
##
## The following object is masked from 'package:purrr':
##
##     compact
##
## Loading required package: scales
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following objects are masked from 'package:psych':
##
##     alpha, rescale
##
## The following object is masked from 'package:readr':
##
##     col_factor
##
## Loading required package: grid
```

```
df <- fread("http://bit.ly/CarreFourDataset")
df <- data.frame(df)
```

##Checking the Data

Determining the no. of records in the dataset:

```
dim(df)

## [1] 1000   16

#the dataset has 1000 rows and 16  columns
```

Previewing the top of the dataset:

```
head(df)
```

```
##    Invoice.ID Branch Customer.type Gender         Product.line Unit.pric
e
## 1 750-67-8428      A        Member Female      Health and beauty      74.6
9
## 2 226-31-3081      C        Normal Female Electronic accessories      15.2
8
## 3 631-41-3108      A        Normal   Male     Home and lifestyle      46.3
3
## 4 123-19-1176      A        Member   Male      Health and beauty      58.2
2
## 5 373-73-7910      A        Normal   Male      Sports and travel      86.3
1
## 6 699-14-3026      C        Normal   Male Electronic accessories      85.3
9
##   Quantity     Tax      Date  Time     Payment   cogs gross.margin.percent
age
## 1        7 26.1415  1/5/2019 13:08     Ewallet 522.83                4.761
905
## 2        5  3.8200  3/8/2019 10:29        Cash  76.40                4.761
905
## 3        7 16.2155  3/3/2019 13:23 Credit card 324.31                4.761
905
## 4        8 23.2880 1/27/2019 20:33     Ewallet 465.76                4.761
905
## 5        7 30.2085  2/8/2019 10:37     Ewallet 604.17                4.761
905
## 6        7 29.8865 3/25/2019 18:30     Ewallet 597.73                4.761
905
##   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
```

Previewing the bottom of the dataset:

```
tail(df)
```

```
##       Invoice.ID Branch Customer.type Gender           Product.line Unit.p
rice
## 995  652-49-6720      C        Member Female Electronic accessories      6
0.95
## 996  233-67-5758      C        Normal   Male      Health and beauty      4
0.35
## 997  303-96-2227      B        Normal Female     Home and lifestyle      9
7.38
## 998  727-02-1313      A        Member   Male     Food and beverages      3
```

```
1.84
## 999  347-56-2442      A        Normal   Male     Home and lifestyle      6
5.82
## 1000 849-09-3807      A        Member Female    Fashion accessories      8
8.34
##      Quantity     Tax      Date  Time Payment   cogs gross.margin.percenta
ge
## 995         1  3.0475 2/18/2019 11:40 Ewallet  60.95                4.7619
05
## 996         1  2.0175 1/29/2019 13:46 Ewallet  40.35                4.7619
05
## 997        10 48.6900  3/2/2019 17:16 Ewallet 973.80                4.7619
05
## 998         1  1.5920  2/9/2019 13:22    Cash  31.84                4.7619
05
## 999         1  3.2910 2/22/2019 15:33    Cash  65.82                4.7619
05
## 1000        7 30.9190 2/18/2019 13:28    Cash 618.38                4.7619
05
##      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
```

Checking datatype of each column:

```
str(df)

## 'data.frame':    1000 obs. of  16 variables:
##  $ Invoice.ID             : chr  "750-67-8428" "226-31-3081" "631-41-3108"
"123-19-1176" ...
##  $ Branch                 : chr  "A" "C" "A" "A" ...
##  $ Customer.type          : chr  "Member" "Normal" "Normal" "Member" ...
##  $ Gender                 : chr  "Female" "Female" "Male" "Male" ...
##  $ Product.line           : chr  "Health and beauty" "Electronic accessori
es" "Home and lifestyle" "Health and beauty" ...
##  $ 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" "1/27/20
19" ...
##  $ 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 ...
```

##Tidying the Dataset

```r
#checking 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
#converting column names to lowercase
colnames(df) = tolower(colnames(df))
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
#checking for missing values
data.frame(colSums(is.na(df)))
```

```
##                         colSums.is.na.df..
## invoice.id                               0
## branch                                   0
## customer.type                            0
## gender                                   0
## product.line                             0
## unit.price                               0
## quantity                                 0
## tax                                      0
## date                                     0
## time                                     0
## payment                                  0
## cogs                                     0
## gross.margin.percentage                  0
## gross.income                             0
## rating                                   0
## total                                    0
```

There were no missing values.

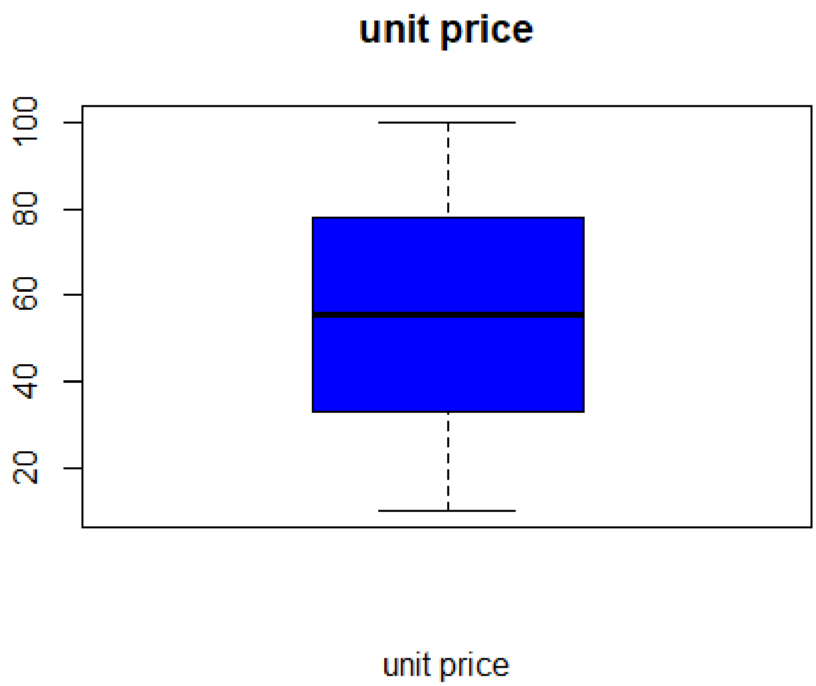
```
#checking for duplicates
nrow(df[duplicated(df),])

## [1] 0
```

There were no duplicates.

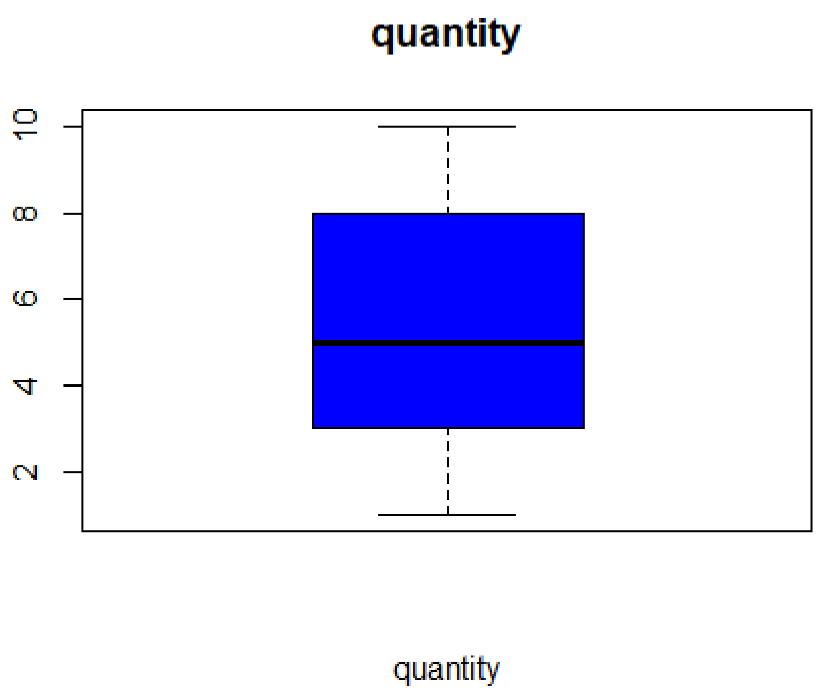
```
#date should be converted to datetime format
str(df$date)

##  chr [1:1000] "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" "2/8/2019" ...

#loading the lubridate library to work with dates
library(lubridate)

##
## Attaching package: 'lubridate'

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

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

# converting date to posixct
df$date <- as.POSIXct(df$date, format="%m/%d/%Y")
str(df$date)

##  POSIXct[1:1000], format: "2019-01-05" "2019-03-08" "2019-03-03" "2019-01-
27" "2019-02-08" ...

# break date down to month and year and day of week components
df$month = as.factor(month(df$date, label=TRUE))
df$day = as.factor(wday(df$date, label=TRUE, week_start=1))
df$year = year(df$date)
head(df)

##    invoice.id branch customer.type gender           product.line unit.pric
e
## 1 750-67-8428      A        Member Female      Health and beauty      74.6
9
## 2 226-31-3081      C        Normal Female Electronic accessories      15.2
8
## 3 631-41-3108      A        Normal   Male     Home and lifestyle      46.3
3
## 4 123-19-1176      A        Member   Male      Health and beauty      58.2
2
## 5 373-73-7910      A        Normal   Male      Sports and travel      86.3
```

```
1
## 6 699-14-3026      C        Normal   Male Electronic accessories      85.3
9
##   quantity     tax       date  time      payment   cogs gross.margin.percen
tage
## 1        7 26.1415 2019-01-05 13:08     Ewallet 522.83                4.76
1905
## 2        5  3.8200 2019-03-08 10:29        Cash  76.40                4.76
1905
## 3        7 16.2155 2019-03-03 13:23 Credit card 324.31                4.76
1905
## 4        8 23.2880 2019-01-27 20:33     Ewallet 465.76                4.76
1905
## 5        7 30.2085 2019-02-08 10:37     Ewallet 604.17                4.76
1905
## 6        7 29.8865 2019-03-25 18:30     Ewallet 597.73                4.76
1905
##   gross.income rating    total month day year
## 1      26.1415    9.1 548.9715   Jan Sat 2019
## 2       3.8200    9.6  80.2200   Mar Fri 2019
## 3      16.2155    7.4 340.5255   Mar Sun 2019
## 4      23.2880    8.4 489.0480   Jan Sun 2019
## 5      30.2085    5.3 634.3785   Feb Fri 2019
## 6      29.8865    4.1 627.6165   Mar Mon 2019
```

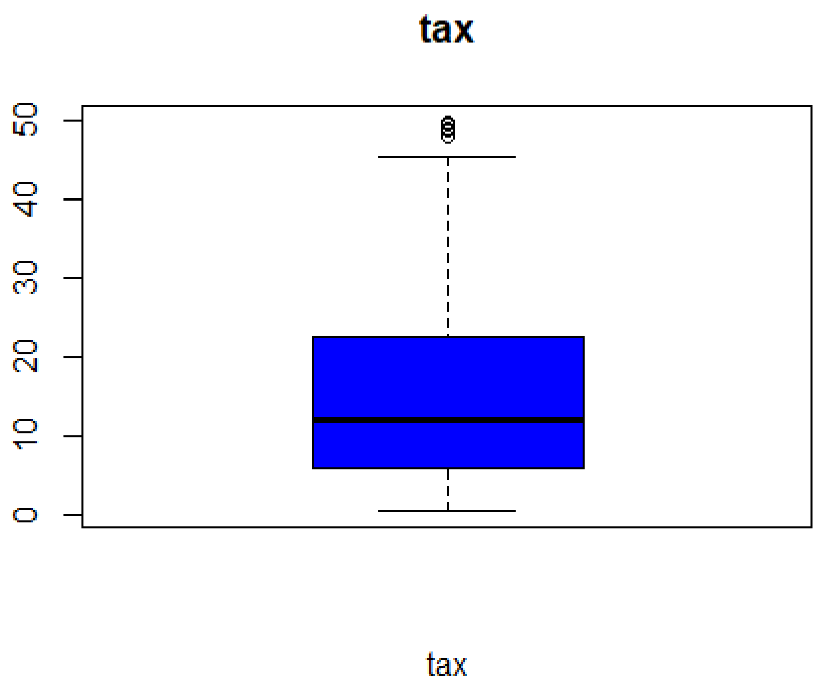
```
#separating continuous and categorical

contin = c("unit.price", "quantity", "tax", "cogs", "gross.margin.percentage"
, "gross.income", "rating", "total")
cat = c("invoice.id", "branch", "customer.type", "gender", "product.line","mo
nth", "day","year", "payment")

#checking for outliers in continuous columns

#function to replace period in column names with blankspace
repl <- function(x){
 gsub(".", " ", x,fixed=TRUE)

}
#checking for outliers in continuous columns
for (x in contin){
 boxplot(df[x], main=repl(x), xlab=repl(x), col="blue")
}
```

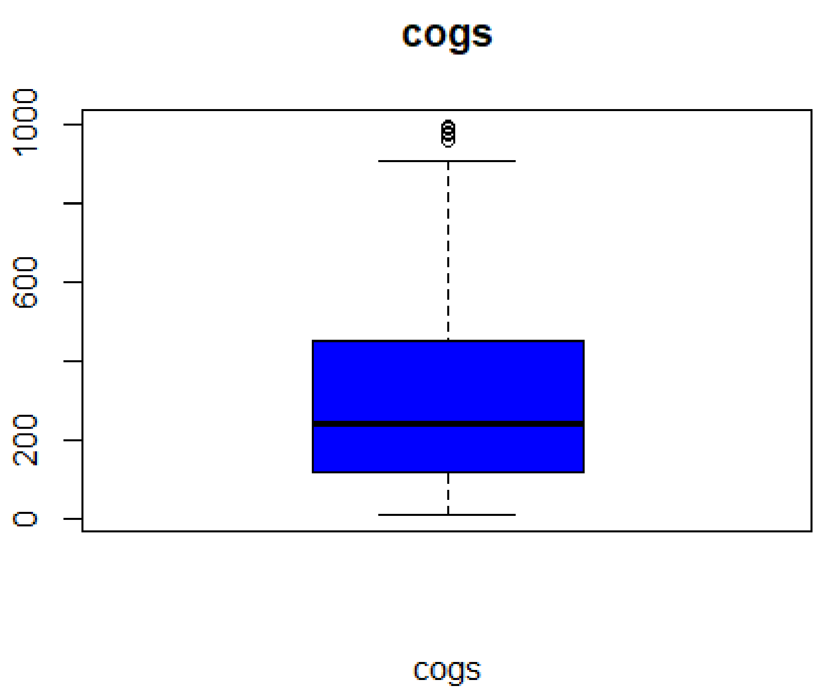


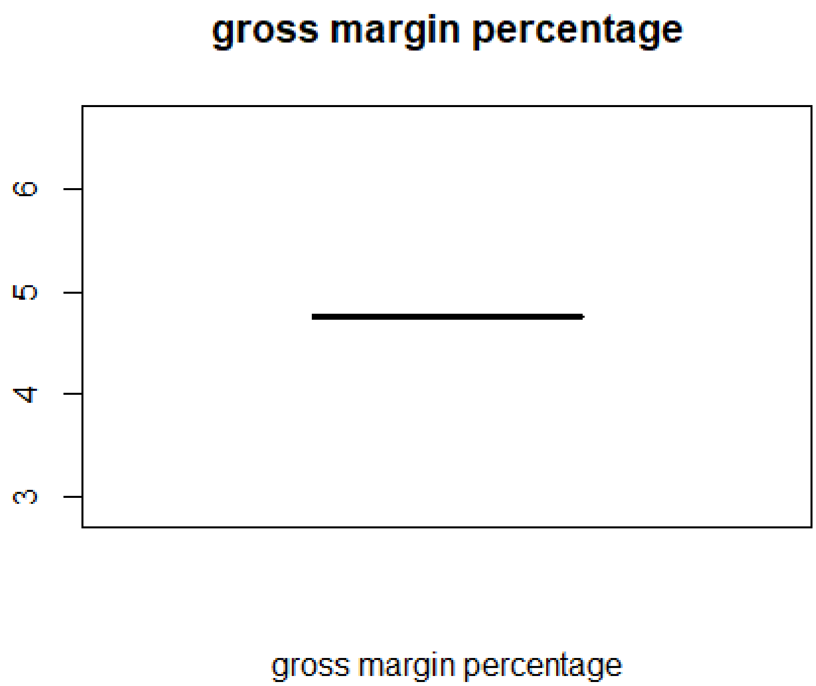
unit price
100
80
60
40
20
unit price

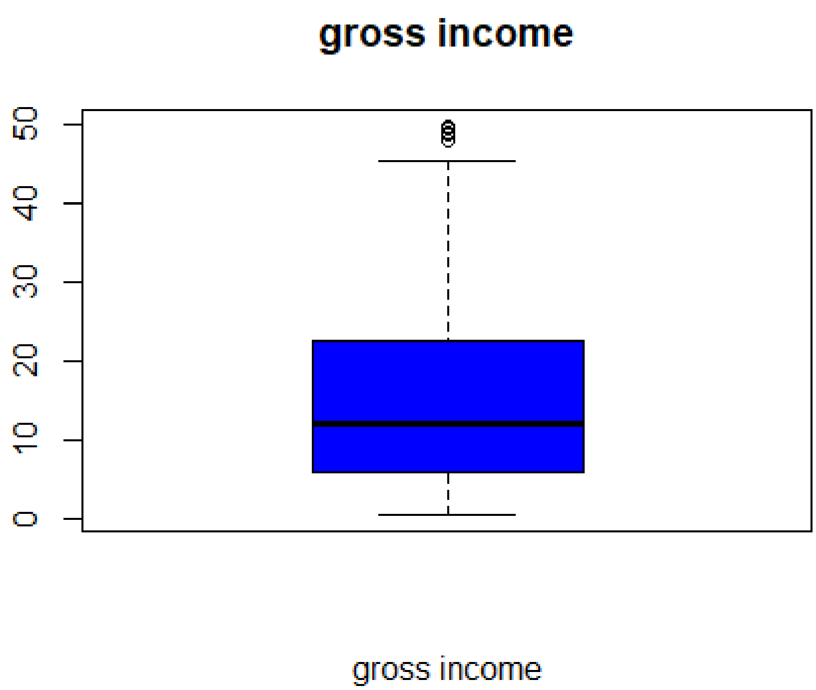


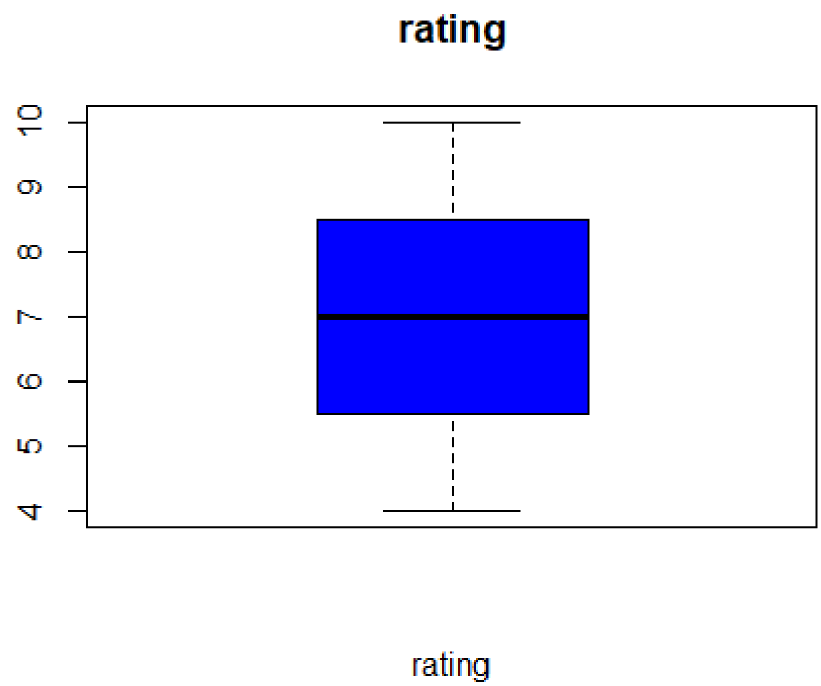
quantity
10
8
6
4
2
quantity

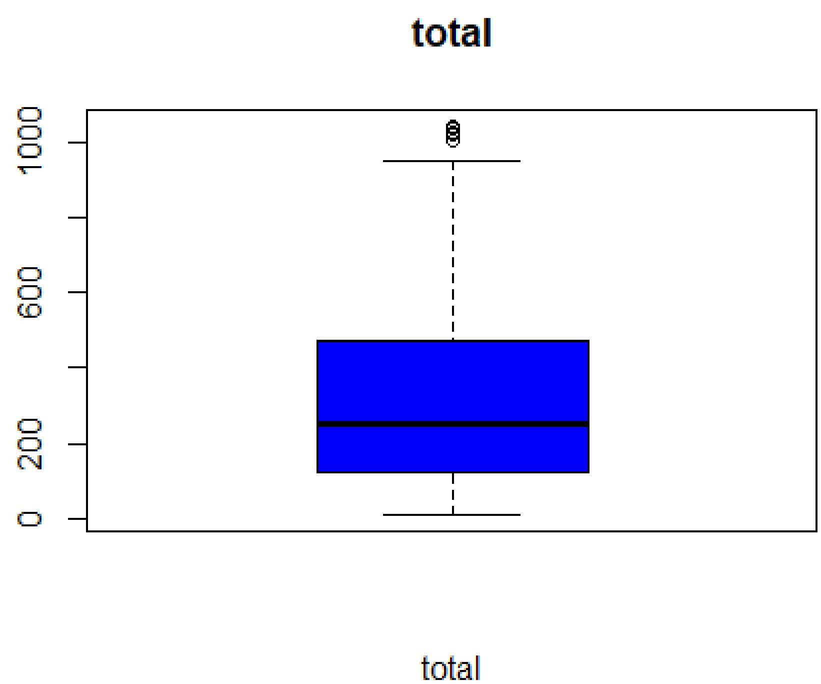

tax
50
40
30
20
10
0
tax


cogs
1000
600
200
0
cogs


gross margin percentage
3
4
5
6
gross margin percentage


gross income
0
10
20
30
40
50
gross income





There were a few outliers in the tax, cogs(cost of goods), gross income, and total columns. They will not be dropped as it is possible for some customers to carry out shopping that costs much more than average.

```r
#checking for anomalies in continuous
#the values should not be less than zero.

for (x in contin){
  print(paste(x, nrow(subset(df, df[x] < 0))))
}
```

```
## [1] "unit.price 0"
## [1] "quantity 0"
## [1] "tax 0"
## [1] "cogs 0"
## [1] "gross.margin.percentage 0"
## [1] "gross.income 0"
## [1] "rating 0"
## [1] "total 0"
```

```r
#none of the values were less than zero

#checking for number of unique values in categorical columns
for (x in cat){
  print(paste(x, length(unique(df[[x]]))))
}
```

```
## [1] "invoice.id 1000"
## [1] "branch 3"
## [1] "customer.type 2"
## [1] "gender 2"
## [1] "product.line 6"
## [1] "month 3"
## [1] "day 7"
## [1] "year 1"
## [1] "payment 3"
```

```r
# dropping id column because it is unique for each row, similar to the index
df <- subset(df, select=-invoice.id)
colnames(df)
```

```
##  [1] "branch"                  "customer.type"
##  [3] "gender"                  "product.line"
##  [5] "unit.price"              "quantity"
##  [7] "tax"                     "date"
##  [9] "time"                    "payment"
## [11] "cogs"                    "gross.margin.percentage"
## [13] "gross.income"            "rating"
## [15] "total"                   "month"
## [17] "day"                     "year"
```

```r
#checking for anomalies in categorical

for (x in cat[2:9]){
  print(x)
```

```
  print(unique(df[[x]]))

  print("***************************************")
}
```

```
## [1] "branch"
## [1] "A" "C" "B"
## [1] "***************************************"
## [1] "customer.type"
## [1] "Member" "Normal"
## [1] "***************************************"
## [1] "gender"
## [1] "Female" "Male"
## [1] "***************************************"
## [1] "product.line"
## [1] "Health and beauty"      "Electronic accessories" "Home and lifestyle"
## [4] "Sports and travel"      "Food and beverages"     "Fashion accessories
"
## [1] "***************************************"
## [1] "month"
## [1] Jan Mar Feb
## 12 Levels: Jan < Feb < Mar < Apr < May < Jun < Jul < Aug < Sep < ... < Dec
## [1] "***************************************"
## [1] "day"
## [1] Sat Fri Sun Mon Thu Wed Tue
## Levels: Mon < Tue < Wed < Thu < Fri < Sat < Sun
## [1] "***************************************"
## [1] "year"
## [1] 2019
## [1] "***************************************"
## [1] "payment"
## [1] "Ewallet"     "Cash"        "Credit card"
## [1] "***************************************"
```

No anomalous values observed. The year is 2019 only so will drop year column

```
df <- subset(df, select=-year)
```

##Univariate Analysis

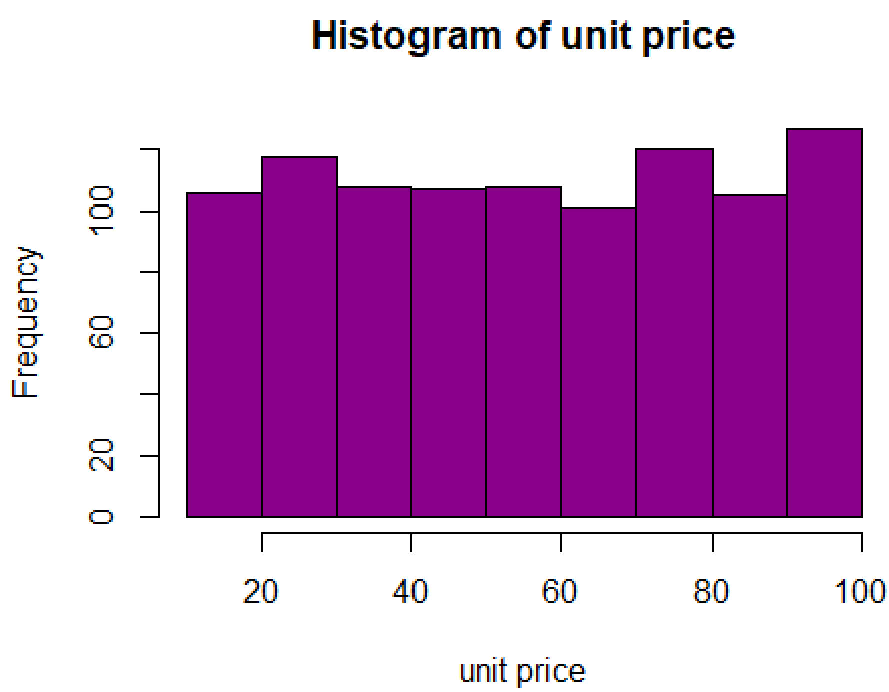
```
#loading ggplot 2 library for visualisation
library(ggplot2)

contin
```

```
## [1] "unit.price"              "quantity"
## [3] "tax"                     "cogs"
## [5] "gross.margin.percentage" "gross.income"
## [7] "rating"                  "total"
```

```r
#statistical summary of unit price variable
data.frame(describe(df$unit.price))
```

```
##    vars    n     mean       sd median trimmed      mad   min   max range
## X1    1 1000 55.67213 26.49463  55.23 55.6178 33.36591 10.08 99.96 89.88
##          skew  kurtosis        se
## X1 0.00705623 -1.222062 0.8378337
```

```r
#plotting unit.price histogram
hist(df$unit.price, col="darkmagenta",
     main="Histogram of unit price",
     xlab="unit price")
```

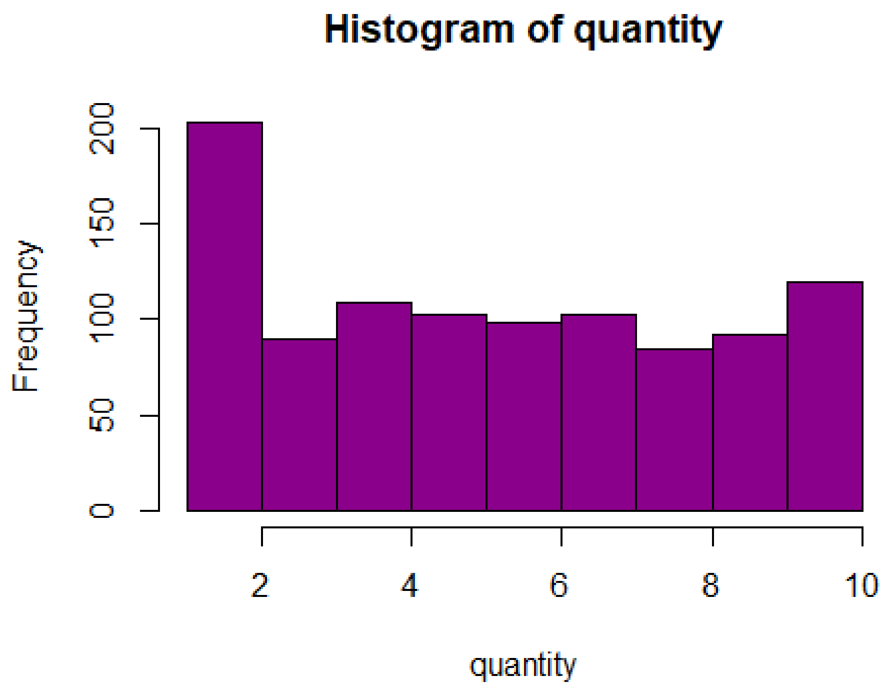


In most invoices the unit price was between 90 and 100

```r
#statistical sumary of quantity
describe(df$quantity)
```

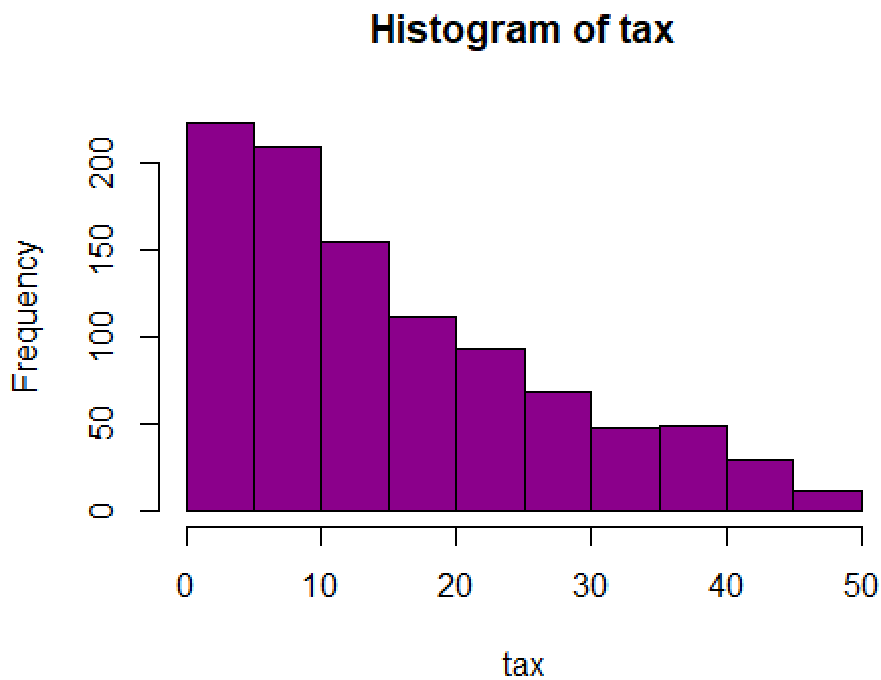
```
##    vars    n mean   sd median trimmed  mad min max range skew kurtosis   s
e
## X1    1 1000 5.51 2.92      5    5.51 2.97   1  10     9 0.01    -1.22 0.0
9
```

```r
#histogram of quantity
hist(df$quantity, col="darkmagenta",
     main="Histogram of quantity",
     xlab="quantity")
```



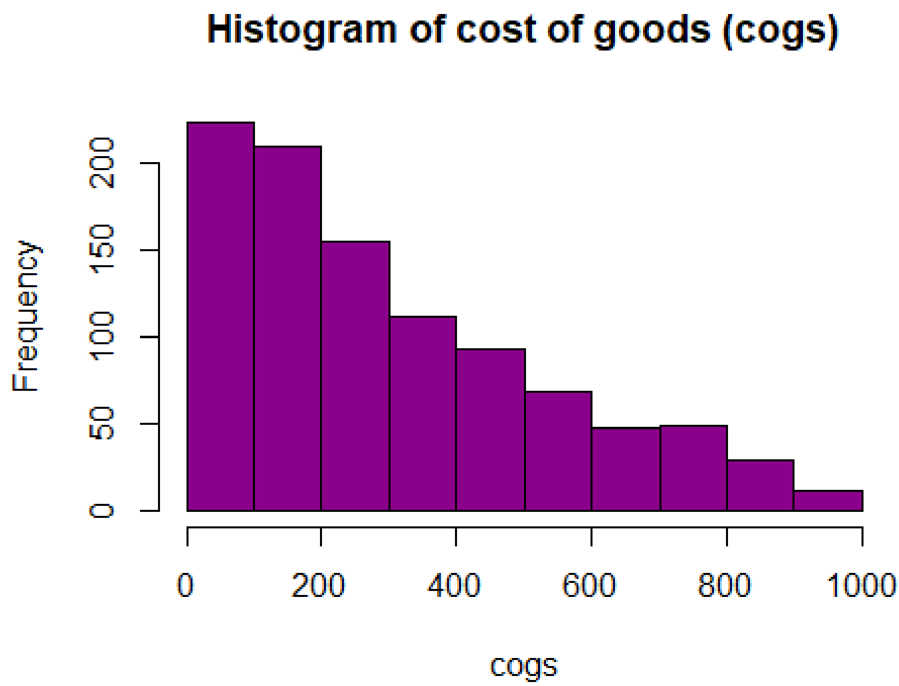
In most invoices the quantity of units was ranging from 1 to 2

```
#statistical sumary of tax
describe(df$tax)

##    vars    n  mean    sd median trimmed   mad  min   max range skew kurtos
is
## X1    1 1000 15.38 11.71  12.09      14 11.13 0.51 49.65 49.14 0.89    -0.
09
##      se
## X1 0.37

#histogram of tax
hist(df$tax, col="darkmagenta",
     main="Histogram of tax",
     xlab="tax")
```



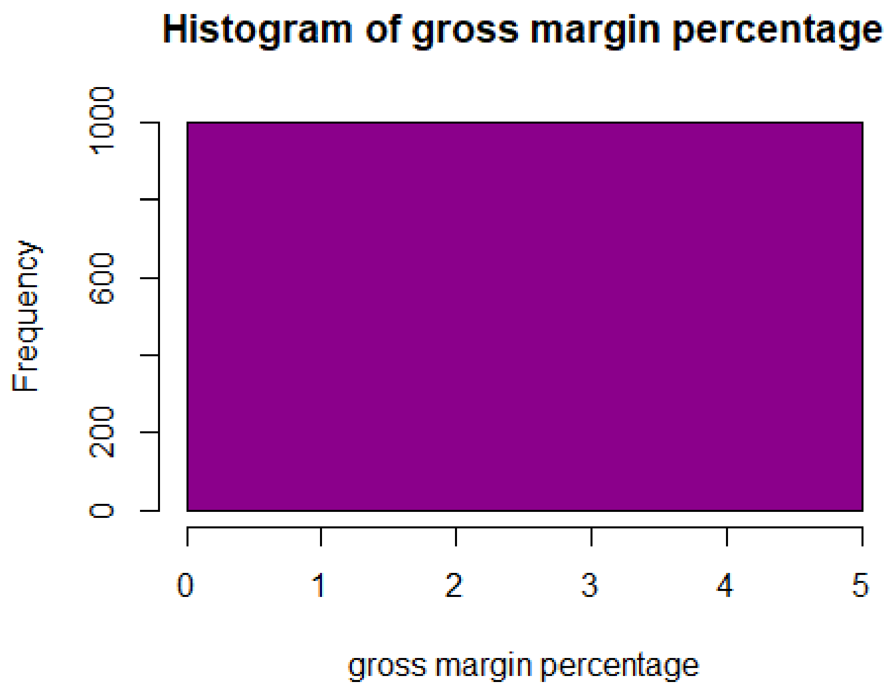
For most invoices, the value of tax ranged from 0 to 5.

```
#statistical summary of cogs variable
describe(df$cogs)

##    vars    n   mean     sd median trimmed    mad   min max  range skew kur
tosis
## X1    1 1000 307.59 234.18 241.76  279.91 222.65 10.17 993 982.83 0.89
-0.09
##      se
## X1 7.41

#histogram of informational_duration
hist(df$cogs, col="darkmagenta",
     main="Histogram of cost of goods (cogs)",
     xlab="cogs")
```



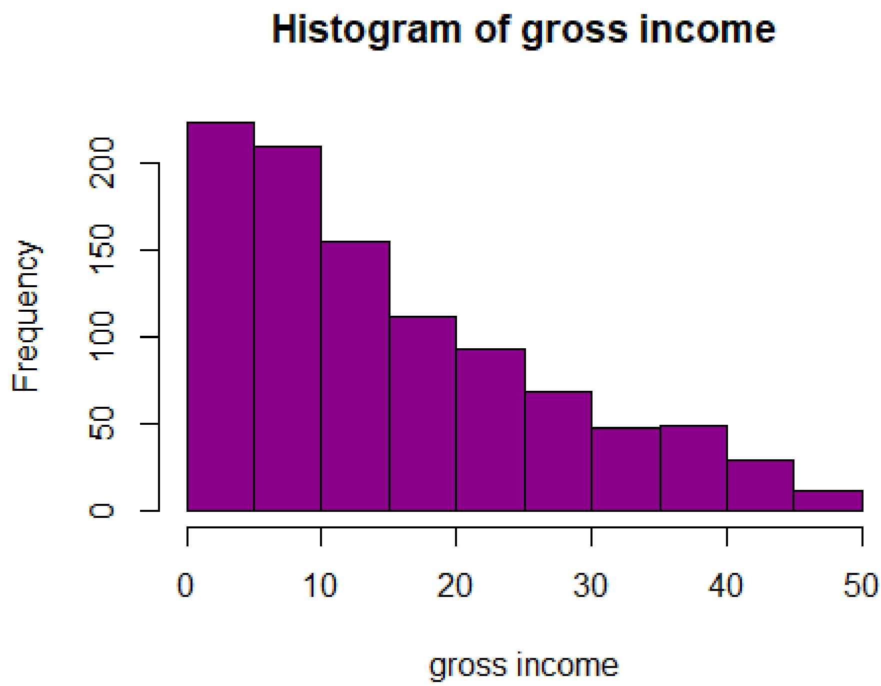
The cost of goods (without tax) in most invoices ranged from 0 to 200

```
#statistical sumary of gross margin percentage variable
describe(df$gross.margin.percentage)

##    vars    n mean sd median trimmed mad  min  max range skew kurtosis se
## X1    1 1000 4.76  0   4.76    4.76   0 4.76 4.76     0  NaN      NaN  0

#histogram of gross margin percentage
hist(df$gross.margin.percentage, col="darkmagenta",
     main="Histogram of gross margin percentage",
     xlab="gross margin percentage")
```



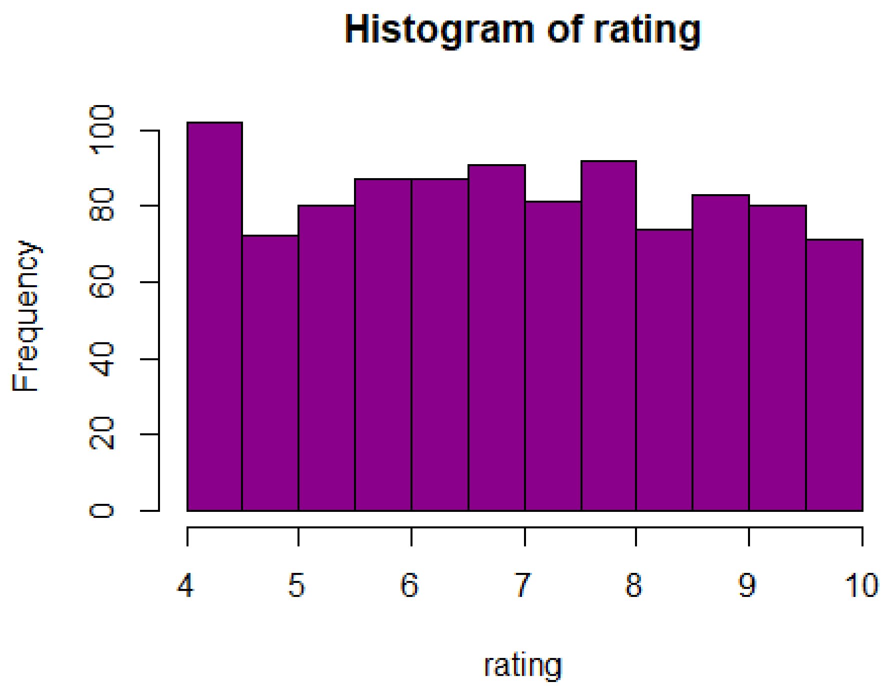
All values of gross margin percentage fell between 0 and 5

```
#statistical sumary of gross income variable
describe(df$gross.income)

##    vars    n  mean    sd median trimmed   mad  min   max range skew kurtos
is
## X1    1 1000 15.38 11.71  12.09      14 11.13 0.51 49.65 49.14 0.89    -0.
09
##      se
## X1 0.37

#histogram of gross income
hist(df$gross.income, col="darkmagenta",
     main="Histogram of gross income",
     xlab="gross income")
```



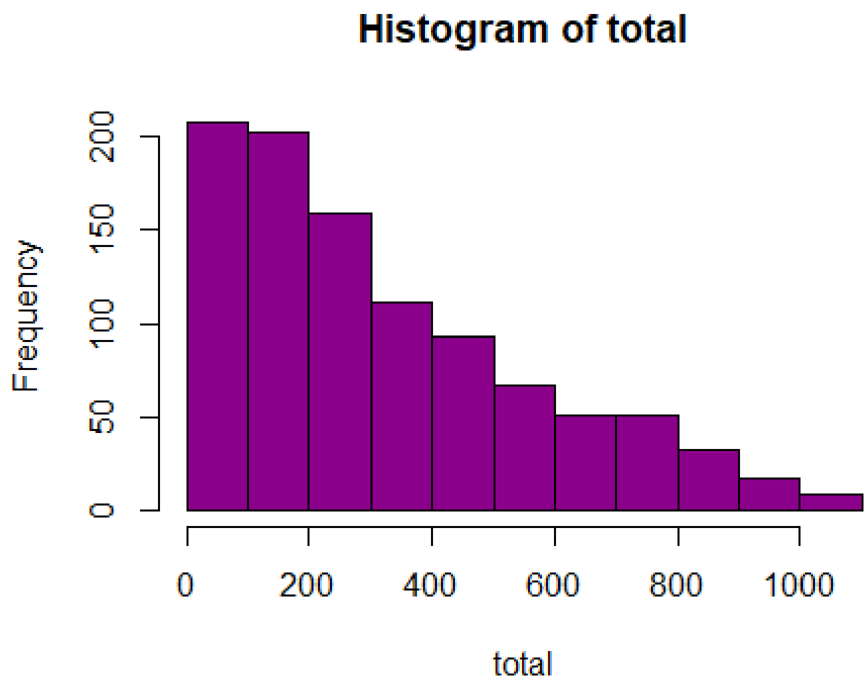
Gross income from most sales ranged between 0 and 10

```
#statistical sumary of rating variable
describe(df$rating)

##    vars    n mean   sd median trimmed  mad min max range skew kurtosis   s
e
## X1    1 1000 6.97 1.72      7    6.97 2.22   4  10     6 0.01    -1.16 0.0
5

#histogram of rating
hist(df$rating, col="darkmagenta",
     main="Histogram of rating",
     xlab="rating")
```



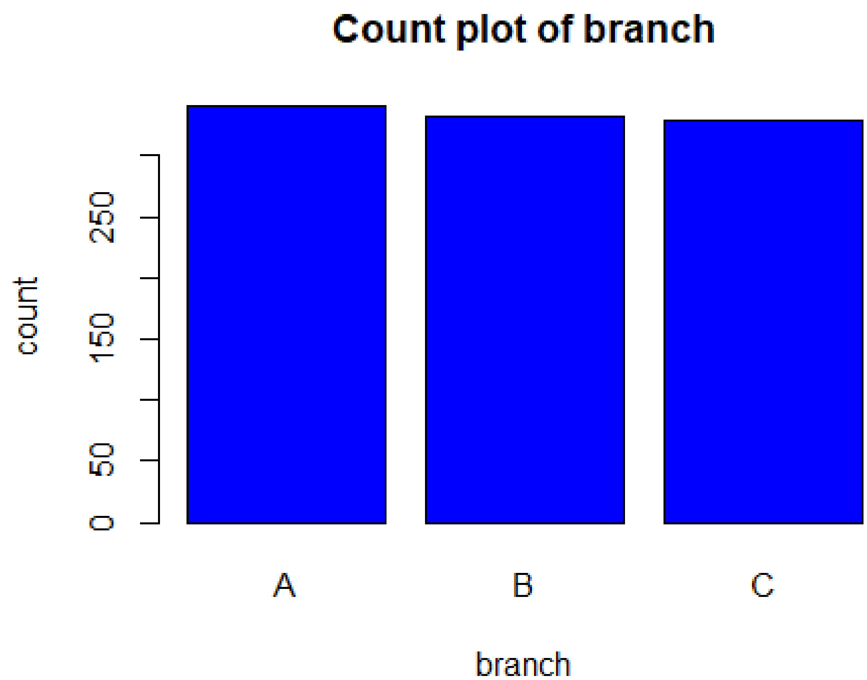
Most ratings were between 4 and 4.5

```
#statistical sumary of total variable
describe(df$total)

##    vars    n   mean     sd median trimmed    mad   min     max   range ske
w
## X1    1 1000 322.97 245.89 253.85  293.91 233.78 10.68 1042.65 1031.97 0.8
9
##    kurtosis   se
## X1    -0.09 7.78

#histogram of total
hist(df$total, col="darkmagenta",
     main="Histogram of total",
     xlab="total")
```



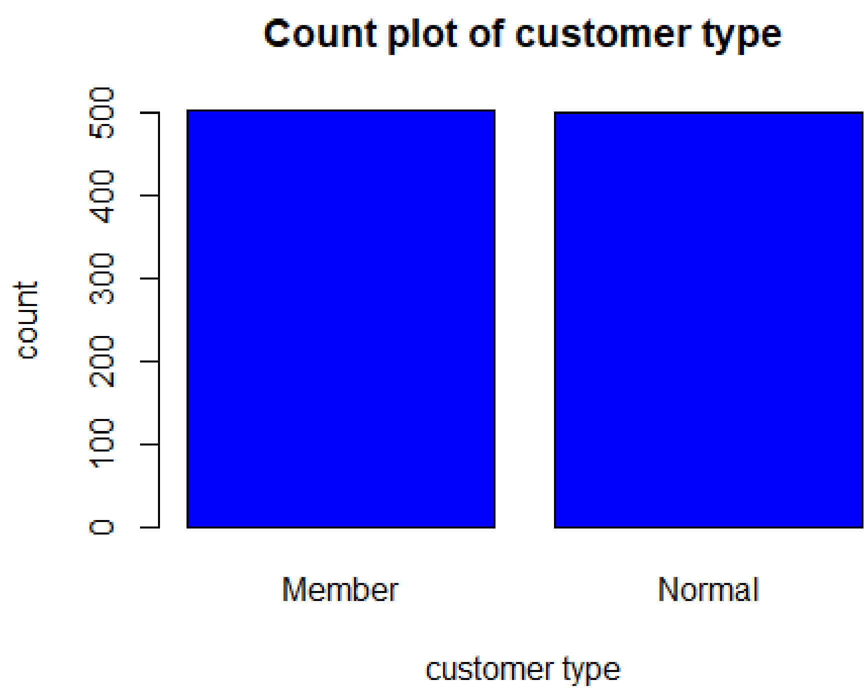
Most total sale values ranged between 0 and 200

```
cat

## [1] "invoice.id"    "branch"         "customer.type" "gender"
## [5] "product.line"  "month"          "day"           "year"
## [9] "payment"

#Count plot of branch
barplot(table(df$branch), col="blue", main="Count plot of branch",
        xlab = "branch", ylab="count")
```



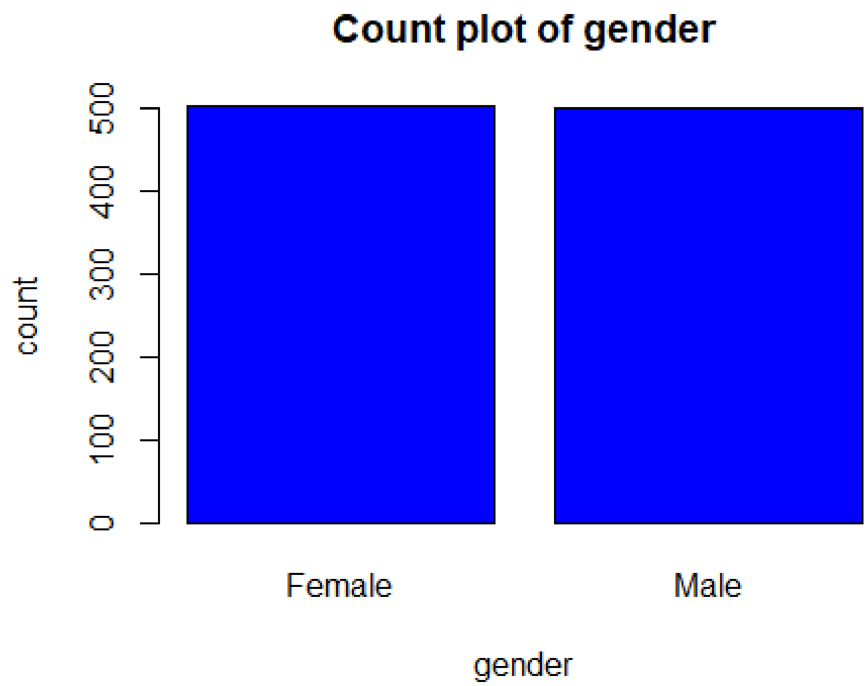
There is minimal difference in the number of invoices from the different branches

```
#count plot of customer type

barplot(table(df$customer.type), col="blue", main="Count plot of customer typ
e",
        xlab = "customer type", ylab="count")
```



The number of records from member and normal customers are almost equal

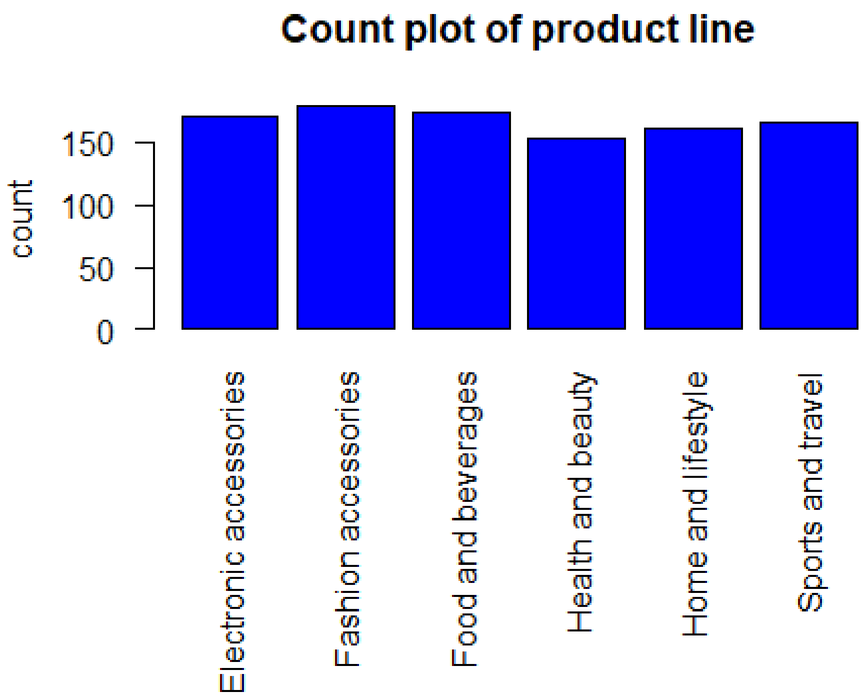
```
#count plot of  gender
barplot(table(df$gender), col="blue", main="Count plot of gender",
        xlab = "gender", ylab="count")
```




```
table(df$gender)

##
## Female   Male
##    501    499
```

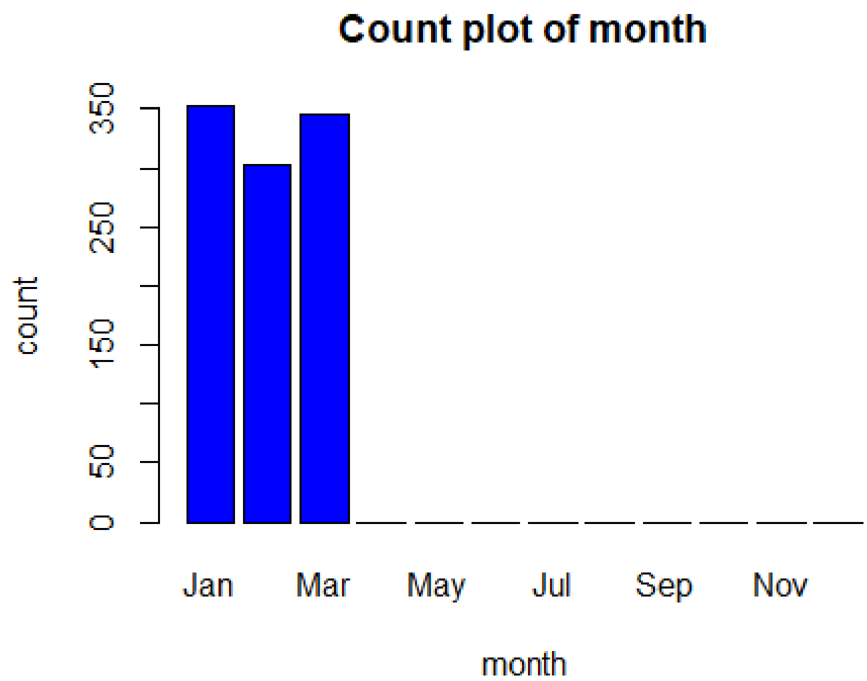
There is minimal difference in the counts of male and female customers in the dataset.

```
#count plot of product line
par(mar= c(10.1,4.1,4.1,2.1))
par(las=2)
barplot(table(df$product.line), col="blue", main="Count plot of product line"
, ylab="count")
```



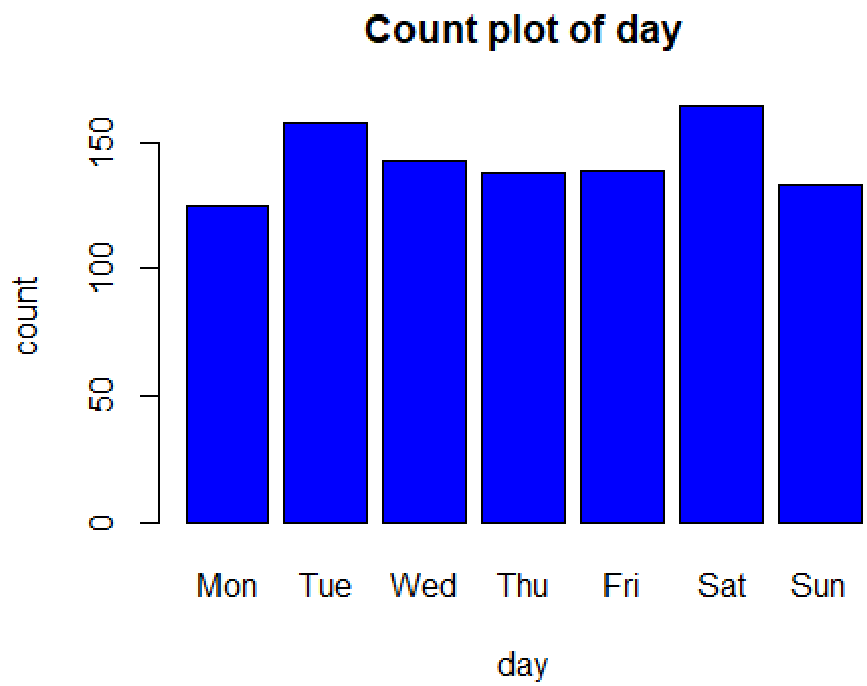
Fashion accessories were the product line with most sale records

```
#count plot of month
barplot(table(df$month), col="blue",
        main="Count plot of month",
        xlab = "month", ylab="count")
```



Between January and March 2019, January was the month with the most invoices (sales)

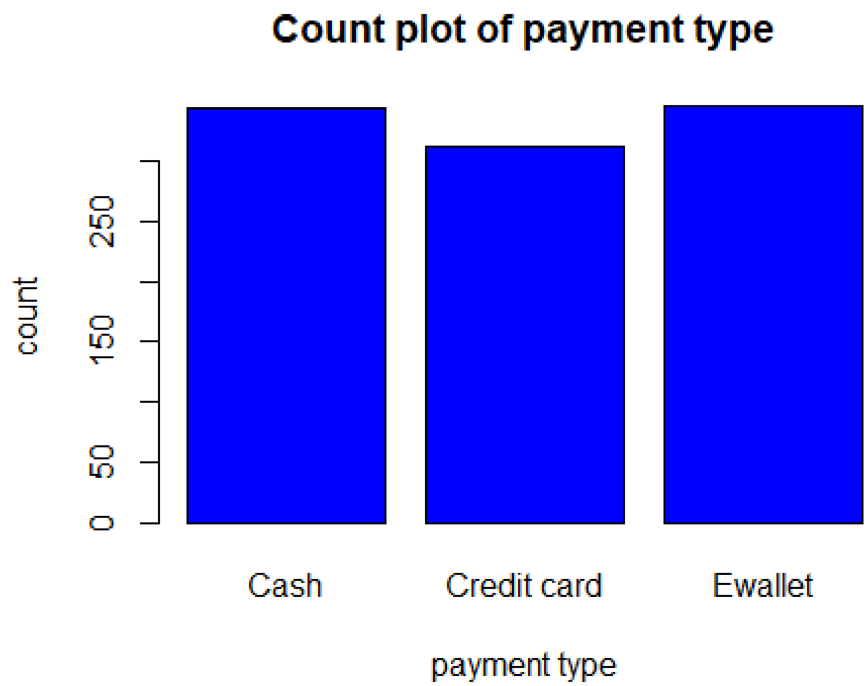
```
#count plot of day
barplot(table(df$day), col="blue",
        main="Count plot of day",
        xlab = "day", ylab="count")
```




```
table(df$day)

##
## Mon Tue Wed Thu Fri Sat Sun
## 125 158 143 138 139 164 133
```

Saturday was the day with the most invoices (sales), closely followed by Tuesday.

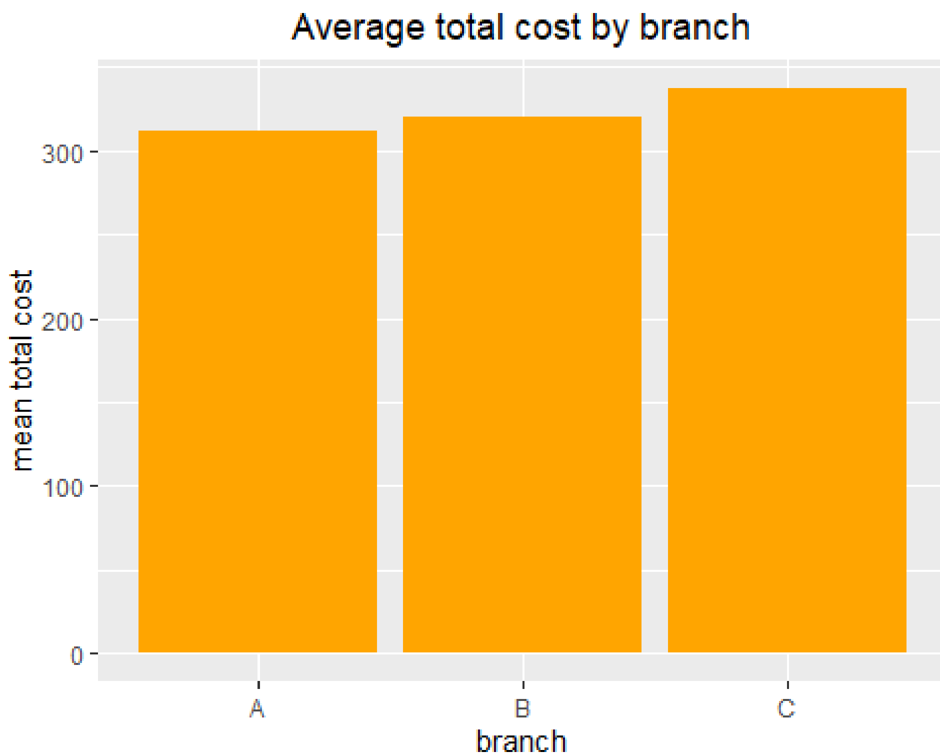
```
#count plot of payment
barplot(table(df$payment), col="blue",
        main="Count plot of payment type",
        xlab = "payment type", ylab="count")
```



Most payments were through cash and ewallet

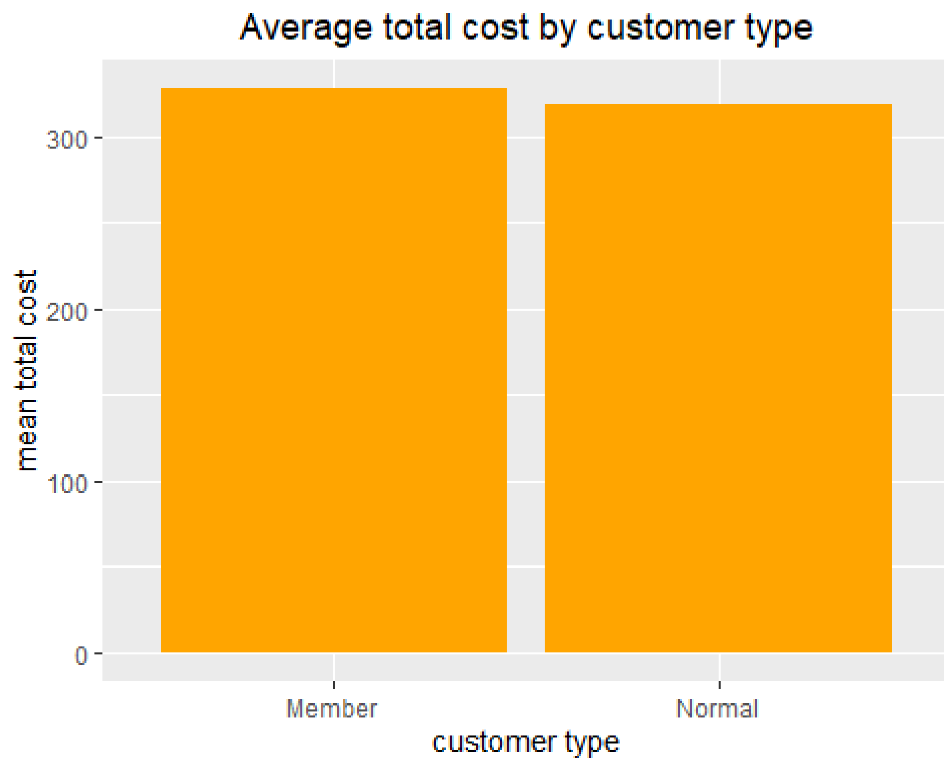
Bivariate Analysis

```
#loading library to use functions
library("dplyr")

group_by

## function (.data, ..., .add = FALSE, .drop = group_by_drop_default(.data))
## {
##     UseMethod("group_by")
## }
## <bytecode: 0x000001fb954428c8>
## <environment: namespace:dplyr>

#plotting average total by branch
m = df %>% dplyr::group_by(branch) %>%
 dplyr::summarise(mean=mean(total))
ggplot() + geom_col(
 data=m,
 aes(x=as.factor(branch), y=mean),
 fill="orange") + labs(title = "Average total cost by branch",
 y="mean total cost", x="branch") + theme(plot.title =
element_text(hjust=0.5))
```



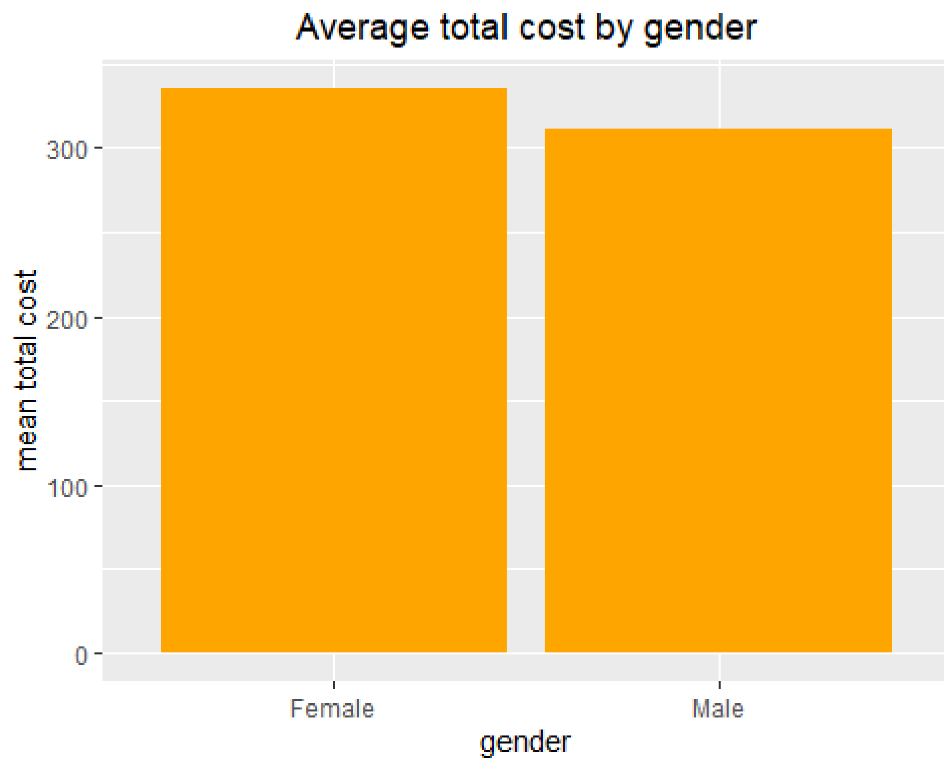
The mean sales in branch C were higher than in the other branches

```
#plotting average total by customer type
m = df %>% dplyr::group_by(customer.type) %>%
 dplyr::summarise(mean=mean(total))
ggplot() + geom_col(
 data=m,
 aes(x=as.factor(customer.type), y=mean),
 fill="orange") + labs(title = "Average total cost by customer type",
 y="mean total cost", x="customer type") + theme(plot.title =
element_text(hjust=0.5))
```



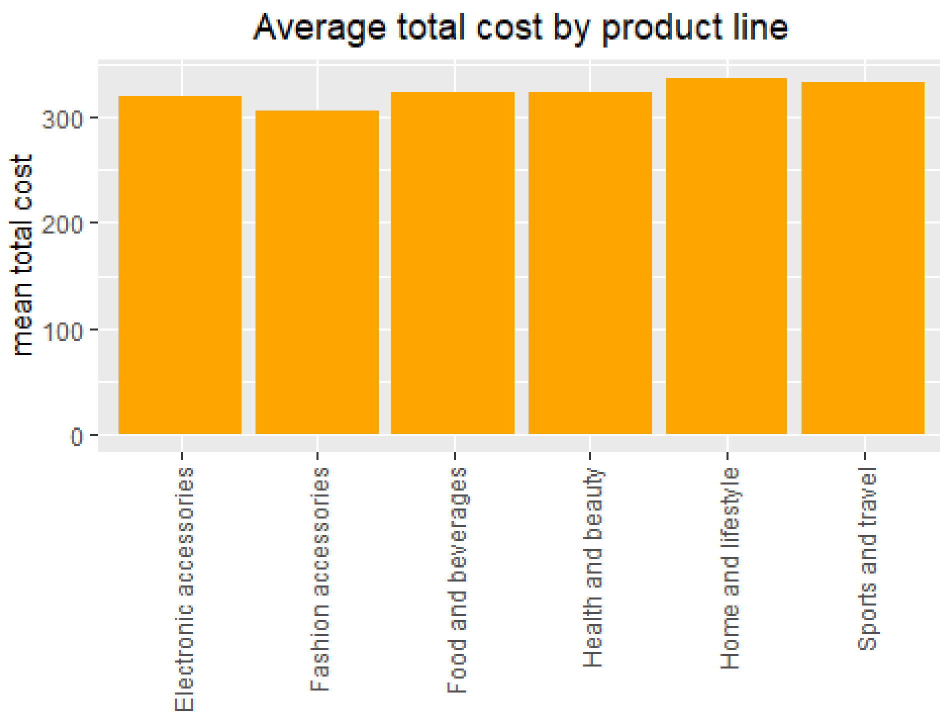
The average sales from members were higher than from normal customers

```
#plotting average total by gender
m = df %>% dplyr::group_by(gender) %>%
 dplyr::summarise(mean=mean(total))
ggplot() + geom_col(
 data=m,
 aes(x=as.factor(gender), y=mean),
 fill="orange") + labs(title = "Average total cost by gender",
 y="mean total cost", x="gender") + theme(plot.title =
element_text(hjust=0.5))
```



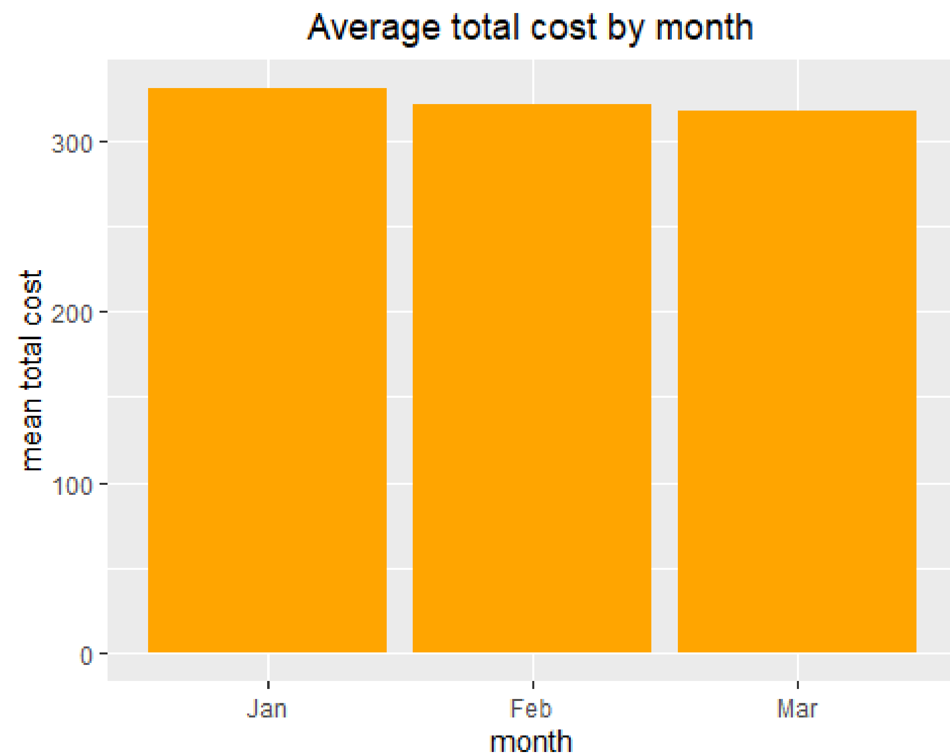
The mean sales were higher among female than male customers

```
#plotting average total by product line
m = df %>% dplyr::group_by(product.line) %>%
 dplyr::summarise(mean=mean(total))
ggplot() + geom_col(
 data=m,
 aes(x=as.factor(product.line), y=mean),
 fill="orange") + labs(title = "Average total cost by product line",
 y="mean total cost", x="product line") + theme(plot.title =
element_text(hjust=0.5), axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)
)
```



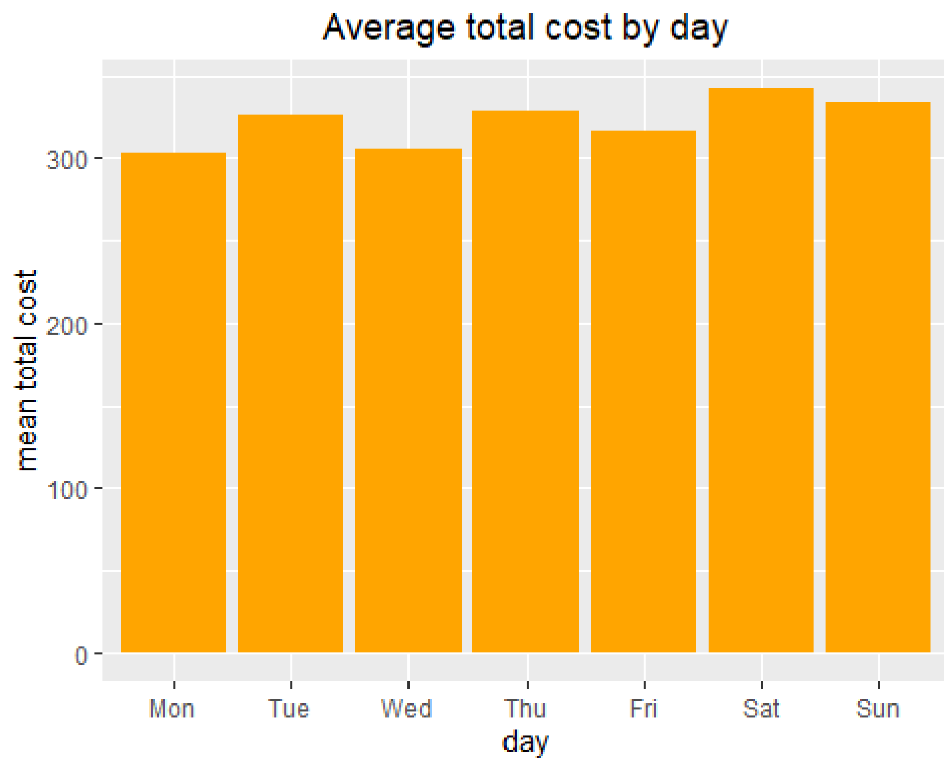
The product line with the highest average sales was home and lifestyle

```
#plotting average total by month
m = df %>% dplyr::group_by(month) %>%
 dplyr::summarise(mean=mean(total))
ggplot() + geom_col(
 data=m,
 aes(x=as.factor(month), y=mean),
 fill="orange") + labs(title = "Average total cost by month",
 y="mean total cost", x="month") + theme(plot.title =
element_text(hjust=0.5))
```



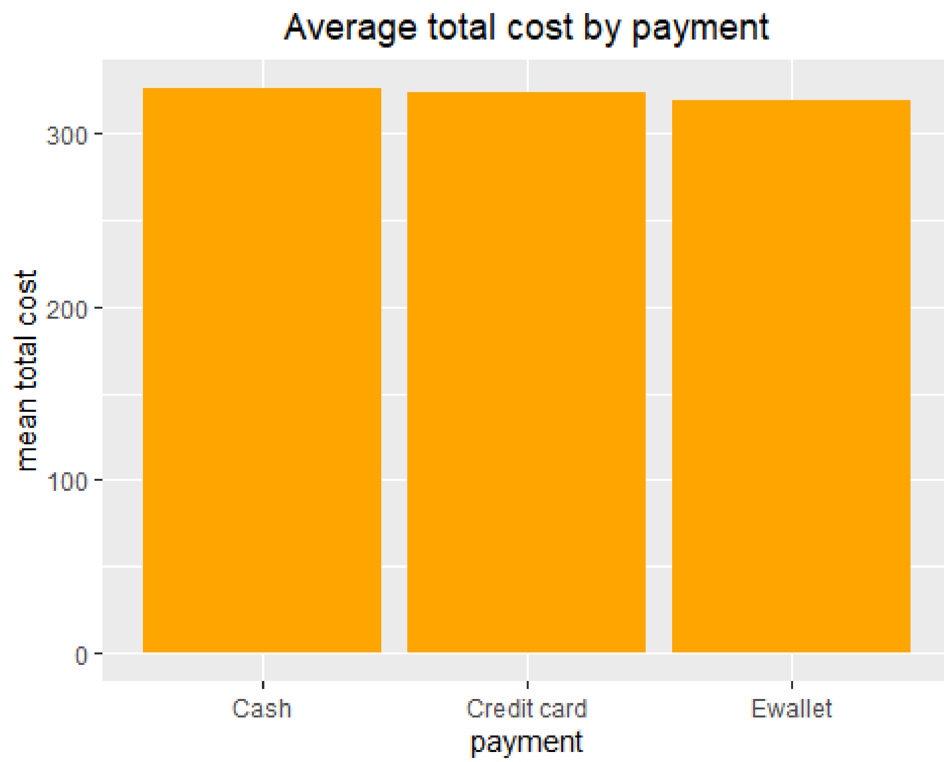
January had the highest average sales

```
#plotting average total by day
m = df %>% dplyr::group_by(day) %>%
 dplyr::summarise(mean=mean(total))
ggplot() + geom_col(
 data=m,
 aes(x=as.factor(day), y=mean),
 fill="orange") + labs(title = "Average total cost by day",
 y="mean total cost", x="day") + theme(plot.title =
element_text(hjust=0.5))
```



Saturday had the highest average sales

```
#plotting average total by payment
m = df %>% dplyr::group_by(payment) %>%
 dplyr::summarise(mean=mean(total))
ggplot() + geom_col(
 data=m,
 aes(x=as.factor(payment), y=mean),
 fill="orange") + labs(title = "Average total cost by payment",
 y="mean total cost", x="payment") + theme(plot.title =
element_text(hjust=0.5))
```



The mean sales were almost equal across the different payment methods

Scatterplots of continuous columns

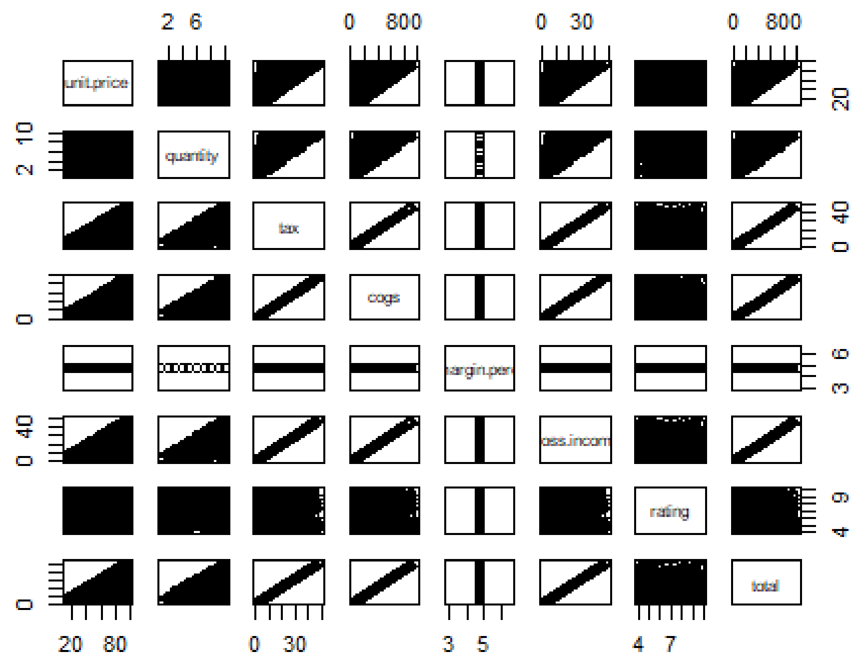
```
#continuous columns
contin

## [1] "unit.price"              "quantity"
## [3] "tax"                     "cogs"
## [5] "gross.margin.percentage" "gross.income"
## [7] "rating"                  "total"

#creating dataframe that containing the continuous variables
scatterp = subset(df, select = c("unit.price","quantity","tax"
,"cogs" ,"gross.margin.percentage", "gross.income" , "rating","total"))
head(scatterp)

##   unit.price quantity     tax   cogs gross.margin.percentage gross.income
## 1      74.69        7 26.1415 522.83                4.761905      26.1415
## 2      15.28        5  3.8200  76.40                4.761905       3.8200
## 3      46.33        7 16.2155 324.31                4.761905      16.2155
## 4      58.22        8 23.2880 465.76                4.761905      23.2880
## 5      86.31        7 30.2085 604.17                4.761905      30.2085
## 6      85.39        7 29.8865 597.73                4.761905      29.8865
##   rating    total
## 1    9.1 548.9715
## 2    9.6  80.2200
## 3    7.4 340.5255
```

```
## 4     8.4 489.0480
## 5     5.3 634.3785
## 6     4.1 627.6165
```

```
#loading library for pair plot
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
#plotting scatterplots of continuous variables
plot(scatterp)
```



Total sales has a positive correlation with gross income, cogs(cost of goods), tax, quantity and unit price

Correlation matrix

```
str(df)
```

```
## 'data.frame':    1000 obs. of  17 variables:
##  $ 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 accessori
es" "Home and lifestyle" "Health and beauty" ...
##  $ 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                    : POSIXct, format: "2019-01-05" "2019-03-08" ...
##  $ 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 ...
##  $ month                   : Ord.factor w/ 12 levels "Jan"<"Feb"<"Mar"<..:
1 3 3 1 2 3 2 2 1 2 ...
##  $ day                     : Ord.factor w/ 7 levels "Mon"<"Tue"<"Wed"<..: 6
5 7 7 5 1 1 7 4 3 ...
```

```
#apply function to time column to extract the hour of day
df$hour <- hour(as.POSIXct(df$time, format="%H:%M"))
str(df$time)
```

```
##  chr [1:1000] "13:08" "10:29" "13:23" "20:33" "10:37" "18:30" "14:36" ...
```

```
str(df$hour)
```

```
##  int [1:1000] 13 10 13 20 10 18 14 11 17 13 ...
```

```
unique(df$gross.margin.percentage)
```

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

```
#no variation in that column
```

```
#converting categorical to numerical
#removing date column(was broken down to components), time column(leaving the
hour aspect)
#removing gross margin percent column as there is no variation - same value
#dataframe for correlation matrix
enc_df <- subset(df, select=-c(date, time, gross.margin.percentage))

enc_df$month <- as.numeric(factor(enc_df$month))
enc_df$branch <- as.numeric(factor(enc_df$branch))
enc_df$customer.type <- as.numeric(factor(enc_df$customer.type))
enc_df$product.line <- as.numeric(factor(enc_df$product.line))
enc_df$gender <- as.numeric(factor(enc_df$gender))
enc_df$day <- as.numeric(factor(enc_df$day))
enc_df$gender <- as.numeric(factor(enc_df$gender))
enc_df$payment <- as.numeric(factor(enc_df$payment))
#checking that datatype conversion worked
str(enc_df)
```

```
## 'data.frame':    1000 obs. of  15 variables:
##  $ branch       : num  1 3 1 1 1 3 1 3 1 2 ...
##  $ customer.type: num  1 2 2 1 2 2 1 2 1 1 ...
##  $ gender       : num  1 1 2 2 2 2 1 1 1 1 ...
```

```
##  $ product.line : num  4 1 5 4 6 1 1 5 4 3 ...
##  $ unit.price   : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ quantity     : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ tax          : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ payment      : num  3 1 2 3 3 3 3 3 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 ...
##  $ month        : num  1 3 3 1 2 3 2 2 1 2 ...
##  $ day          : num  6 5 7 7 5 1 1 7 4 3 ...
##  $ hour         : int  13 10 13 20 10 18 14 11 17 13 ...
```

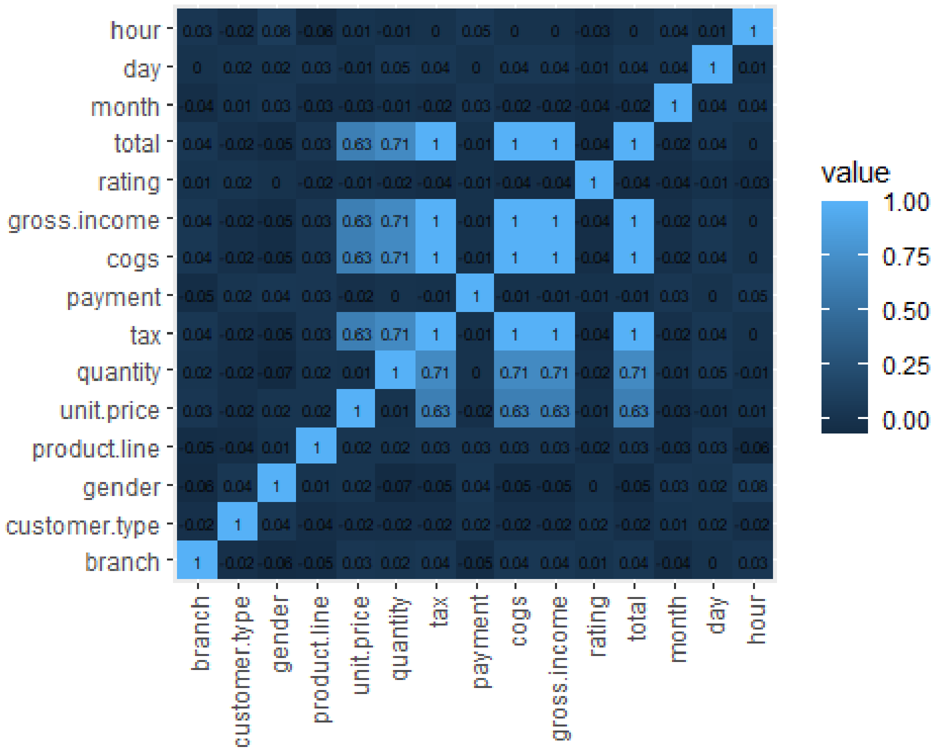
```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

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

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```
#plotting the correlation heatmap
datam = melt(round(cor(enc_df),2))
ggplot(data=datam, aes(x=Var1, y=Var2, fill=value)) + geom_tile() + geom_text
(aes(Var2, Var1, label=value), color="black",size=2) + theme(axis.text.x=elem
ent_text(angle=90,vjust=0.5,hjust=1), axis.title.x = element_blank(), axis.ti
tle.y = element_blank())
```



Total sales column has a perfect positive correlation with gross income, cogs (cost of goods), and tax. It also has a strong positive correlation with quantity and unit price.

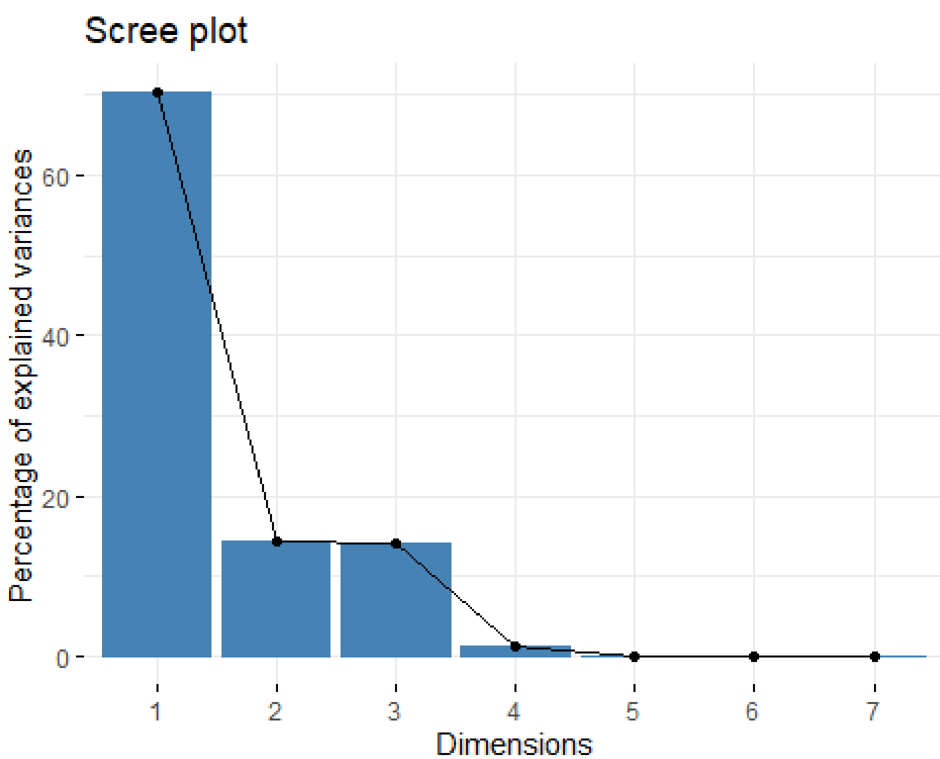
##Part 1: Dimensionality Reduction

###PCA

```
# Selecting the numerical data (excluding the categorical variables)
df_num <- subset(df, select=c("unit.price","quantity"
,"tax","cogs", "gross.income", "rating"       ,"total" ))

#performing pca
df.pca <- prcomp(df_num, center = TRUE, scale. = TRUE)
summary(df.pca)

## Importance of components:
##                           PC1    PC2    PC3     PC4       PC5       PC6
## Standard deviation     2.2185 1.0002 0.9939 0.30001 2.981e-16 1.493e-16
## Proportion of Variance 0.7031 0.1429 0.1411 0.01286 0.000e+00 0.000e+00
## Cumulative Proportion  0.7031 0.8460 0.9871 1.00000 1.000e+00 1.000e+00
##                              PC7
## Standard deviation     9.831e-17
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

7 principal components were formed using the seven variables consisting of numerical (continuous) data. Principal component 1 explains 70.31% of the variance

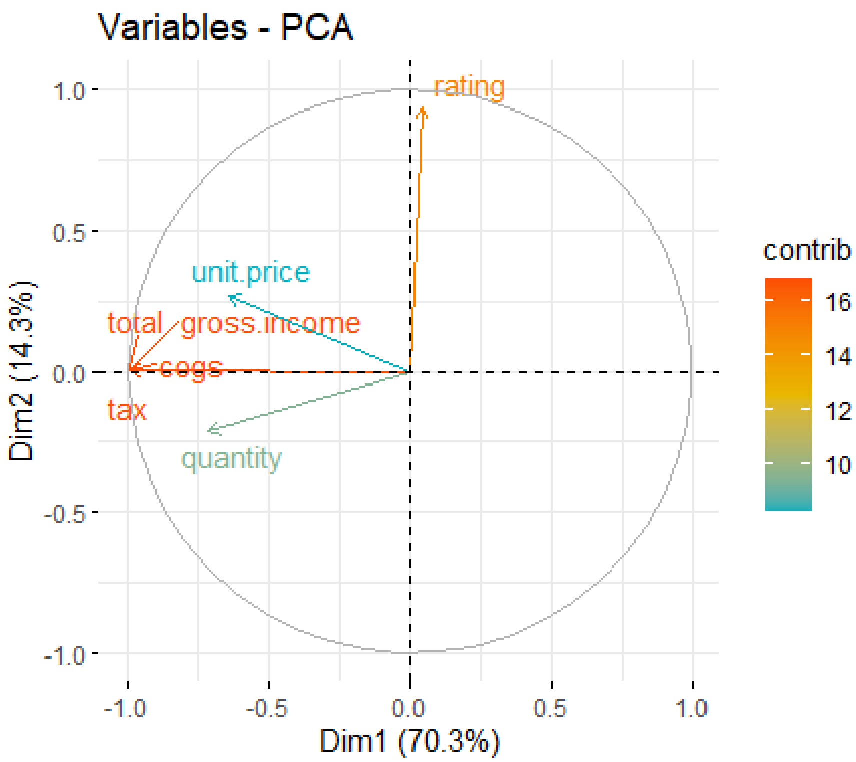
```
library(factoextra)

#visualising explained variances by the principal components
fviz_eig(df.pca)
```



Principal component 1 explains the most variance

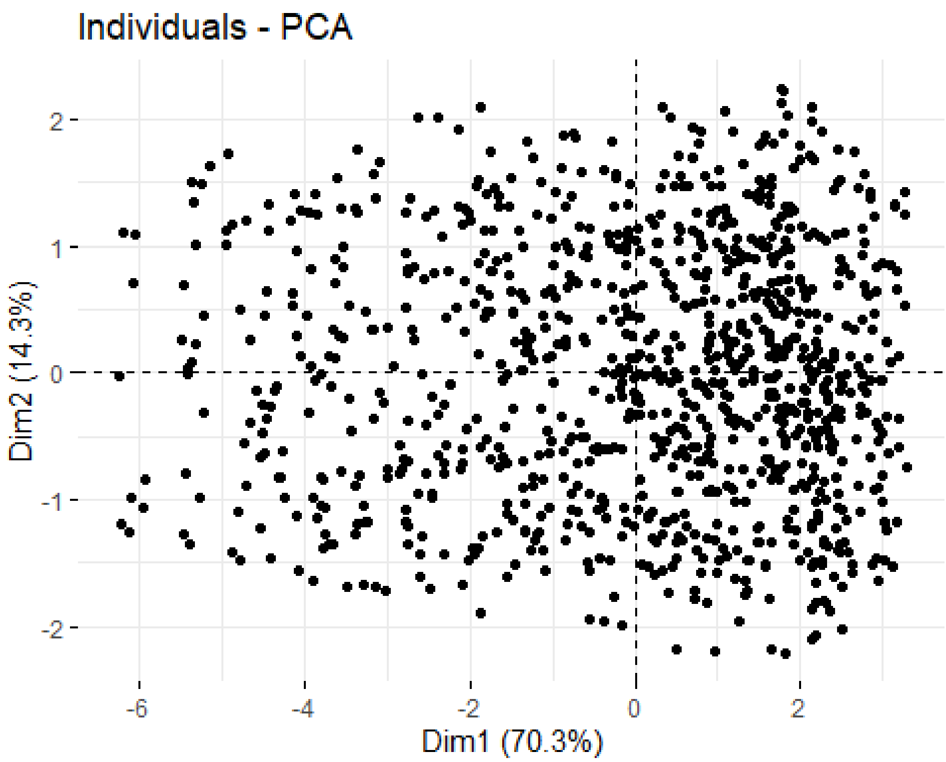
```
# looking at the PCA object
# ---
#
str(df.pca)

## List of 5
##  $ sdev    : num [1:7] 2.22 1.00 9.94e-01 3.00e-01 2.98e-16 ...
##  $ rotation: num [1:7, 1:7] -0.292 -0.325 -0.45 -0.45 -0.45 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:7] "unit.price" "quantity" "tax" "cogs" ...
##   .. ..$ : chr [1:7] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:7] 55.67 5.51 15.38 307.59 15.38 ...
##   ..- attr(*, "names")= chr [1:7] "unit.price" "quantity" "tax" "cogs" ...
##  $ scale   : Named num [1:7] 26.49 2.92 11.71 234.18 11.71 ...
##   ..- attr(*, "names")= chr [1:7] "unit.price" "quantity" "tax" "cogs" ...
##  $ x       : num [1:1000, 1:7] -2.005 2.306 -0.186 -1.504 -2.8 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:1000] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:7] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

```r
#checking how the variables contribute to each component
#looking at the absolute values, total income, tax, cogs and gross income con
tribute most to pc1
df.pca$rotation

##                      PC1          PC2          PC3         PC4           P
C5
## unit.price   -0.29176275  0.270866890 -0.693584569  0.60037161  6.582429e-
16
## quantity     -0.32452880 -0.212748396  0.633152868  0.66972877  7.430508e-
16
## tax          -0.44977957  0.004196356  0.001836202 -0.21835146 -8.277641e-
01
## cogs         -0.44977957  0.004196356  0.001836202 -0.21835146  9.549992e-
02
## gross.income -0.44977957  0.004196356  0.001836202 -0.21835146  2.290536e-
01
## rating        0.01867926  0.938775165  0.343575909 -0.01754621 -1.194541e-
17
## total        -0.44977957  0.004196356  0.001836202 -0.21835146  5.032105e-
01
##                        PC6           PC7
## unit.price    5.894232e-17 -7.635490e-17
## quantity      1.864419e-16 -1.721827e-16
## tax           1.656386e-02  2.540320e-01
## cogs         -5.810190e-01 -6.350565e-01
## gross.income  7.836445e-01 -2.888526e-01
## rating        1.850076e-17 -7.208985e-17
## total        -2.191893e-01  6.698770e-01

library(devtools)

## Loading required package: usethis

library(ggbiplot)

#plotting variable contributions pca
fviz_pca_var(df.pca,
             col.var = "contrib", # Color by contributions to the PC(1)?
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE     # Avoid text overlapping
             )
```

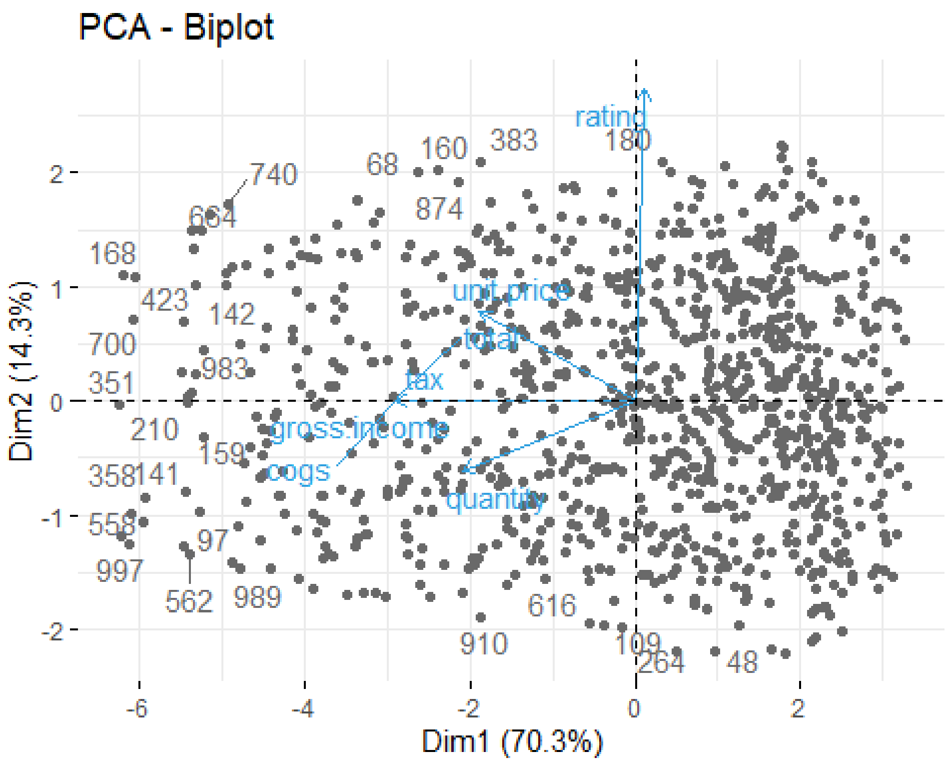


total income, tax, cogs and gross income contribute most to pc1

```
#plotting the observations with using first 2 principal components
fviz_pca_ind(df.pca,
             label="none"
             )
```

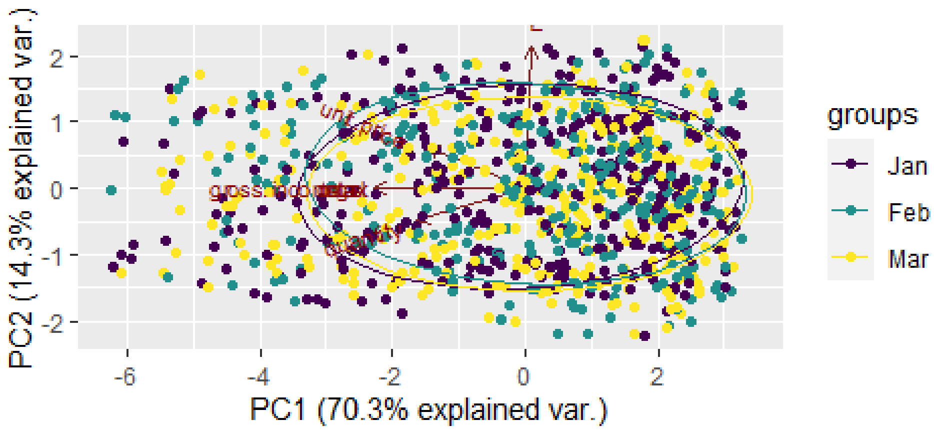



```r
#plot of observations and variables
fviz_pca_biplot(df.pca, repel = TRUE,
                col.var = "#2E9FDF", # Variables color
                col.ind = "#696969"  # Individuals color
                )

## Warning: ggrepel: 973 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

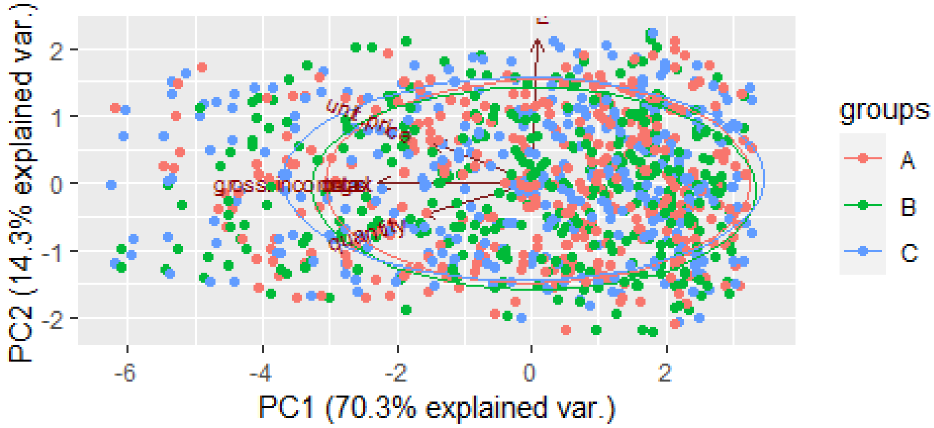


observations with higher values of Cogs, quantity, gross income, total, unit price and quantity are towards the left side of the plot. Most observations are towards the right side, indicating that most transactions had values on the lower end of these variables.

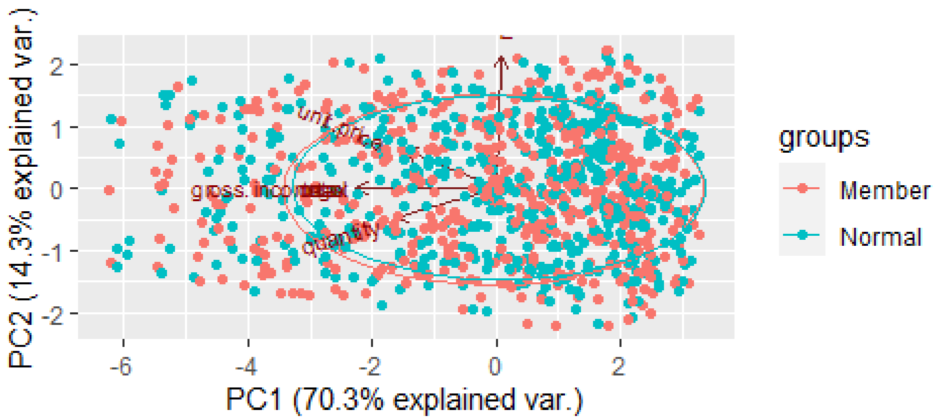
```
cat

## [1] "invoice.id"    "branch"         "customer.type" "gender"
## [5] "product.line"  "month"          "day"           "year"
## [9] "payment"

#colouring observations by month
ggbiplot(df.pca,ellipse=TRUE, groups=df$month, obs.scale = 1, var.scale = 1)
```




```
# no distinct clustering of observations. observations of the different group
s are spread out in a similar manner

#colouring observations by branch
ggbiplot(df.pca,ellipse=TRUE, groups=df$branch, obs.scale = 1, var.scale = 1)
```




```r
# observations of the different groups are spread out in a similar manner

#colouring observations by cuatomer type
ggbiplot(df.pca,ellipse=TRUE, groups=df$customer.type, obs.scale = 1, var.sca
le = 1)
```




```
# the groups are spread out similarly
```

##Part 2: Feature Selection

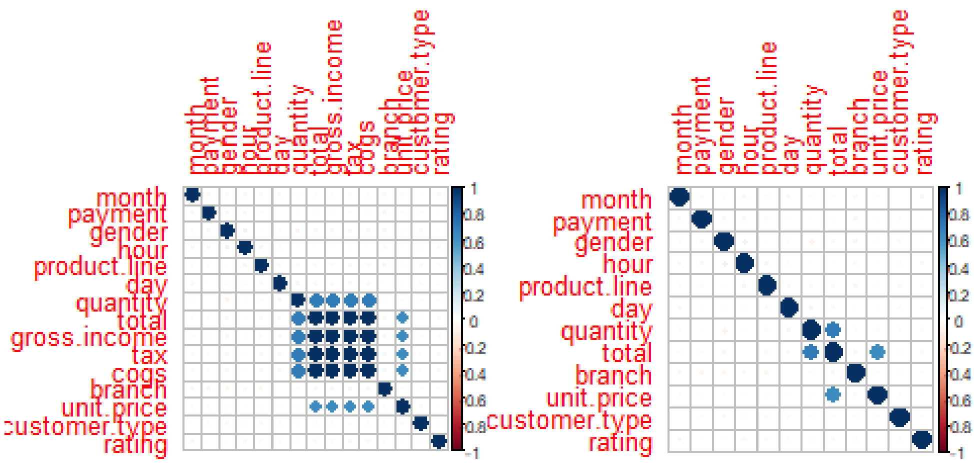
####Filter method

```
library(caret)
library(corrplot)

## corrplot 0.92 loaded

# Calculating the correlation matrix

matrix <- cor(enc_df)

# Finding variables that are highly correlated
# findCorrelation function - removes redundancy by correlation

#
highlyCorrelated <- findCorrelation(matrix
                                    , cutoff=0.80)

# Highly correlated attributes

highlyCorrelated

## [1]  7  9 10
```

```
names(enc_df[,highlyCorrelated])

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

Tax, cost of goods (cogs), and gross income have been identified as redundant features that will be dropped

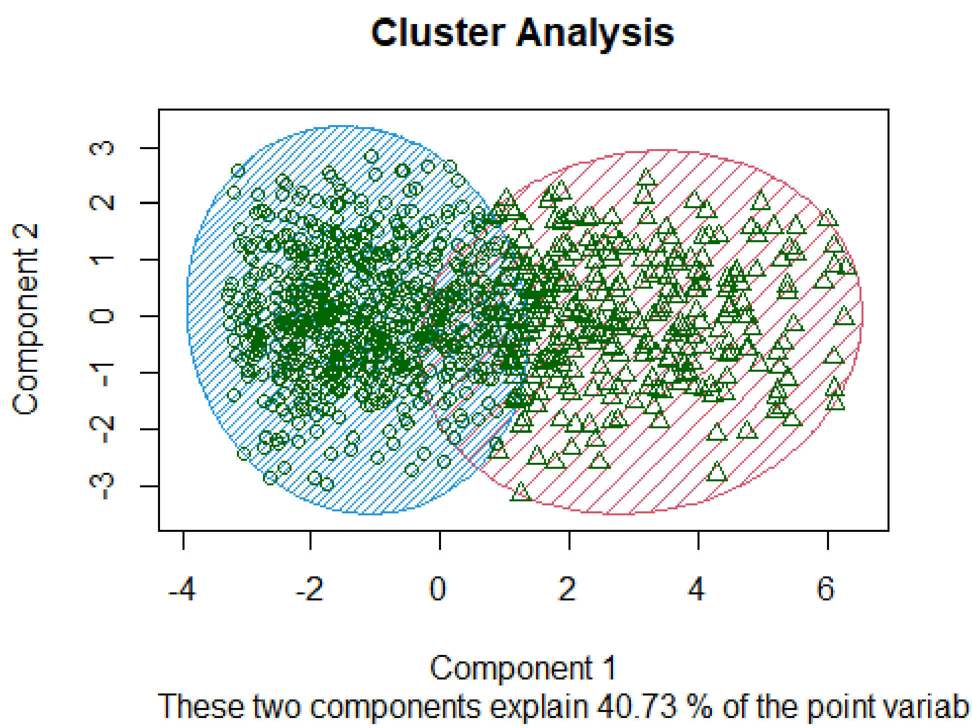
```
# Removing redundant features

enc_dfc<-enc_df[-highlyCorrelated]

# plotting the correlation matrices before and after removing redundant featu
res

par(mfrow = c(1, 2))# figure arrangement c(rows, columns)
corrplot(matrix, order = "hclust", tl.cex=0.8, cl.cex=0.5)
corrplot(cor(enc_dfc), order = "hclust", tl.cex=0.8, cl.cex=0.5)
```



####Embedded method

```
library(wskm)

## Loading required package: latticeExtra

##
## Attaching package: 'latticeExtra'
```

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

## Loading required package: fpc

# Model - entropy weighted k means.
# dropping the redundant columns identified above before modelling
set.seed(2)
model <- ewkm(subset(enc_df, select=-c(tax,cogs,gross.income)), 2)

library("cluster")

# cluster plot against first 2 principal components
clusplot(enc_df, model$cluster, color=TRUE, shade=TRUE,
         lines=1,main='Cluster Analysis')
```




```
# 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.

round(model$weights*100,2)

##   branch customer.type gender product.line unit.price quantity payment rat
ing
## 1      0         48.99  51.00            0          0        0       0
0
## 2      0         47.38  52.61            0          0        0       0
```

```
0
##   total month day hour
## 1     0     0   0    0
## 2     0     0   0    0
```

The variables that were determined to be most important in determining membership of observations to the various clusters included customer type and gender

##Part 3: Association Rules

```
# Loading the arules library
#
library(arules)

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
## Attaching package: 'arules'

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

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

# Loading the dataset from the csv file
# ---
# using read.transactions fuction which will load data from comma-separated f
iles
# and convert them to the class transactions for use in developing associatio
n rules
# ---
#
data <-"http://bit.ly/SupermarketDatasetII"

Transactions<-read.transactions(data, sep = ",")

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

Transactions

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

```r
# checking that object's class is transactions

class(Transactions)

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

# Previewing the first 10 transactions
#
inspect(Transactions[1:10])

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

```
## [9]  {frozen vegetables,
##       green tea,
##       spaghetti}
## [10] {french fries}
```

```
# previewing items in dataset
items<-as.data.frame(itemLabels(Transactions))
colnames(items) <- "Item"

# checking number of items
length(items$Item)
```

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

```
#previewing first 10
head(items, 10)
```

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

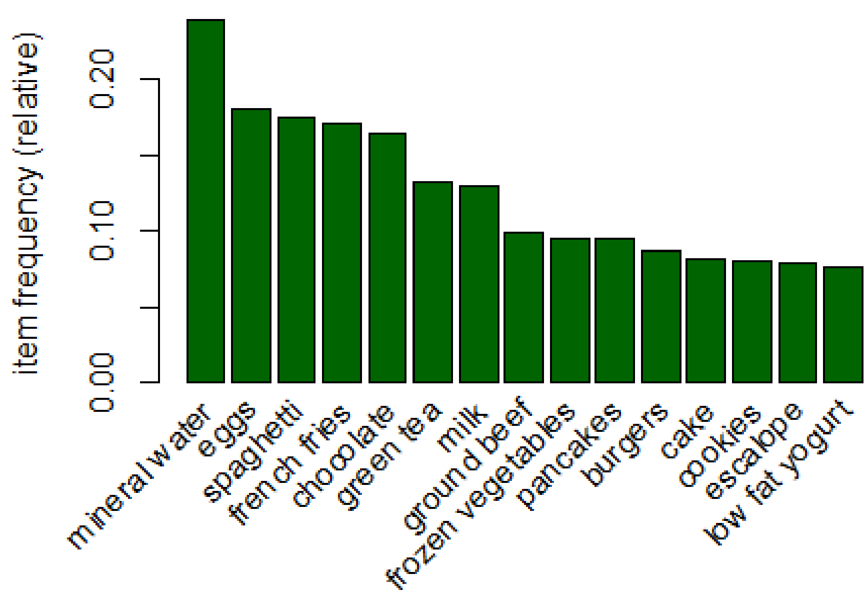
```
#there are 119 items in the dataset

# summary of the dataset

summary(Transactions)
```

```
## transactions as itemMatrix in sparse format with
##  7501 rows (elements/itemsets/transactions) and
##  119 columns (items) and a density of 0.03288973
##
## most frequent items:
## mineral water          eggs     spaghetti  french fries     chocolate
##          1788          1348          1306          1282          1229
##       (Other)
##         22405
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
16
## 1754 1358 1044  816  667  493  391  324  259  139  102   67   40   22   17
4
```

```
##   18   19   20
##    1    2    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.914   5.000  20.000
##
## includes extended item information - examples:
##              labels
## 1            almonds
## 2 antioxydant juice
## 3          asparagus
```

Most transactions involved one item

```r
# plotting the frequency of the top 15 most frequent items
itemFrequencyPlot(Transactions, topN = 15,col="darkgreen")
```



Mineral water is the most frequently occuring item in the transactions

```r
# Building a model based on association rules
# using the apriori function
# support - proportion of transactions in which an itemset appears
#confidence - How often one item A appears whenever another item B appears in a transaction. usually a conditional probability.
#lift


rules <- apriori (Transactions, parameter = list(supp = 0.01, conf = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 75
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.01s].
## sorting and recoding items ... [75 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules
```

```
## set of 0 rules
```

No rules obtained when the support is 0.01 and confidence 0.8.

```
#Decreasing the support and confidence
rulesa <- apriori (Transactions, parameter = list(supp = 0.001, conf = 0.7))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.7    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 7
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.01s].
## sorting and recoding items ... [116 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 6 done [0.02s].
```

```
## writing ... [200 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

rulesa

## set of 200 rules
```

200 rules obtained with lower support of 0.001 and confidence of 0.7

```
# summary of the model
# most rules have 4 items
# lift - A rule with a lift ~ 1 would imply that the probability of occurrenc
e of the antecedent and that of the consequent are independent of each other.
# A rule with a lift of > 1 it would imply that those two occurrences are dep
endent on one another and useful for predicting in future datasets.

summary(rulesa)

## set of 200 rules
##
## rule length distribution (lhs + rhs):sizes
##   3   4   5   6
##  44 122  33   1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   4.000   4.000   3.955   4.000   6.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001067   Min.   :0.7000   Min.   :0.001067   Min.   : 2.937
##  1st Qu.:0.001067   1st Qu.:0.7273   1st Qu.:0.001466   1st Qu.: 3.088
##  Median :0.001200   Median :0.7500   Median :0.001466   Median : 3.616
##  Mean   :0.001330   Mean   :0.7767   Mean   :0.001728   Mean   : 4.160
##  3rd Qu.:0.001466   3rd Qu.:0.8139   3rd Qu.:0.001866   3rd Qu.: 4.418
##  Max.   :0.003066   Max.   :1.0000   Max.   :0.004133   Max.   :12.722
##      count
##  Min.   : 8.00
##  1st Qu.: 8.00
##  Median : 9.00
##  Mean   : 9.98
##  3rd Qu.:11.00
##  Max.   :23.00
##
## mining info:
##          data ntransactions support confidence
##  Transactions          7501   0.001        0.7
##                                                                       call
##  apriori(data = Transactions, parameter = list(supp = 0.001, conf = 0.7))

# Ordering the rules by confidence and previewing the top 10
```

```
sorted_rules<-sort(rulesa, by="confidence", decreasing=TRUE)

inspect(sorted_rules[1:10])

##      lhs                        rhs                 support confidence
coverage      lift count
## [1]  {french fries,
##       mushroom cream sauce,
##       pasta}                 => {escalope}      0.001066524  1.0000000 0.0
01066524 12.606723     8
## [2]  {ground beef,
##       light cream,
##       olive oil}             => {mineral water} 0.001199840  1.0000000 0.0
01199840  4.195190     9
## [3]  {cake,
##       meatballs,
##       mineral water}         => {milk}          0.001066524  1.0000000 0.0
01066524  7.717078     8
## [4]  {cake,
##       olive oil,
##       shrimp}                => {mineral water} 0.001199840  1.0000000 0.0
01199840  4.195190     9
## [5]  {mushroom cream sauce,
##       pasta}                 => {escalope}      0.002532996  0.9500000 0.0
02666311 11.976387    19
## [6]  {red wine,
##       soup}                  => {mineral water} 0.001866418  0.9333333 0.0
01999733  3.915511    14
## [7]  {eggs,
##       mineral water,
##       pasta}                 => {shrimp}        0.001333156  0.9090909 0.0
01466471 12.722185    10
## [8]  {herb & pepper,
##       mineral water,
##       rice}                  => {ground beef}   0.001333156  0.9090909 0.0
01466471  9.252498    10
## [9]  {ground beef,
##       pancakes,
##       whole wheat rice}      => {mineral water} 0.001333156  0.9090909 0.0
01466471  3.813809    10
## [10] {frozen vegetables,
##       milk,
##       spaghetti,
##       turkey}                => {mineral water} 0.001199840  0.9000000 0.0
01333156  3.775671     9

#the first rule can be interpreted as: if an individual buys french fries, mu
shroom cream sauce and pasta, they are 100% likely to buy escalope. The remai
ning rules can be interpreted similarly
```

Examining rules related some of the top 10 frequently occuring items:

Milk

```
# ---
# Items that the customers bought before purchasing milk
# ---
#
milk <- subset(rulesa, subset = rhs %pin% "milk")

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

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

# Items that the customers might buy after purchasing milk
# ---
#
milk <- subset(rulesa, subset = lhs %pin% "milk")

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

##     lhs                     rhs                     support confidence
coverage     lift count
## [1] {frozen vegetables,
##      milk,
##      spaghetti,
##      turkey}             => {mineral water}     0.001199840  0.9000000 0.0
01333156 3.775671     9
## [2] {cake,
##      meatballs,
##      milk}               => {mineral water}     0.001066524  0.8888889 0.0
01199840 3.729058     8
## [3] {burgers,
##      milk,
##      salmon}             => {spaghetti}         0.001066524  0.8888889 0.0
01199840 5.105326     8
```

```
## [4] {chocolate,
##      ground beef,
##      milk,
##      mineral water,
##      spaghetti}          => {frozen vegetables} 0.001066524  0.8888889 0.0
01199840 9.325253     8
## [5] {ground beef,
##      nonfat milk}        => {mineral water}     0.001599787  0.8571429 0.0
01866418 3.595877    12
```

Ground beef:

```
# items that the customers bought before purchasing ground beef

#
groundbeef <- subset(rulesa, subset = rhs %pin% "ground beef")

# Then order by confidence
groundbeef<-sort(groundbeef, by="confidence", decreasing=TRUE)
inspect(groundbeef)

##     lhs                                     rhs           support
## [1] {herb & pepper, mineral water, rice} => {ground beef} 0.001333156
## [2] {grated cheese, mineral water, rice} => {ground beef} 0.001066524
## [3] {burgers, herb & pepper, spaghetti}  => {ground beef} 0.001333156
## [4] {green tea, spaghetti, tomato sauce} => {ground beef} 0.001333156
##     confidence coverage    lift     count
## [1] 0.9090909  0.001466471 9.252498 10
## [2] 0.8888889  0.001199840 9.046887  8
## [3] 0.7692308  0.001733102 7.829037 10
## [4] 0.7142857  0.001866418 7.269820 10

# determining items that customers might buy if they have previously bought g
round beef
# ---
#
# Subset the rules
gbeef <- subset(rulesa, subset = lhs %pin% "ground beef")

# Ordering by confidence
gbeeft<-sort(gbeef, by="confidence", decreasing=TRUE)

# inspect top 5
inspect(gbeef[1:5])

##     lhs                                     rhs             support
## [1] {green beans, ground beef}           => {spaghetti}     0.001066524
## [2] {ground beef, whole weat flour}      => {mineral water} 0.001066524
## [3] {ground beef, nonfat milk}           => {mineral water} 0.001599787
## [4] {extra dark chocolate, ground beef}  => {spaghetti}     0.001466471
## [5] {green tea, ground beef, tomato sauce} => {spaghetti}   0.001333156
```

```
##     confidence coverage    lift     count
## [1] 0.7272727  0.001466471 4.177085  8
## [2] 0.7272727  0.001466471 3.051047  8
## [3] 0.8571429  0.001866418 3.595877 12
## [4] 0.7333333  0.001999733 4.211894 11
## [5] 0.8333333  0.001599787 4.786243 10
```

Eggs:

```
# items that the customers bought before purchasing eggs
# ---
#
eggs <- subset(rulesa, subset = rhs %pin% "eggs")

# Then order by confidence
eggs<-sort(eggs, by="confidence", decreasing=TRUE)
inspect(eggs)

##     lhs                               rhs    support     confidence covera
ge
## [1] {black tea, spaghetti, turkey} => {eggs} 0.001066524 0.8888889  0.0011
99840
## [2] {mineral water, pasta, shrimp} => {eggs} 0.001333156 0.8333333  0.0015
99787
## [3] {black tea, turkey}            => {eggs} 0.001466471 0.7333333  0.0019
99733
##     lift     count
## [1] 4.946258  8
## [2] 4.637117 10
## [3] 4.080663 11

# items that the customers might buy if they purchase eggs
# ---
#
eggs <- subset(rulesa, subset = lhs %pin% "eggs")

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

##     lhs                     rhs                  support confidence    cove
rage      lift count
## [1] {eggs,
##      mineral water,
##      pasta}             => {shrimp}        0.001333156  0.9090909 0.00146
6471 12.722185    10
## [2] {brownies,
##      eggs,
##      ground beef}       => {mineral water} 0.001066524  0.8888889 0.00119
9840  3.729058     8
## [3] {chocolate,
```

```
##      eggs,
##      frozen vegetables,
##      ground beef}        => {mineral water} 0.001466471  0.8461538 0.00173
3102  3.549776    11
## [4] {chocolate,
##      eggs,
##      olive oil,
##      spaghetti}          => {mineral water} 0.001199840  0.8181818 0.00146
6471  3.432428     9
## [5] {cooking oil,
##      eggs,
##      olive oil}          => {mineral water} 0.001066524  0.8000000 0.00133
3156  3.356152     8
```

##Part 4: Anomaly Detection

```
# Load tidyverse and anomalize
# ---
#
library(tidyverse)
library(anomalize)

## ══ Use anomalize to improve your Forecasts by 50%! ═══════════════════════
══════
## Business Science offers a 1-hour course - Lab #18: Time Series Anomaly Det
ection!
## </> Learn more at: https://university.business-science.io/p/learning-labs-
pro </>

data <- fread("http://bit.ly/CarreFourSalesDataset")

data <- tibble::as_tibble(data)
```

###Checking the Data

Determining the no. of records in the dataset:

```
dim(data)

## [1] 1000    2

#the dataset has 1000 rows and 2  columns
```

Previewing the top of the dataset:

```
head(data)

## # A tibble: 6 × 2
##   Date      Sales
##   <chr>     <dbl>
## 1 1/5/2019  549.
## 2 3/8/2019   80.2
## 3 3/3/2019  341.
```

```
## 4 1/27/2019 489.
## 5 2/8/2019  634.
## 6 3/25/2019 628.
```

Previewing the bottom of the dataset:

```
tail(data)
```

```
## # A tibble: 6 × 2
##   Date       Sales
##   <chr>      <dbl>
## 1 2/18/2019   64.0
## 2 1/29/2019   42.4
## 3 3/2/2019  1022.
## 4 2/9/2019    33.4
## 5 2/22/2019   69.1
## 6 2/18/2019  649.
```

Checking datatype of each column:

```
#date column needs to be converted from character to datetime
str(data)
```

```
## tibble [1,000 × 2] (S3: tbl_df/tbl/data.frame)
##  $ Date : chr [1:1000] "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ Sales: num [1:1000] 549 80.2 340.5 489 634.4 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

###Tidying the Dataset

```
#checking column names
colnames(data)
```

```
## [1] "Date"  "Sales"
```

```
#converting column names to lowercase
colnames(data) = tolower(colnames(data))
colnames(data)
```

```
## [1] "date"  "sales"
```

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

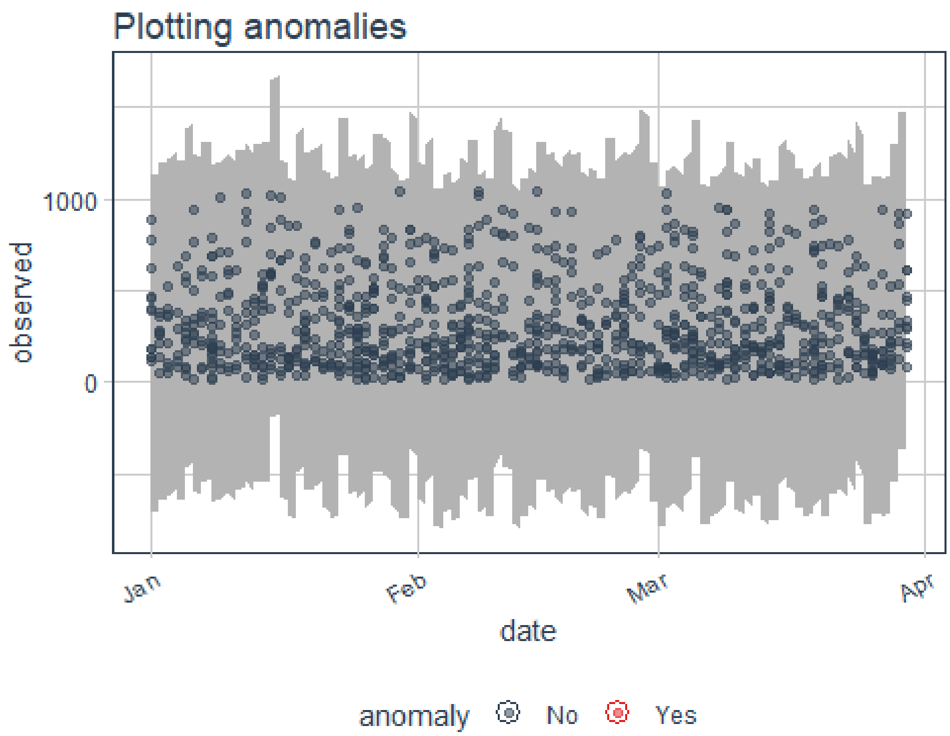
```
##       colSums.is.na.data..
## date                     0
## sales                    0
```

```
#no missing values in any of the columns
```

```
#checking for duplicates
nrow(data[duplicated(data),])
```

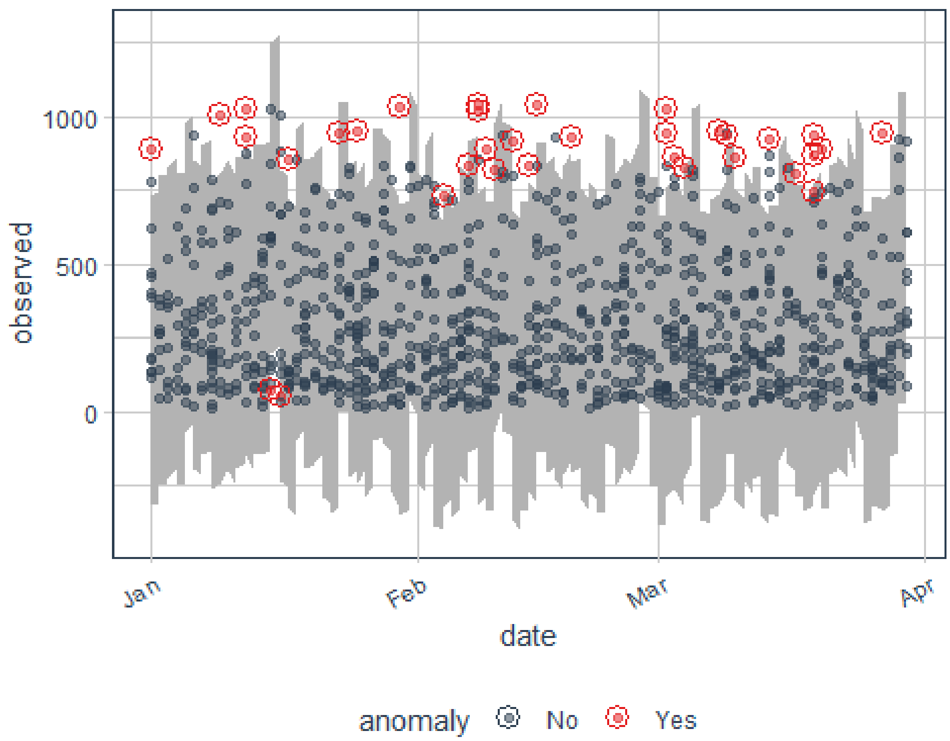
```
## [1] 0

#no duplicates in the dataset

#loading the lubridate library to work with dates
library(lubridate)

# converting date to posixct
str(data$date)

##  chr [1:1000] "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" "2/8/2019" ...

data$date <- as.POSIXct(data$date, format="%m/%d/%Y")
str(data$date)

##  POSIXct[1:1000], format: "2019-01-05" "2019-03-08" "2019-03-03" "2019-01-
27" "2019-02-08" ...

#ordering by date in ascending order
data<-data[order(data$date, decreasing=FALSE),]
head(data, 10)

## # A tibble: 10 × 2
##    date                sales
##    <dttm>              <dbl>
##  1 2019-01-01 00:00:00  457.
##  2 2019-01-01 00:00:00  400.
##  3 2019-01-01 00:00:00  471.
##  4 2019-01-01 00:00:00  388.
##  5 2019-01-01 00:00:00  133.
##  6 2019-01-01 00:00:00  132.
##  7 2019-01-01 00:00:00  621.
##  8 2019-01-01 00:00:00  114.
##  9 2019-01-01 00:00:00  779.
## 10 2019-01-01 00:00:00  184.

#decomposing the sales column into "observed", "season", "trend", and "remain
der" columns.
td_sales <- data %>%
    time_decompose(sales)

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

## frequency = 11 seconds

## trend = 11 seconds

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

head(td_sales)
```

```
## # A time tibble: 6 × 5
## # Index: date
##   date                observed season trend remainder
##   <dttm>                 <dbl>  <dbl> <dbl>     <dbl>
## 1 2019-01-01 00:00:00     457. -11.7   455.      14.1
## 2 2019-01-01 00:00:00     400.  -5.11  420.     -14.9
## 3 2019-01-01 00:00:00     471.   6.94  384.      79.3
## 4 2019-01-01 00:00:00     388.  20.7   355.      12.2
## 5 2019-01-01 00:00:00     133. -27.2   326.    -166.
## 6 2019-01-01 00:00:00     132. -12.3   303.    -159.

# anomaly detection on the decomposed data using
# the remainder column through
# produces 3 new columns; “remainder_l1” (lower limit),
# “remainder_l2” (upper limit), and “anomaly” (Yes/No Flag).
#default alpha is 0.05
anom <- td_sales %>%
    anomalize(remainder)
head(anom[6:8])

## # A tibble: 6 × 3
##   remainder_l1 remainder_l2 anomaly
##          <dbl>        <dbl> <chr>
## 1        -912.         929. No
## 2        -912.         929. No
## 3        -912.         929. No
## 4        -912.         929. No
## 5        -912.         929. No
## 6        -912.         929. No

#creating the lower and upper bounds around the “observed” values and plottin
g anomalies
anom %>% time_recompose() %>%
plot_anomalies(time_recomposed = TRUE, alpha_dots = 0.5) + ggtitle("Plotting
anomalies")
```



No anomalies observed with alpha at default 0.05

```
#adjusting alpha  parameter
data %>%
    time_decompose(sales) %>%
    anomalize(remainder, alpha=0.1) %>%
    time_recompose() %>%
    plot_anomalies(time_recomposed = TRUE, alpha_dots = 0.5)

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

## frequency = 11 seconds

## trend = 11 seconds
```



With alpha set at 0.1, there are some anomalous sales identified

```
#exploring anomalous sales
datar <- data %>%
    time_decompose(sales) %>%
    anomalize(remainder, alpha=0.1) %>%
    time_recompose()

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

## frequency = 11 seconds

## trend = 11 seconds

datao <- subset(filter(datar, anomaly == "Yes"), select=c(date, observed, ano
maly))
datao <- data.frame(datao)

datao[order(datao$observed, decreasing=TRUE),]

##            date  observed anomaly
## 19 2019-02-15 1042.6500     Yes
## 13 2019-02-08 1039.2900     Yes
## 10 2019-01-30 1034.4600     Yes
## 4  2019-01-12 1023.7500     Yes
## 22 2019-03-02 1022.4900     Yes
## 14 2019-02-08 1020.7050     Yes
```

```
## 2  2019-01-09 1002.1200     Yes
## 25 2019-03-08  951.8250     Yes
## 9  2019-01-25  950.2500     Yes
## 34 2019-03-27  943.2990     Yes
## 8  2019-01-23  942.9000     Yes
## 21 2019-03-02  942.4485     Yes
## 31 2019-03-19  937.8180     Yes
## 26 2019-03-09  935.2665     Yes
## 20 2019-02-19  932.3370     Yes
## 3  2019-01-12  931.0350     Yes
## 28 2019-03-14  921.1860     Yes
## 17 2019-02-12  914.5500     Yes
## 1  2019-01-01  888.6150     Yes
## 15 2019-02-09  888.4050     Yes
## 33 2019-03-20  887.9220     Yes
## 30 2019-03-19  867.6150     Yes
## 23 2019-03-03  860.6850     Yes
## 27 2019-03-10  860.4750     Yes
## 7  2019-01-17  852.7050     Yes
## 18 2019-02-14  836.3040     Yes
## 12 2019-02-07  833.5950     Yes
## 24 2019-03-04  829.0800     Yes
## 16 2019-02-10  820.3650     Yes
## 29 2019-03-17  807.6600     Yes
## 32 2019-03-19  743.8200     Yes
## 11 2019-02-04  734.0760     Yes
## 5  2019-01-15   72.0090     Yes
## 6  2019-01-16   53.9280     Yes
```

##Conclusion

Following data preparation (where missing values, duplicates, outliers, column creation etc were dealt with accordingly), univariate and bivariate analysis were carried out on the first dataset providing valuable insights. Some general bivariate analysis insights include: member customers, women and Saturdays have higher average sales when compared to non-members, men and other days of the week, respectively, etc.

PCA: 7 principal components were formed using the seven variables consisting of numerical (continuous) data. Principal component 1 explains 70.31% of the variance. Observations with higher values of Cogs, quantity, gross income, total, unit price and quantity were towards the left side of the plot. Most observations are towards the right side, indicating that most transactions had values on the lower end of these variables. Colouring the observations by unique values of branch, month, and customer type did not reveal a distinct difference in how the observations belonging to the different groups were spread.

Feature selection - The first method used was filtering. Tax, cost of goods (cogs), and gross income were identified as redundant features that were dropped. - The next approach was of an embedded method. Model - entropy weighted k means. The variables that were determined to be most important in determining membership of observations to the various clusters included customer type and gender. The redundant features identified in filtering were dropped before modelling.

Association analysis: Different values of support and confidence were tested (200 rules obtained with a support of 0.001 and confidence of 0.7). The top rules by confidence were identified, and rules related to some of the top 10 frequently occuring items were examined.

Anomaly detection: Anomalous sales and their dates were identified, and ranked in descending order.

##Recommendations

Top 10 frequently bought items are mineral water, eggs, spaghetti, French fries, chocolate, green tea, milk, ground beef, frozen vegetables and pancakes. Some strategies to further promote the sales of some of these items:

- Have occasional sales(discounting the price) on these items as this would encourage people to increase the quantity they are purchasing and therefore boost overall amount of sales.
- Ground beef is often bought with spaghetti. A promotion such as buy x kgs of ground beef and get spaghetti free could boost the sales of the beef which is a high value commodity.
- A similar promotion could be done with eggs and black tea, and other item sets that have fast moving commodities.

The average sales were higher on Saturdays, therefore it is a key day for promotions to be run.

Fraud detection - some transactions on various dates were flagged as being anomalous and should therefore be investigated further to determine if they were legitimate.