Carrefour Kenya Analysis

Naomi Chebet

9/14/2020

1. Business Understanding

1.1 Define the Question

As a Data analyst at Carrefour Kenya, you are tasked with conducting analysis of sales data to inform the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax). In this project you are required to implement PCA and Feature Selection methods.

1.2 Understanding the Context

Carrefour is a French Multinational Corporation that specializes in retail. In 2016 Carrefour opened it's stores in Kenya to what is now called Carrefour Kenya. Carrefour Kenya competes with other big retail shops such as Quickmart and Naivas. As a data scientist, you've been provided with sales data to conduct analysis, the results from the analysis will inform the marketing team on the best strategies that will lead to high number of sales.

The dataset provided has 1000 entries and 16 columns, some of the columns in the dataset are sales, Branch, Customer.type,Gender, Product.line, Quantity, and Date.

1.3 Metrics of Success

1. Successful implementation of PCA and obtaining insights from the analysis
2. Successful implementation of at least two feature selection methods
3. Providing recommendation based on EDA and solution implementation analysis

1.4 Experimental Design

The flow of our project includes:

1. Business Understanding
2. Loading and Checking the Data
3. Tidying the Data
4. EDA with Univariate, and Bivariate Analysis
5. Implementing the solution with PCA and Feature Selection Methods

2. Importing libraries

```
library(tidyverse)
```

```
## -- Attaching packages ----------------
```

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

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

```r
library(ggbiplot)
```

```
## Loading required package: plyr

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

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

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

##
## Attaching package: 'plyr'

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

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

## Loading required package: scales

##
## Attaching package: 'scales'

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

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

## Loading required package: grid
```

```r
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```r
library(devtools)
```

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

```r
library(tidyr)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(ggcorrplot)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```r
library(clustvarsel)
```

```
## Loading required package: mclust

## Package 'mclust' version 5.4.6
## Type 'citation("mclust")' for citing this R package in publications.

##
## Attaching package: 'mclust'

## The following object is masked from 'package:purrr':
##
##     map

## Package 'clustvarsel' version 2.3.3

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

```r
library(mclust)
library(wskm)
```

```
## Loading required package: latticeExtra

##
## Attaching package: 'latticeExtra'

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

## Loading required package: fpc
```

```r
library(cluster)
# library(FSelector)
```

3. Loading and Checking the Data

```r
sales <- read.csv("~/Moringa School/R Programming/R datasets/Supermarket_Dataset_1 - Sales Data.csv")
glimpse(sales)
```

```
## 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", "No...
## $ Gender                 <chr> "Female", "Female", "Male", "Male", "Male",...
## $ Product.line           <chr> "Health and beauty", "Electronic accessorie...
## $ Unit.price             <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ Quantity               <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ Tax                    <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ Date                   <chr> "1/5/2019", "3/8/2019", "3/3/2019", "1/27/2...
## $ 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....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.income           <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ Rating                 <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ Total                  <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634....
```

```r
class(sales)
```

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

```r
# previewing the top of the data
head(sales)
```

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

```r
# previewing the bottom
tail(sales)
```

```
##       Invoice.ID Branch Customer.type Gender           Product.line Unit.price
## 995  652-49-6720      C        Member Female Electronic accessories      60.95
## 996  233-67-5758      C        Normal   Male      Health and beauty      40.35
## 997  303-96-2227      B        Normal Female     Home and lifestyle      97.38
## 998  727-02-1313      A        Member   Male     Food and beverages      31.84
## 999  347-56-2442      A        Normal   Male     Home and lifestyle      65.82
## 1000 849-09-3807      A        Member Female    Fashion accessories      88.34
##      Quantity     Tax      Date  Time Payment   cogs gross.margin.percentage
## 995         1  3.0475 2/18/2019 11:40 Ewallet  60.95                4.761905
```

```
## 996         1  2.0175 1/29/2019 13:46 Ewallet  40.35                4.761905
## 997        10 48.6900  3/2/2019 17:16 Ewallet 973.80                4.761905
## 998         1  1.5920  2/9/2019 13:22    Cash  31.84                4.761905
## 999         1  3.2910 2/22/2019 15:33    Cash  65.82                4.761905
## 1000        7 30.9190 2/18/2019 13:28    Cash 618.38                4.761905
##      gross.income Rating     Total
## 995        3.0475    5.9   63.9975
## 996        2.0175    6.2   42.3675
## 997       48.6900    4.4 1022.4900
## 998        1.5920    7.7   33.4320
## 999        3.2910    4.1   69.1110
## 1000      30.9190    6.6  649.2990
```

```r
# checking column names
colnames(sales)
```

```
##  [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 the total number of unique values in each column
sapply(sales, function(x)length(unique(x)))
```

```
##              Invoice.ID                  Branch           Customer.type
##                    1000                       3                       2
##                  Gender            Product.line              Unit.price
##                       2                       6                     943
##                Quantity                     Tax                    Date
##                      10                     990                      89
##                    Time                 Payment                    cogs
##                     506                       3                     990
## gross.margin.percentage            gross.income                  Rating
##                       1                     990                      61
##                   Total
##                     990
```

```r
# displaying the unique values in columns of interest
unique.df <- select(sales, Branch, Customer.type,Gender, Product.line, Quantity, Date,
                    Payment, gross.margin.percentage, Rating)
un <- lapply(unique.df, unique)
un
```

```
## $Branch
## [1] "A" "C" "B"
##
## $Customer.type
## [1] "Member" "Normal"
##
## $Gender
## [1] "Female" "Male"
##
## $Product.line
```

```
## [1] "Health and beauty"      "Electronic accessories" "Home and lifestyle"
## [4] "Sports and travel"      "Food and beverages"     "Fashion accessories"
##
## $Quantity
##  [1]  7  5  8  6 10  2  3  4  1  9
##
## $Date
##  [1] "1/5/2019"  "3/8/2019"  "3/3/2019"  "1/27/2019" "2/8/2019"  "3/25/2019"
##  [7] "2/25/2019" "2/24/2019" "1/10/2019" "2/20/2019" "2/6/2019"  "3/9/2019"
## [13] "2/12/2019" "2/7/2019"  "3/29/2019" "1/15/2019" "3/11/2019" "1/1/2019"
## [19] "1/21/2019" "3/5/2019"  "3/15/2019" "2/17/2019" "3/2/2019"  "3/22/2019"
## [25] "3/10/2019" "1/25/2019" "1/28/2019" "1/7/2019"  "3/23/2019" "1/17/2019"
## [31] "2/2/2019"  "3/4/2019"  "3/16/2019" "2/27/2019" "2/10/2019" "3/19/2019"
## [37] "2/3/2019"  "3/7/2019"  "2/28/2019" "3/27/2019" "1/20/2019" "3/12/2019"
## [43] "2/15/2019" "3/6/2019"  "2/14/2019" "3/13/2019" "1/24/2019" "1/6/2019"
## [49] "2/11/2019" "1/22/2019" "1/13/2019" "1/9/2019"  "1/12/2019" "1/26/2019"
## [55] "1/23/2019" "2/23/2019" "1/2/2019"  "2/9/2019"  "3/26/2019" "3/1/2019"
## [61] "2/1/2019"  "3/28/2019" "3/24/2019" "2/5/2019"  "1/19/2019" "1/16/2019"
## [67] "1/8/2019"  "2/18/2019" "1/18/2019" "2/16/2019" "2/22/2019" "1/29/2019"
## [73] "1/4/2019"  "3/30/2019" "1/30/2019" "1/3/2019"  "3/21/2019" "2/13/2019"
## [79] "1/14/2019" "3/18/2019" "3/20/2019" "2/21/2019" "1/31/2019" "1/11/2019"
## [85] "2/26/2019" "3/17/2019" "3/14/2019" "2/4/2019"  "2/19/2019"
##
## $Payment
## [1] "Ewallet"     "Cash"        "Credit card"
##
## $gross.margin.percentage
## [1] 4.761905
##
## $Rating
##  [1]  9.1  9.6  7.4  8.4  5.3  4.1  5.8  8.0  7.2  5.9  4.5  6.8  7.1  8.2  5.7
## [16]  4.6  6.9  8.6  4.4  4.8  5.1  9.9  6.0  8.5  6.7  7.7  7.5  7.0  4.7  7.6
## [31]  7.9  6.3  5.6  9.5  8.1  6.5  6.1  6.6  5.4  9.3 10.0  6.4  4.3  4.0  8.7
## [46]  9.4  5.5  8.3  7.3  4.9  4.2  9.2  7.8  5.2  9.0  8.8  6.2  9.8  9.7  5.0
## [61]  8.9
```

There are three branches: A, B, C. Customer type, we have Member and Normal. We have 6 product lines. Quantity, this is the number of items that a customer bought. We have different options from 1 - 10. There are three payment options: Ewallet, Cash or Credit Card The gross margin percent is constant at 4.761905

```r
# converting column names to lower case for uniformity purposes
colnames(sales) <- tolower(colnames(sales))
glimpse(sales)
```

```
## 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", "No...
## $ gender                  <chr> "Female", "Female", "Male", "Male", "Male",...
## $ product.line            <chr> "Health and beauty", "Electronic accessorie...
## $ unit.price              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ quantity                <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ tax                     <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ date                    <chr> "1/5/2019", "3/8/2019", "3/3/2019", "1/27/2...
```

```
## $ 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....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.income            <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ rating                  <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ total                   <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634....
```

```
# splitting the Date column to date, month and year
sales <- sales %>%
    mutate(date = mdy(date)) %>%
    mutate_at(vars(date), funs(month, day, year)) %>%
    glimpse() # to confirm that the data column was successfully split
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
## Rows: 1,000
## Columns: 19
## $ 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", "No...
## $ gender                  <chr> "Female", "Female", "Male", "Male", "Male",...
## $ product.line            <chr> "Health and beauty", "Electronic accessorie...
## $ unit.price              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ quantity                <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ tax                     <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ date                    <date> 2019-01-05, 2019-03-08, 2019-03-03, 2019-0...
## $ 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....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.income            <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ rating                  <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ total                   <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634....
## $ month                   <dbl> 1, 3, 3, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2, 2, 3...
## $ day                     <int> 5, 8, 3, 27, 8, 25, 25, 24, 10, 20, 6, 9, 1...
## $ year                    <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2...
```

```
# finding the unique values in month, year and day
unique.date <- select(sales, month, day, year)
uni.date <- sapply(unique.date, unique)
uni.date
```

```
## $month
```

```
## [1] 1 3 2
##
## $day
##  [1]  5  8  3 27 25 24 10 20  6  9 12  7 29 15 11  1 21 17  2 22 28 23  4 16 19
## [26] 14 13 26 18 30 31
##
## $year
## [1] 2019
```

There are three months in the dataset: January, February and March. The data is from 2019, and all the dates of a month are listed. There are no strange values in the month, day and year columns.

```r
# converting the time column into time format

sales <- sales %>% mutate(time = hm(time))
glimpse(sales)
```

```
## Rows: 1,000
## Columns: 19
## $ 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", "No...
## $ gender                  <chr> "Female", "Female", "Male", "Male", "Male",...
## $ product.line            <chr> "Health and beauty", "Electronic accessorie...
## $ unit.price              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ quantity                <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ tax                     <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ date                    <date> 2019-01-05, 2019-03-08, 2019-03-03, 2019-0...
## $ time                    <Period> 13H 8M 0S, 10H 29M 0S, 13H 23M 0S, 20H 3...
## $ payment                 <chr> "Ewallet", "Cash", "Credit card", "Ewallet"...
## $ cogs                    <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.income            <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ rating                  <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ total                   <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634....
## $ month                   <dbl> 1, 3, 3, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2, 2, 3...
## $ day                     <int> 5, 8, 3, 27, 8, 25, 25, 24, 10, 20, 6, 9, 1...
## $ year                    <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2...
```

```r
# splitting time to hour and minute into a new column
sales <- sales %>% mutate(hour = hour(time), minute = minute(time))
glimpse(sales)
```

```
## Rows: 1,000
## Columns: 21
## $ 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", "No...
## $ gender                  <chr> "Female", "Female", "Male", "Male", "Male",...
## $ product.line            <chr> "Health and beauty", "Electronic accessorie...
## $ unit.price              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ quantity                <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ tax                     <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ date                    <date> 2019-01-05, 2019-03-08, 2019-03-03, 2019-0...
## $ time                    <Period> 13H 8M 0S, 10H 29M 0S, 13H 23M 0S, 20H 3...
## $ payment                 <chr> "Ewallet", "Cash", "Credit card", "Ewallet"...
```

```
## $ cogs                    <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.income            <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ rating                  <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ total                   <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634....
## $ month                   <dbl> 1, 3, 3, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2, 2, 3...
## $ day                     <int> 5, 8, 3, 27, 8, 25, 25, 24, 10, 20, 6, 9, 1...
## $ year                    <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2...
## $ hour                    <dbl> 13, 10, 13, 20, 10, 18, 14, 11, 17, 13, 18,...
## $ minute                  <dbl> 8, 29, 23, 33, 37, 30, 36, 38, 15, 27, 7, 3...
```

```r
# converting day and month, hour and minute to factor
sales$month <- as.factor(sales$month)
sales$day <- as.factor(sales$day)
sales$hour <- as.factor(sales$hour)
sales$minute <- as.factor(sales$minute)
```

4. Cleaning the data

4.1 Fixing column names and data types

```r
# changing some column names so that they make more sense
names(sales)[12] <- "cost.of.goods.sold"
names(sales)[16] <- "total.sales.plus.tax"
names(sales)[14] <- "gross.profit"
glimpse(sales)
```

```
## Rows: 1,000
## Columns: 21
## $ 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", "No...
## $ gender                  <chr> "Female", "Female", "Male", "Male", "Male",...
## $ product.line            <chr> "Health and beauty", "Electronic accessorie...
## $ unit.price              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ quantity                <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ tax                     <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ date                    <date> 2019-01-05, 2019-03-08, 2019-03-03, 2019-0...
## $ time                    <Period> 13H 8M 0S, 10H 29M 0S, 13H 23M 0S, 20H 3...
## $ payment                 <chr> "Ewallet", "Cash", "Credit card", "Ewallet"...
## $ cost.of.goods.sold      <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.profit            <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ rating                  <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ total.sales.plus.tax    <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634....
## $ month                   <fct> 1, 3, 3, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2, 2, 3...
## $ day                     <fct> 5, 8, 3, 27, 8, 25, 25, 24, 10, 20, 6, 9, 1...
## $ year                    <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2...
## $ hour                    <fct> 13, 10, 13, 20, 10, 18, 14, 11, 17, 13, 18,...
## $ minute                  <fct> 8, 29, 23, 33, 37, 30, 36, 38, 15, 27, 7, 3...
```

```r
# converting quantity to character datatype
sales$quantity <- as.character(sales$quantity)
```

4.2 Missing values

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

```
##              invoice.id                  branch           customer.type
##                       0                       0                       0
##                  gender            product.line              unit.price
##                       0                       0                       0
##                quantity                     tax                    date
##                       0                       0                       0
##                    time                 payment      cost.of.goods.sold
##                       0                       0                       0
## gross.margin.percentage            gross.profit                  rating
##                       0                       0                       0
##    total.sales.plus.tax                   month                     day
##                       0                       0                       0
##                    year                    hour                  minute
##                       0                       0                       0
```

The data has no missing values

4.3 Duplicated

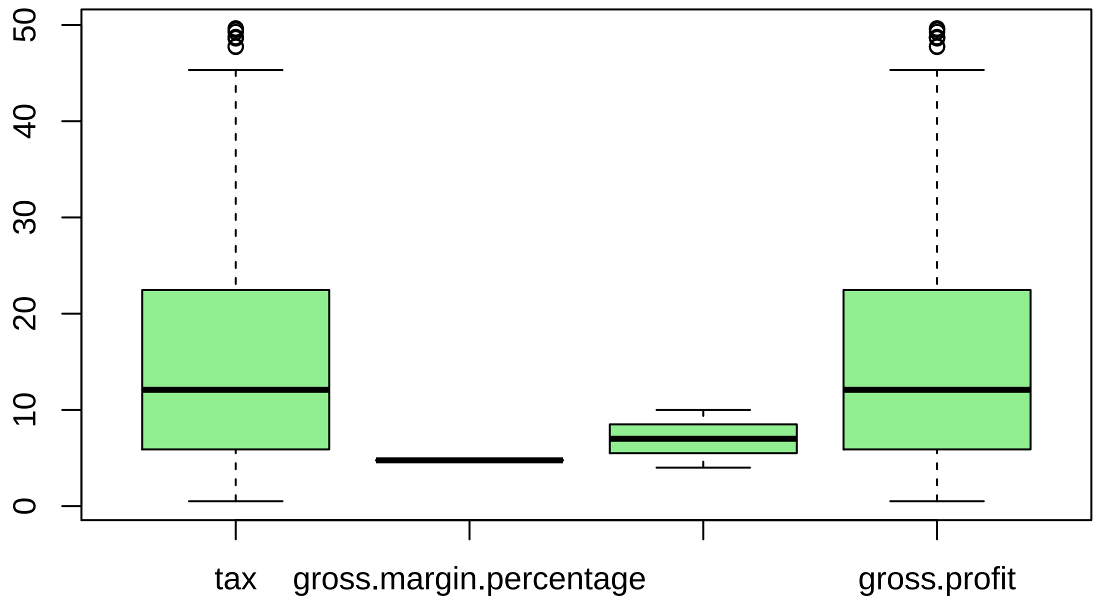
```
dup <- sales[duplicated(sales),]
dup
```

```
##  [1] invoice.id              branch                  customer.type
##  [4] gender                  product.line            unit.price
##  [7] quantity                tax                     date
## [10] time                    payment                 cost.of.goods.sold
## [13] gross.margin.percentage gross.profit            rating
## [16] total.sales.plus.tax    month                   day
## [19] year                    hour                    minute
## <0 rows> (or 0-length row.names)
```

There are no duplicated records in our dataset

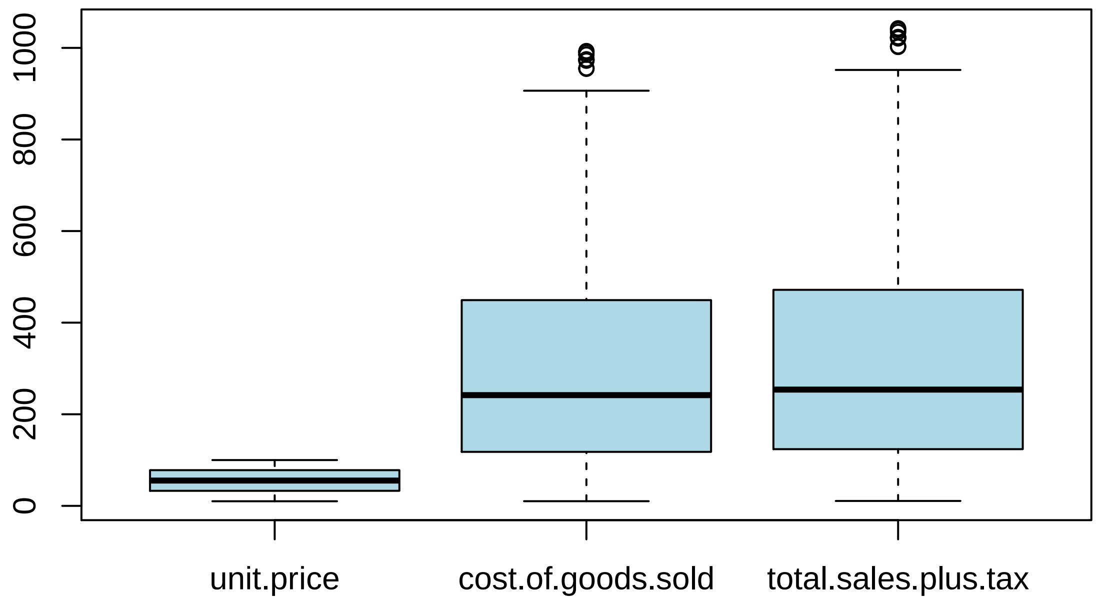
4.4 Outliers

```
boxplot(select(sales, tax, gross.margin.percentage, rating, gross.profit), col = "light green")
```



The data has a few outliers in the tax and gross income columns. We wont be deleting these outliers because we establish that they are true observations and it's possible for some products to be taxed very highly.

```
# more boxplot outliers
boxplot(select(sales, unit.price, cost.of.goods.sold, total.sales.plus.tax), col = "light blue")
```



There are some outliers in cost of goods sold and total.sales.plus tax columns. This observation is in line with the tax outlier seen earlier. We won't be deleting these outliers because we establish that they are true observations.

4.5 Dropping unnecessary columns

```
# deleting unnecessary column
sales$date <- NULL
sales$year <- NULL
sales$time <- NULL
sales$invoice.id <- NULL
```

5. EDA

5.1 Univariate Analysis

```
# we will start with summary stats for numerical columns
num_col <- select(sales,unit.price, tax, cost.of.goods.sold, gross.profit, rating, total.sales.plus.tax
summary(num_col)
```

```
##    unit.price         tax          cost.of.goods.sold  gross.profit
##  Min.   :10.08   Min.   : 0.5085   Min.   : 10.17     Min.   : 0.5085
##  1st Qu.:32.88   1st Qu.: 5.9249   1st Qu.:118.50     1st Qu.: 5.9249
##  Median :55.23   Median :12.0880   Median :241.76     Median :12.0880
##  Mean   :55.67   Mean   :15.3794   Mean   :307.59     Mean   :15.3794
##  3rd Qu.:77.94   3rd Qu.:22.4453   3rd Qu.:448.90     3rd Qu.:22.4453
```

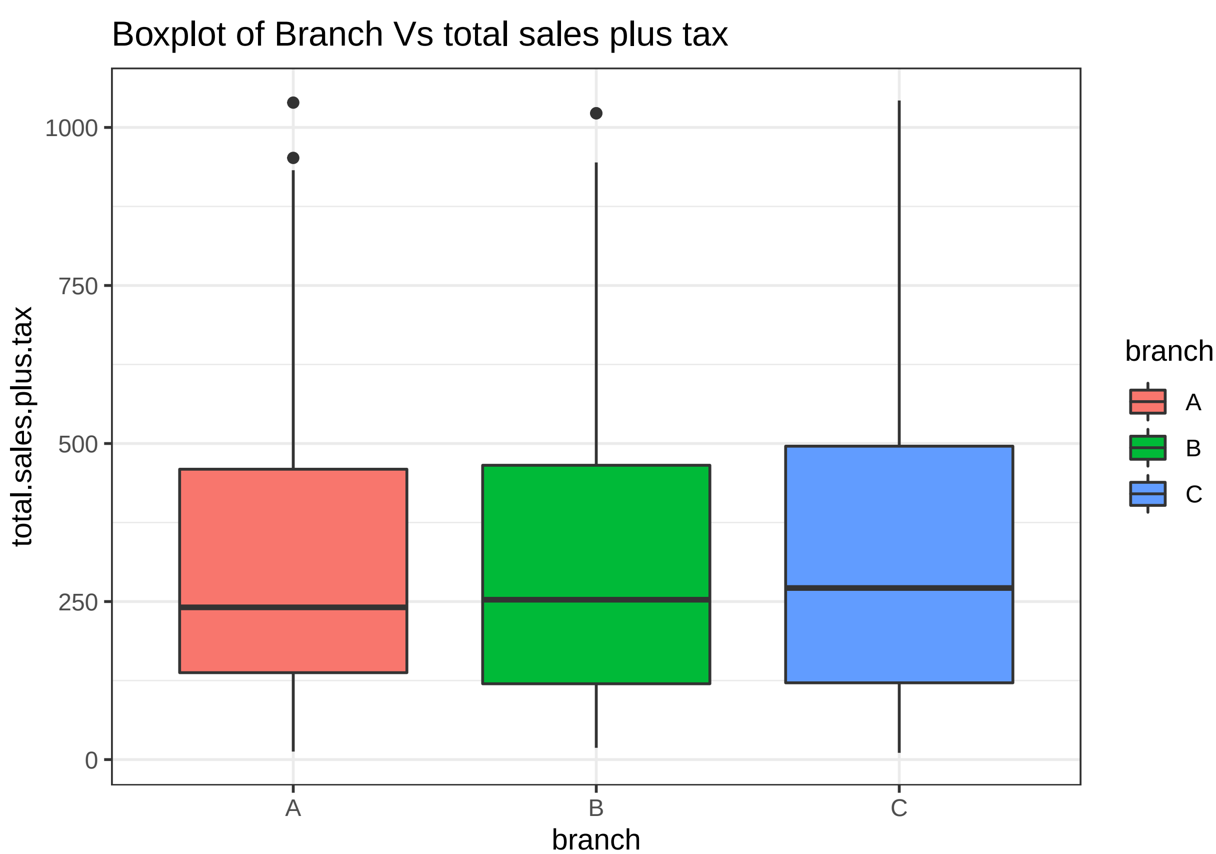
```
##  Max.   :99.96   Max.   :49.6500   Max.   :993.00     Max.   :49.6500
##      rating       total.sales.plus.tax
##  Min.   : 4.000   Min.   :  10.68
##  1st Qu.: 5.500   1st Qu.: 124.42
##  Median : 7.000   Median : 253.85
##  Mean   : 6.973   Mean   : 322.97
##  3rd Qu.: 8.500   3rd Qu.: 471.35
##  Max.   :10.000   Max.   :1042.65
```

Unit price mean = 55.67 The average quantity of goods sold = 5.51 The mean tax = 15.37 The average cost of goods sold = 307.59 Gross margin is constant at 4.762 Gross profit = 15.37 The average rating is 6.97 The average total sales plus tax = 322.97

5.2 Bivariate Analysis

5.2.1 Boxplots for comparing the target variable with categorical variables

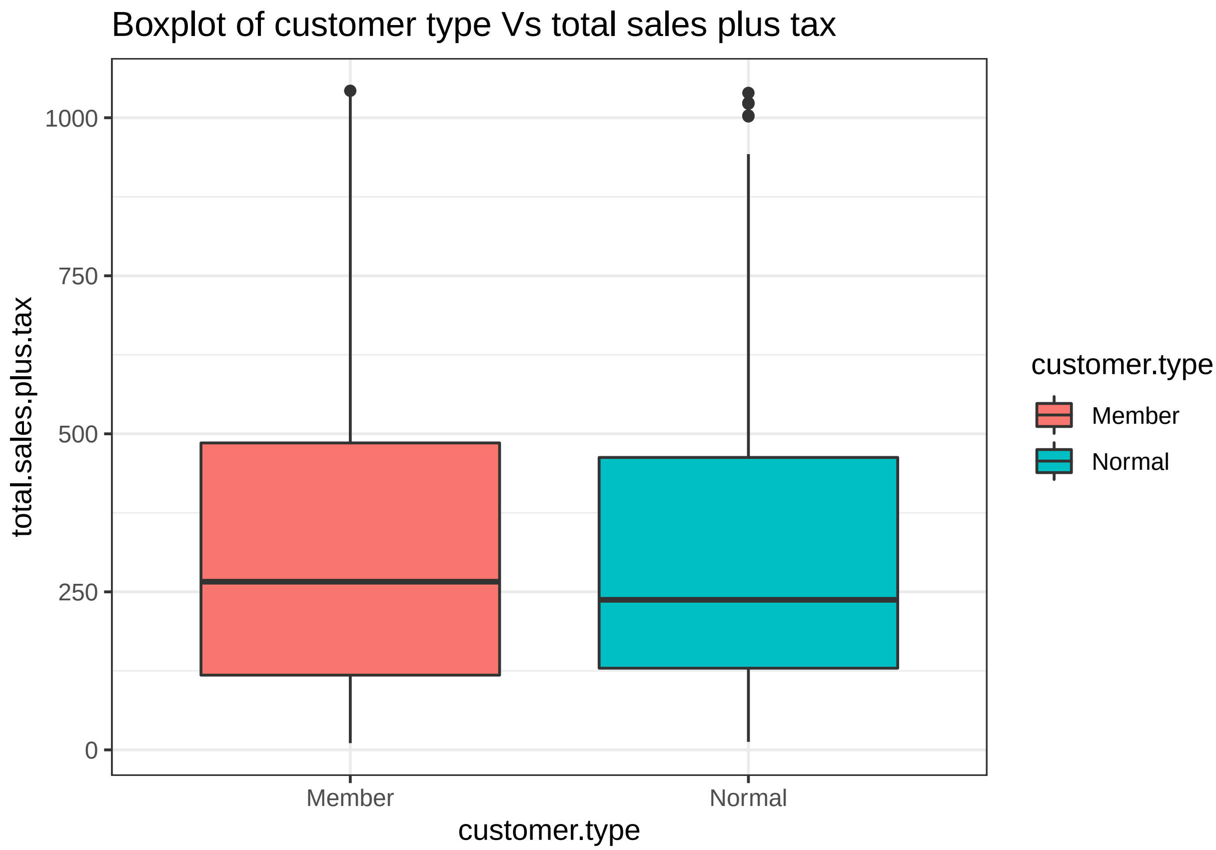
```
# Our target variable is total.sales.plus.tax. Let's do some analysis to see how that compares with oth

ggplot(sales, aes(x=branch, y=total.sales.plus.tax, fill = branch)) +
  theme_bw() +
  geom_boxplot() +
  labs(title = "Boxplot of Branch Vs total sales plus tax")
```



We have a few outliers in branch A and B but none in C. Branch C is bringing in slightly more money.

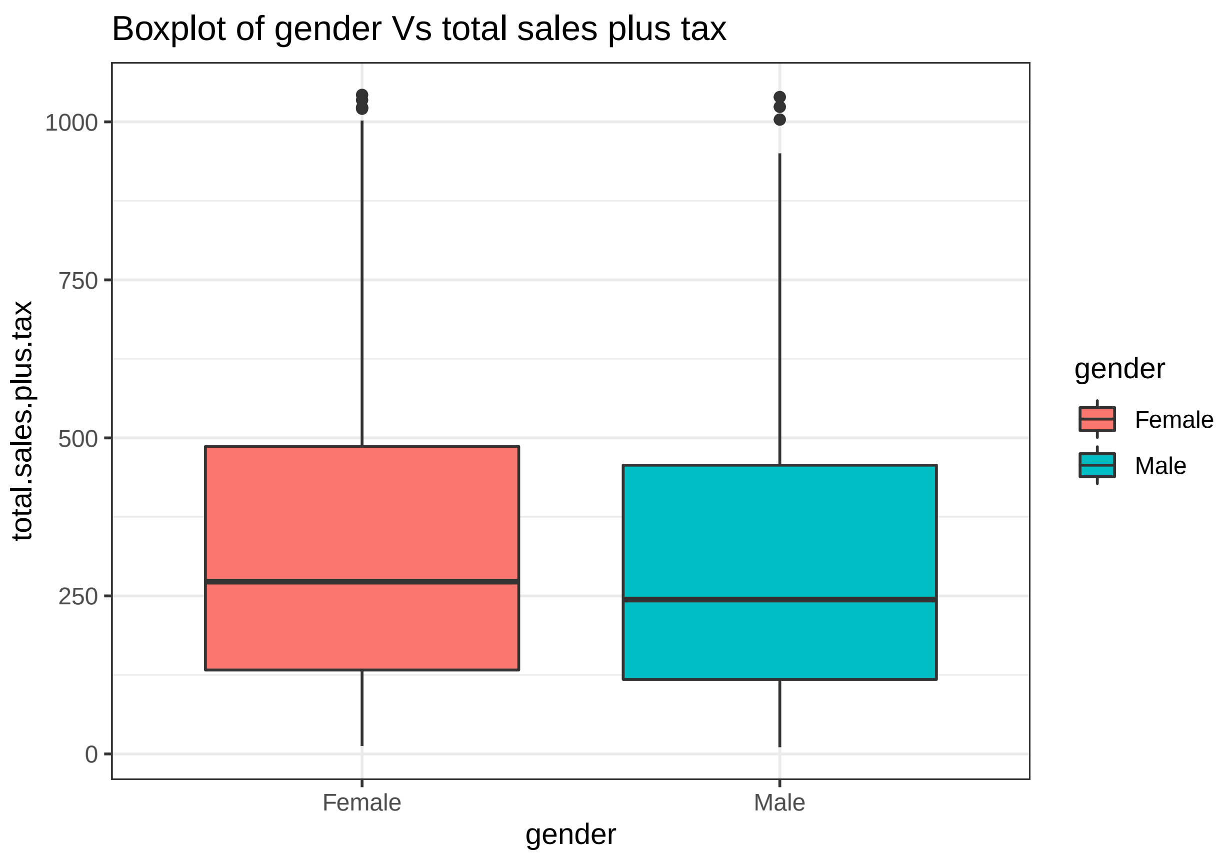
```
ggplot(sales, aes(x=customer.type, y=total.sales.plus.tax, fill = customer.type)) +
  theme_bw() +
```

```r
  geom_boxplot() +
  labs(title = "Boxplot of customer type Vs total sales plus tax")
```



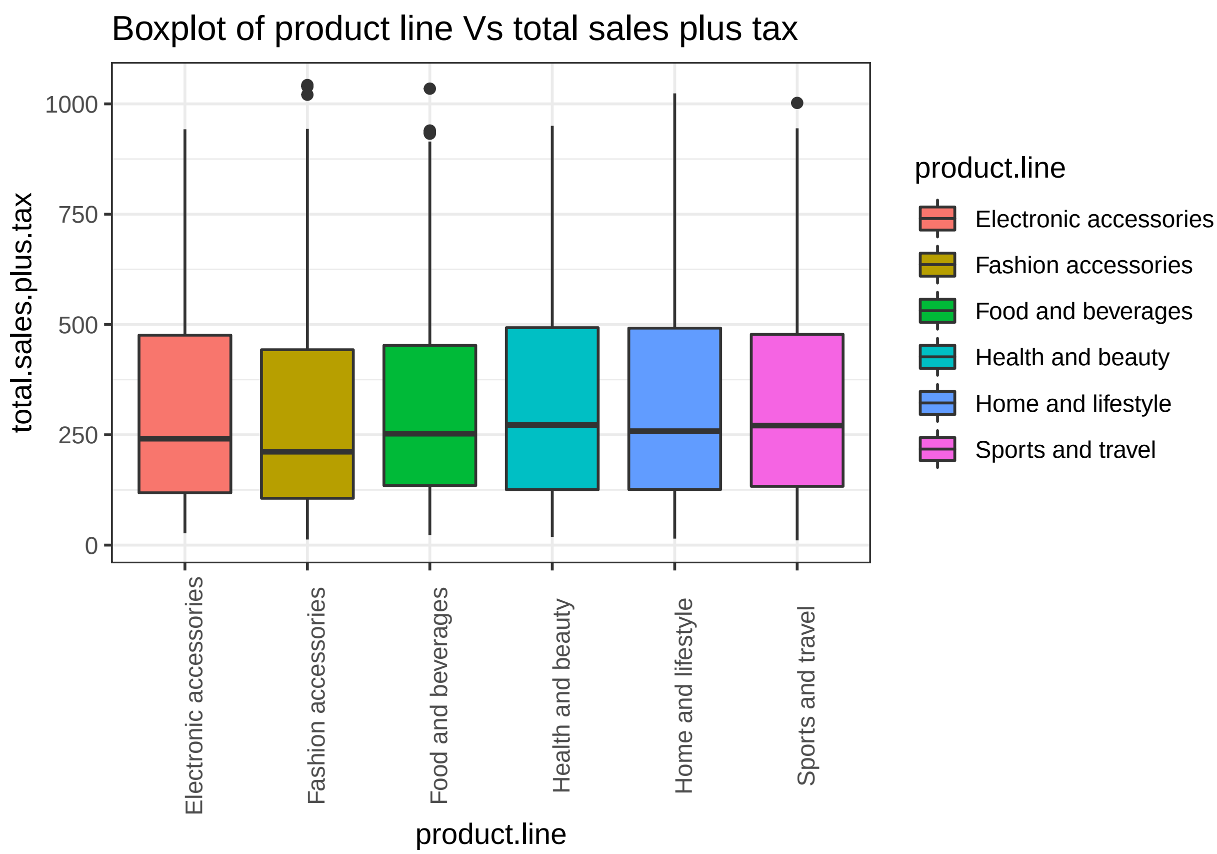
The members customer type on average bring in more revenue through sales.

```r
ggplot(sales, aes(x=gender, y=total.sales.plus.tax, fill = gender)) +
  theme_bw() +
  geom_boxplot() +
  labs(title = "Boxplot of gender Vs total sales plus tax")
```



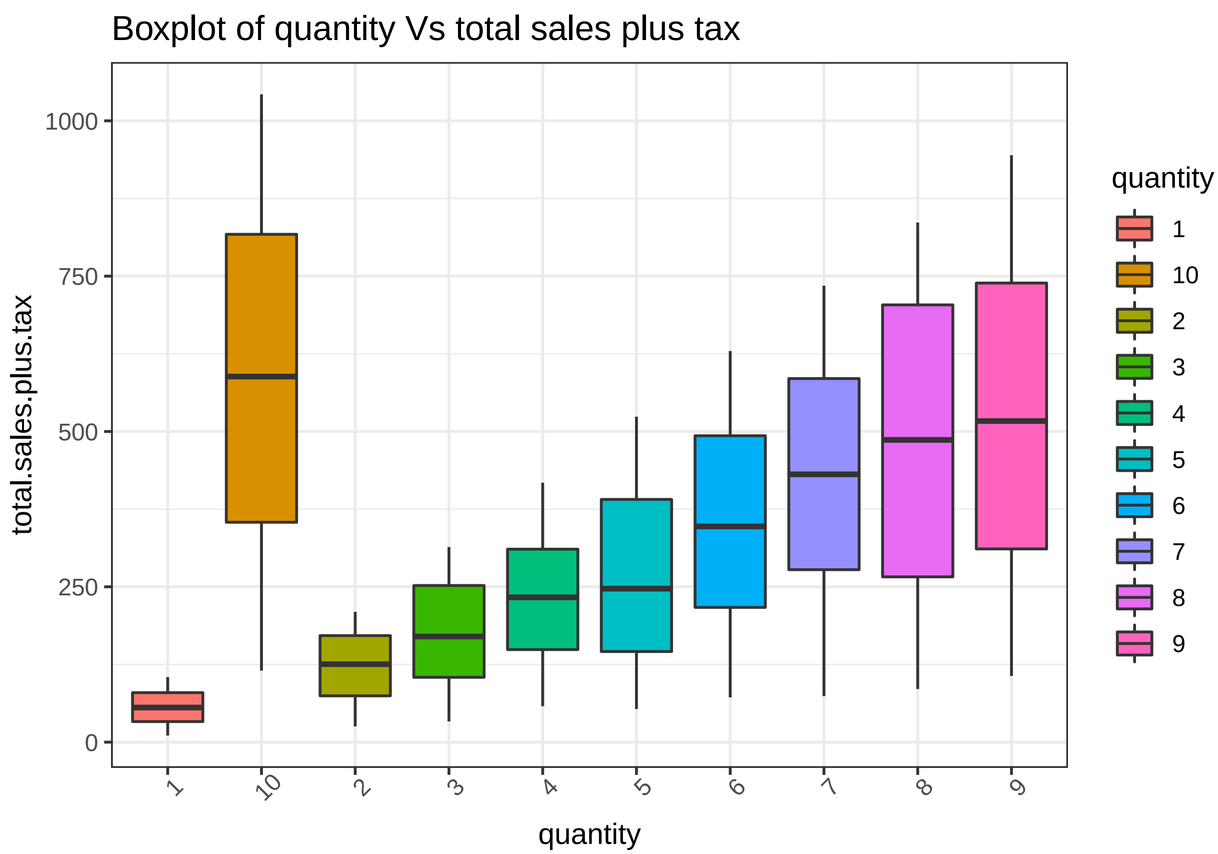
Total sales from women is a bit more compared to that of men.

```
ggplot(sales, aes(x=product.line, y=total.sales.plus.tax, fill = product.line)) +
  theme_bw() +
  geom_boxplot() +
  labs(title = "Boxplot of product line Vs total sales plus tax") +
  theme(axis.text.x = element_text(angle = 90))
```



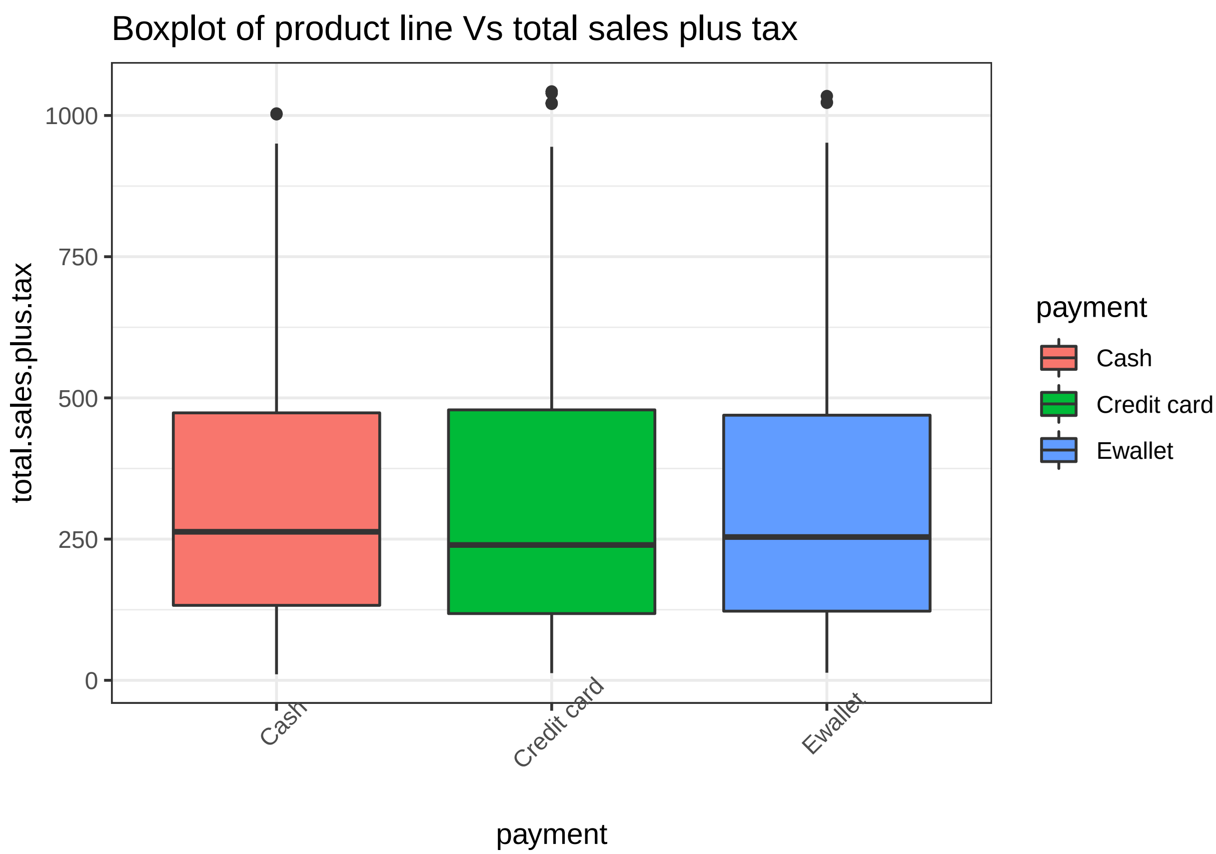
Based on the boxplot, on average, the product line bringing in the most income is health and beauty, and home and lifestyle.

```
ggplot(sales, aes(x=quantity, y=total.sales.plus.tax, fill = quantity)) +
  theme_bw() +
  geom_boxplot() +
  labs(title = "Boxplot of quantity Vs total sales plus tax") +
  theme(axis.text.x = element_text(angle = 45))
```



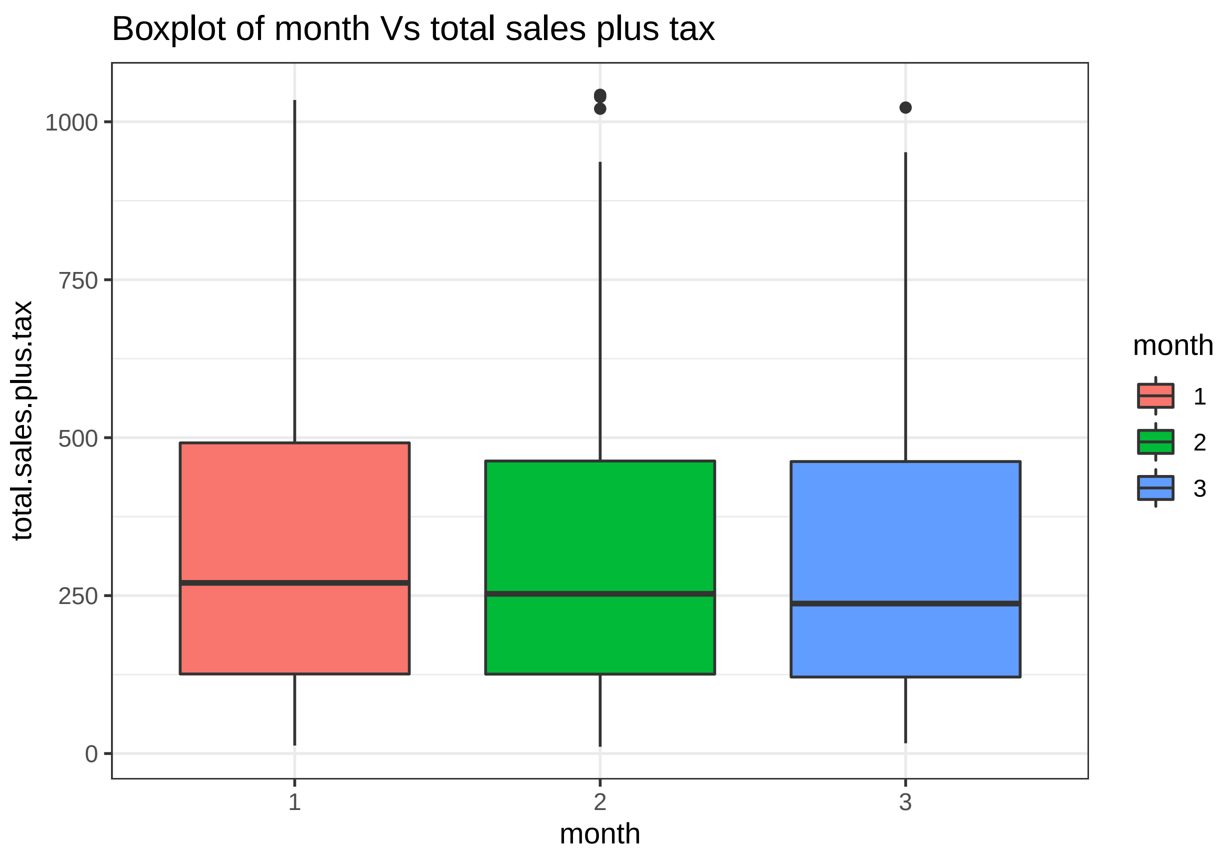
Most number of items bought is 10 followed by 9, 8, 7 all the way to 1.

```
ggplot(sales, aes(x=payment, y=total.sales.plus.tax, fill = payment)) +
  theme_bw() +
  geom_boxplot() +
  labs(title = "Boxplot of product line Vs total sales plus tax") +
  theme(axis.text.x = element_text(angle = 45))
```



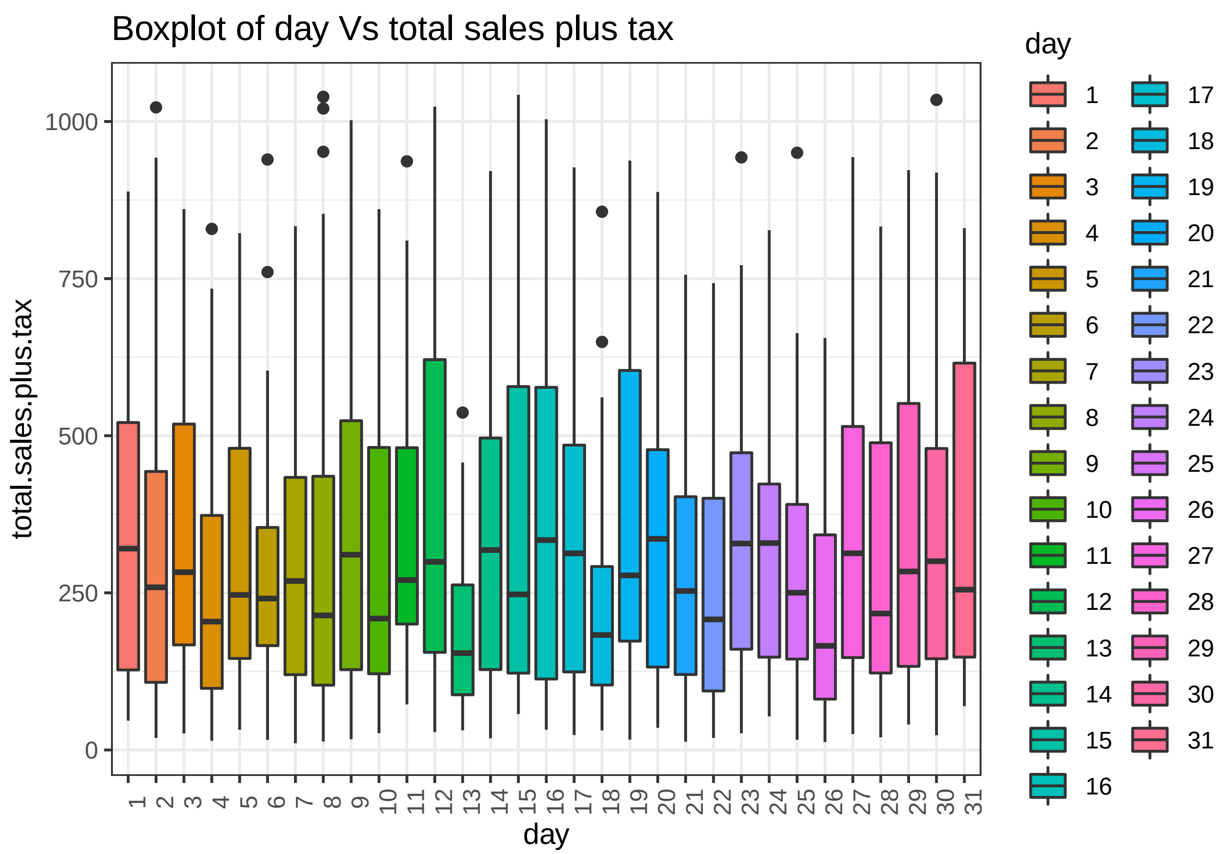
On average, cash is the main mode of payment.

```
ggplot(sales, aes(x=month, y=total.sales.plus.tax, fill = month)) +
  theme_bw() +
  geom_boxplot() +
  labs(title = "Boxplot of month Vs total sales plus tax") +
  theme(axis.text.x = element_text(angle = 0))
```



On average, January brought in more total sales plus taxes.

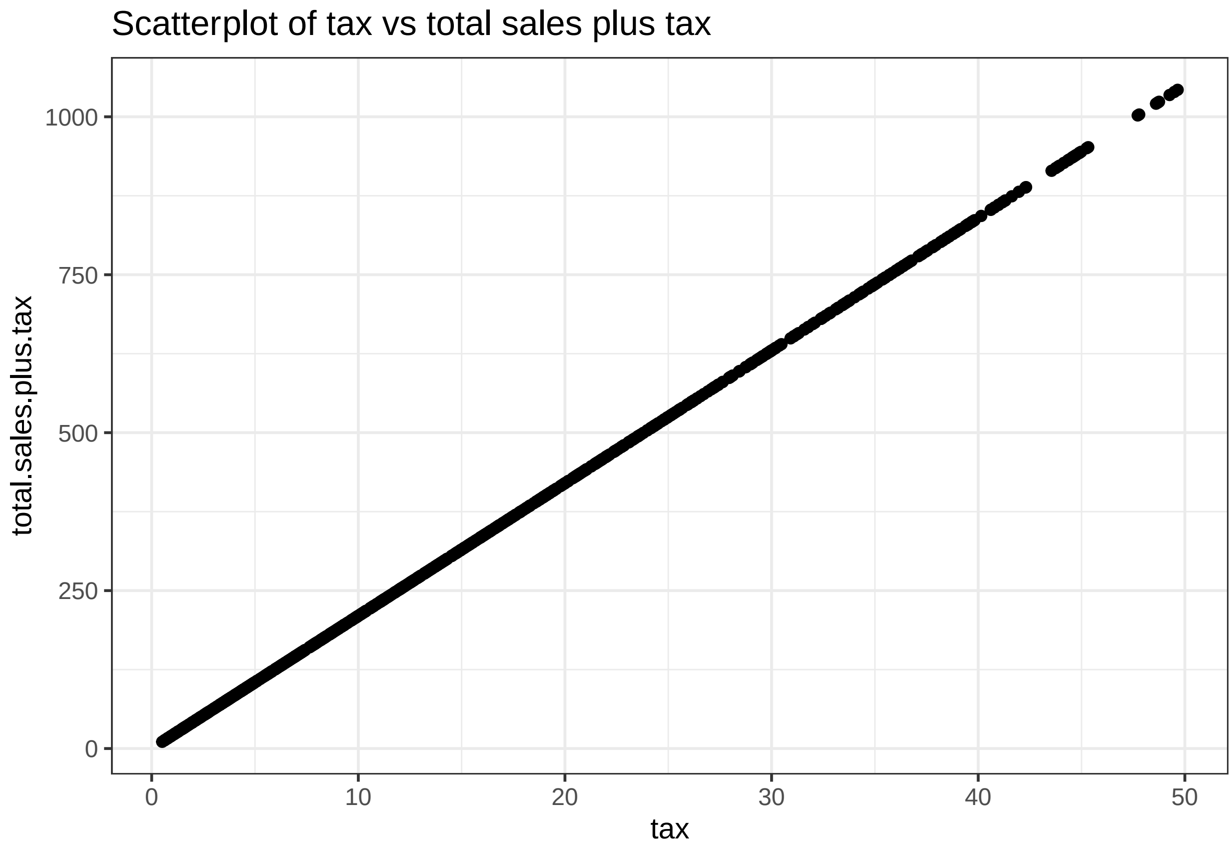
```
ggplot(sales, aes(x=day, y=total.sales.plus.tax, fill = day)) +
  theme_bw() +
  geom_boxplot() +
  labs(title = "Boxplot of day Vs total sales plus tax") +
  theme(axis.text.x = element_text(angle = 90))
```



5.2.2 Scatter plots to determine the relationship between target variable and numerical variables

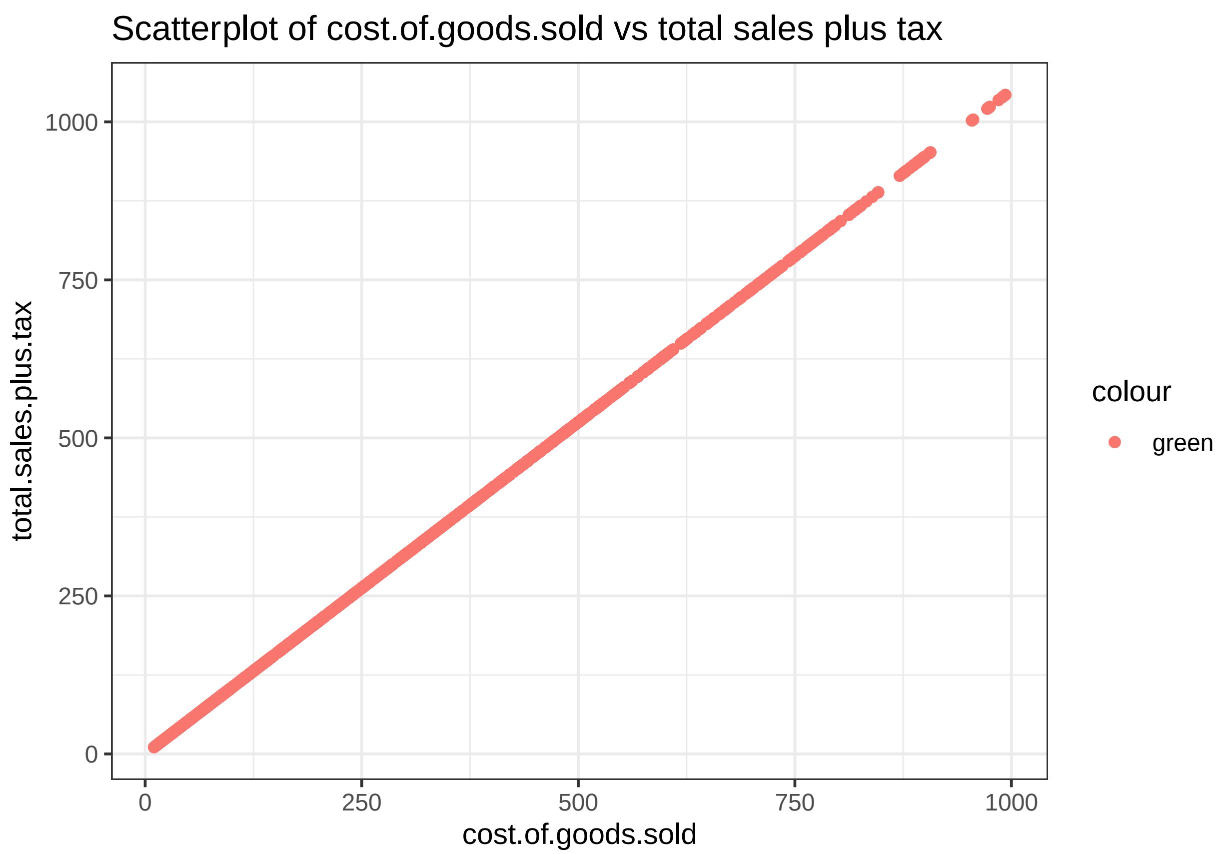
```
ggplot(sales, aes(x = unit.price, y = total.sales.plus.tax)) +
  geom_point(aes(color = "green")) +
  theme_bw() +
  labs(title = "Scatterplot of unit price vs total sales plus tax")
```


```
ggplot(sales, aes(x = tax, y = total.sales.plus.tax)) +
  geom_point() +
  theme_bw() +
  labs(title = "Scatterplot of tax vs total sales plus tax")
```



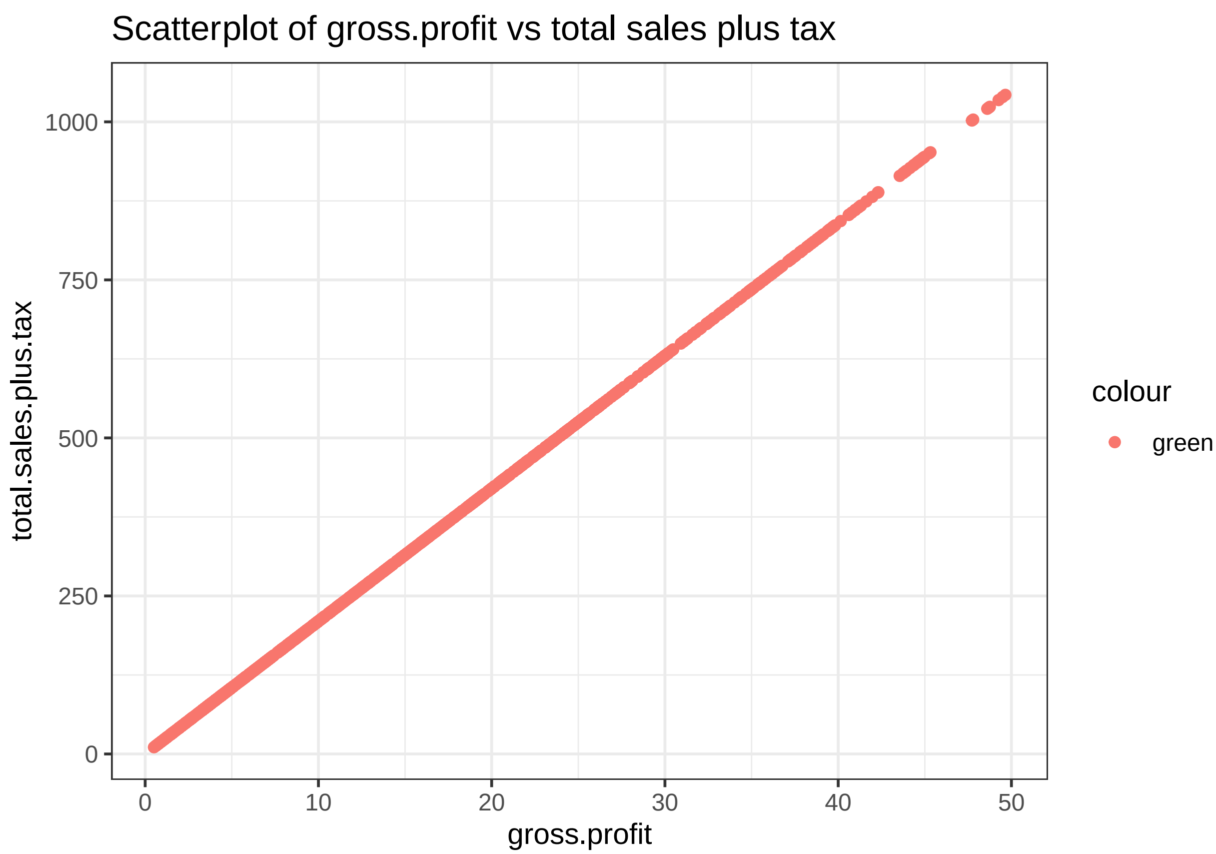
Tax vs sales shows a very strong positive relationship.

```
ggplot(sales, aes(x = cost.of.goods.sold, y = total.sales.plus.tax)) +
  geom_point(aes(color = "green")) +
  theme_bw() +
  labs(title = "Scatterplot of cost.of.goods.sold vs total sales plus tax")
```



Another very strong relationship between cost of goods sold and total sales.

```
ggplot(sales, aes(x = gross.profit, y = total.sales.plus.tax)) +
  geom_point(aes(color = "green")) +
  theme_bw() +
  labs(title = "Scatterplot of gross.profit vs total sales plus tax")
```



Same with the two plots above. Gross profit vs total sales shows that their is a strong positive relationship.

5.3 Data preprocessing

```r
# encoding categorical variables
sales$branch <- factor(sales$branch, levels = c("A", "B", "C"), labels = c(1,2,3))
sales$gender <- factor(sales$gender, levels = c("Female", "Male"), labels = c(1,0))
sales$customer.type <- factor(sales$customer.type, levels = c("Member", "Normal"), labels = c(1,2))
sales$product.line <- factor(sales$product.line, levels =c("Health and beauty", "Electronic accessories
"Food and beverages", "Fashion accessories"), labels = c(1,2,3,4,5,6))
sales$quantity <- as.factor(sales$quantity)
sales$payment <- factor(sales$payment, levels = c("Ewallet", "Cash", "Credit card"), labels = c(1,2,3))
glimpse(sales)
```

```
## Rows: 1,000
## Columns: 17
## $ branch                  <fct> 1, 3, 1, 1, 1, 3, 1, 3, 1, 2, 2, 2, 1, 1, 1...
## $ customer.type           <fct> 1, 2, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2...
## $ gender                  <fct> 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1...
## $ product.line            <fct> 1, 2, 3, 1, 4, 2, 2, 3, 1, 5, 6, 2, 2, 5, 1...
## $ unit.price              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ quantity                <fct> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ tax                     <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ payment                 <fct> 1, 2, 3, 1, 1, 1, 1, 1, 3, 3, 1, 2, 1, 1, 2...
## $ cost.of.goods.sold      <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
```

```
## $ gross.profit           <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ rating                 <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ total.sales.plus.tax   <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634....
## $ month                  <fct> 1, 3, 3, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2, 2, 3...
## $ day                    <fct> 5, 8, 3, 27, 8, 25, 25, 24, 10, 20, 6, 9, 1...
## $ hour                   <fct> 13, 10, 13, 20, 10, 18, 14, 11, 17, 13, 18,...
## $ minute                 <fct> 8, 29, 23, 33, 37, 30, 36, 38, 15, 27, 7, 3...
```

Great! All the categorical variables have been encoded.

6. Implementing the Solution

6.1 Principal Component Analysis

```r
glimpse(sales)
```

```
## Rows: 1,000
## Columns: 17
## $ branch                 <fct> 1, 3, 1, 1, 1, 3, 1, 3, 1, 2, 2, 2, 1, 1, 1...
## $ customer.type          <fct> 1, 2, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2...
## $ gender                 <fct> 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1...
## $ product.line           <fct> 1, 2, 3, 1, 4, 2, 2, 3, 1, 5, 6, 2, 2, 5, 1...
## $ unit.price             <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ quantity               <fct> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ tax                    <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ payment                <fct> 1, 2, 3, 1, 1, 1, 1, 1, 3, 3, 1, 2, 1, 1, 2...
## $ cost.of.goods.sold     <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.profit           <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ rating                 <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ total.sales.plus.tax   <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634....
## $ month                  <fct> 1, 3, 3, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2, 2, 3...
## $ day                    <fct> 5, 8, 3, 27, 8, 25, 25, 24, 10, 20, 6, 9, 1...
## $ hour                   <fct> 13, 10, 13, 20, 10, 18, 14, 11, 17, 13, 18,...
## $ minute                 <fct> 8, 29, 23, 33, 37, 30, 36, 38, 15, 27, 7, 3...
```

```r
# since it's unsupervised learning, we will drop the target variable(total.sales.plus.tax) before apply
# first we create a copy of our cleaned dataset
sales.copy <- sales
sales.df <- sales.copy[,c(1:12, 14:17)] # selecting all columns minus the target variable
glimpse(sales.df)
```

```
## Rows: 1,000
## Columns: 16
## $ branch                 <fct> 1, 3, 1, 1, 1, 3, 1, 3, 1, 2, 2, 2, 1, 1, 1...
## $ customer.type          <fct> 1, 2, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2...
## $ gender                 <fct> 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1...
## $ product.line           <fct> 1, 2, 3, 1, 4, 2, 2, 3, 1, 5, 6, 2, 2, 5, 1...
## $ unit.price             <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ quantity               <fct> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5, 10,...
## $ tax                    <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ payment                <fct> 1, 2, 3, 1, 1, 1, 1, 1, 3, 3, 1, 2, 1, 1, 2...
## $ cost.of.goods.sold     <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.profit           <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
```

```
## $ rating                 <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ month                  <fct> 1, 3, 3, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2, 2, 3...
## $ day                    <fct> 5, 8, 3, 27, 8, 25, 25, 24, 10, 20, 6, 9, 1...
## $ hour                   <fct> 13, 10, 13, 20, 10, 18, 14, 11, 17, 13, 18,...
## $ minute                 <fct> 8, 29, 23, 33, 37, 30, 36, 38, 15, 27, 7, 3...
```

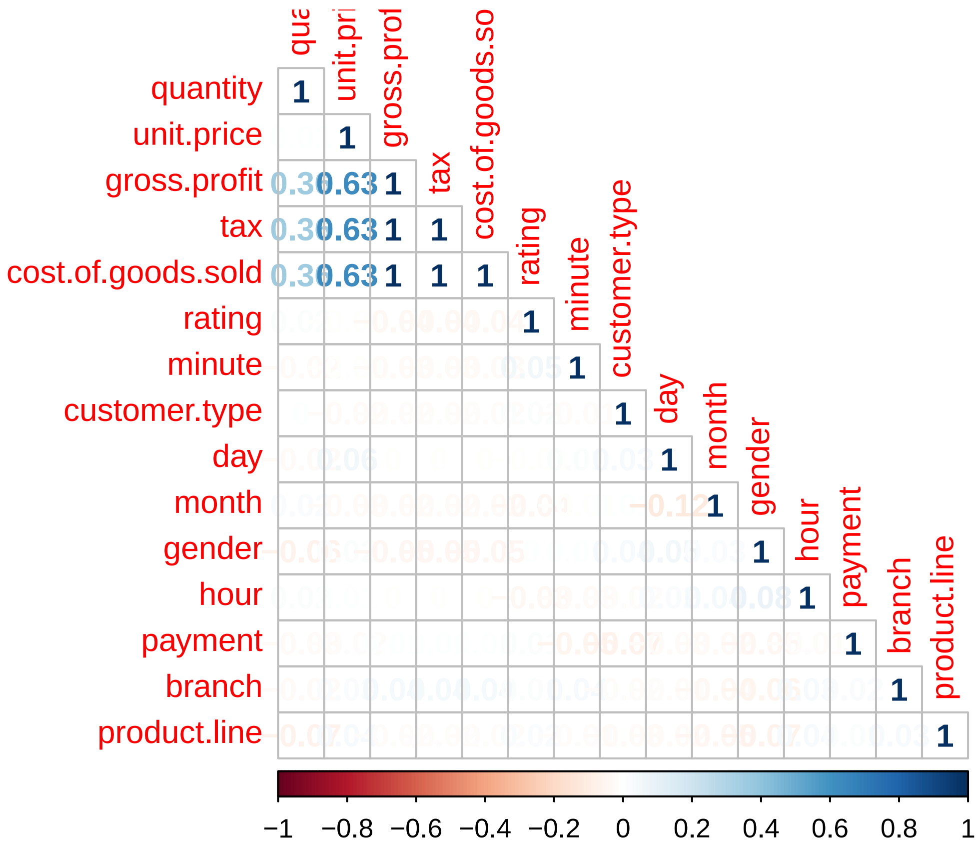
```r
# converting the factors to numeric

sales.df$branch <- as.numeric(sales.df$branch)
sales.df$customer.type <- as.numeric(sales.df$customer.type)
sales.df$gender <- as.numeric(sales.df$gender)
sales.df$product.line <- as.numeric(sales.df$product.line)
sales.df$quantity <- as.numeric(sales.df$quantity)
sales.df$payment <- as.numeric(sales.df$payment)
sales.df$month <- as.numeric(sales.df$month)
sales.df$day <- as.numeric(sales.df$day)
sales.df$hour <- as.numeric(sales.df$hour)
sales.df$minute <- as.numeric(sales.df$minute)
glimpse(sales.df)
```

```
## Rows: 1,000
## Columns: 16
## $ branch                 <dbl> 1, 3, 1, 1, 1, 3, 1, 3, 1, 2, 2, 2, 1, 1, 1...
## $ customer.type          <dbl> 1, 2, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2...
## $ gender                 <dbl> 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1...
## $ product.line           <dbl> 1, 2, 3, 1, 4, 2, 2, 3, 1, 5, 6, 2, 2, 5, 1...
## $ unit.price             <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 6...
## $ quantity               <dbl> 8, 6, 8, 9, 8, 8, 7, 2, 3, 4, 5, 5, 6, 2, 2...
## $ tax                    <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ payment                <dbl> 1, 2, 3, 1, 1, 1, 1, 1, 3, 3, 1, 2, 1, 1, 2...
## $ cost.of.goods.sold     <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597....
## $ gross.margin.percentage <dbl> 4.761905, 4.761905, 4.761905, 4.761905, 4.7...
## $ gross.profit           <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085,...
## $ rating                 <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2...
## $ month                  <dbl> 1, 3, 3, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2, 2, 3...
## $ day                    <dbl> 5, 8, 3, 27, 8, 25, 25, 24, 10, 20, 6, 9, 1...
## $ hour                   <dbl> 4, 1, 4, 11, 1, 9, 5, 2, 8, 4, 9, 8, 1, 7, ...
## $ minute                 <dbl> 9, 30, 24, 34, 38, 31, 37, 39, 16, 28, 8, 4...
```

```r
# let's make a copy of the cleaned df before deleting gross.margin.percentage column that we won't need

sales.cleaned <- sales.df
```

```r
sales.df$gross.margin.percentage <- NULL
```

```r
corr <- cor(sales.df, method = "pearson")
corrplot(corr, method = "number", type = "lower", order = "hclust")
```




```r
# applying PCA to  sales.df
salesdf.pca <- prcomp(sales.df, center = TRUE, scale. = TRUE)
salesdf.pca
```

```
## Standard deviations (1, .., p=15):
##  [1] 1.902352e+00 1.094060e+00 1.083265e+00 1.051402e+00 1.028494e+00
##  [6] 1.002271e+00 9.964909e-01 9.947646e-01 9.759575e-01 9.362636e-01
## [11] 9.200540e-01 8.918542e-01 6.239360e-01 4.267481e-16 1.755044e-16
##
## Rotation (n x k) = (15 x 15):
##                             PC1          PC2          PC3          PC4
## branch              0.0273694066  0.340928572 -0.120950985  0.128779137
## customer.type      -0.0149226201 -0.339573569 -0.135179876 -0.275012233
## gender             -0.0330190817 -0.520647166 -0.233365004  0.255126077
## product.line       -0.0097333598  0.406034897 -0.175218710  0.300233001
## unit.price          0.3779711874  0.010913138 -0.235974297  0.168691387
## quantity            0.2143920663 -0.119904309  0.334139506 -0.310431607
## tax                 0.5189385679 -0.004432186  0.008128015 -0.002731329
## payment             0.0015364151  0.397636738  0.146012221  0.140745029
## cost.of.goods.sold  0.5189385679 -0.004432186  0.008128015 -0.002731329
## gross.profit        0.5189385679 -0.004432186  0.008128015 -0.002731329
## rating             -0.0217865729  0.138902504 -0.163323789 -0.379826387
## month              -0.0159150267 -0.275084821  0.527648990  0.187287922
## day                 0.0045260022 -0.162624631 -0.585795727 -0.024102676
## hour                0.0002467417 -0.195292029 -0.029978840  0.596116668
## minute             -0.0193518844  0.017272265 -0.209623749 -0.269477372
```

```
##                            PC5          PC6         PC7          PC8
## branch             -0.397236394 -0.017279848 -0.42684622  0.286116629
## customer.type       0.006899145  0.580065776 -0.04654431 -0.054552999
## gender             -0.064051105 -0.272562913  0.14097483 -0.295260950
## product.line       -0.157106801  0.538227869  0.18433526 -0.192602771
## unit.price         -0.039816314 -0.002339985  0.30282385 -0.008221295
## quantity            0.018999580  0.029137798 -0.53135481 -0.084137073
## tax                -0.003919154 -0.004556045  0.01486185 -0.007690004
## payment             0.366804053 -0.386908742 -0.01204285 -0.300044754
## cost.of.goods.sold -0.003919154 -0.004556045  0.01486185 -0.007690004
## gross.profit       -0.003919154 -0.004556045  0.01486185 -0.007690004
## rating             -0.273090681 -0.056250114 -0.06600763 -0.761636698
## month              -0.297676208  0.003974980  0.23180156 -0.007338774
## day                 0.308497395 -0.113259870 -0.23821663  0.151258614
## hour               -0.213942348  0.017942417 -0.50162094 -0.217054741
## minute             -0.610492692 -0.363731471  0.14309078  0.208324847
##                           PC9        PC10        PC11          PC12
## branch              0.53796619  0.28031075 -0.06239925  0.2389807211
## customer.type       0.52198787 -0.38706594  0.13930244 -0.0842397197
## gender              0.15689124  0.21700636  0.37036958  0.4618390344
## product.line       -0.30807353 -0.18077969  0.08416226  0.4396118976
## unit.price          0.10044691  0.04868469 -0.13012445 -0.1865000405
## quantity           -0.28159550 -0.07932242  0.14726300  0.3874975218
## tax                 0.01119192 -0.01034829  0.01567097 -0.0005151875
## payment             0.37819619 -0.50802066  0.09694308  0.1142223168
## cost.of.goods.sold  0.01119192 -0.01034829  0.01567097 -0.0005151875
## gross.profit        0.01119192 -0.01034829  0.01567097 -0.0005151875
## rating              0.03386215  0.18970845 -0.30834589 -0.1116389104
## month               0.13482309 -0.18831694 -0.59152925  0.2535667449
## day                -0.13009186 -0.22483767 -0.54820278  0.2658235267
## hour               -0.15140907 -0.23102958  0.02263424 -0.4285155469
## minute             -0.16318895 -0.49938469  0.19408286 -0.0143798523
##                             PC13          PC14          PC15
## branch              0.0257384233 -2.382931e-17  4.525329e-18
## customer.type       0.0101291399 -6.448407e-17 -1.255668e-16
## gender             -0.0258103131 -8.810457e-17  2.539151e-17
## product.line       -0.0286333690 -3.103373e-17 -1.155330e-17
## unit.price          0.7844379686 -2.997054e-16  1.041429e-16
## quantity            0.4284696429 -2.615067e-16  9.632691e-17
## tax                -0.2512938678 -8.150962e-01  4.780050e-02
## payment             0.0526916416 -1.167458e-16  1.062647e-16
## cost.of.goods.sold -0.2512938678  4.489445e-01  6.819937e-01
## gross.profit       -0.2512938678  3.661516e-01 -7.297942e-01
## rating             -0.0492164258  3.800031e-17  3.665228e-17
## month              -0.0181274544  6.084707e-17  9.785773e-17
## day                -0.0592122830  1.307555e-16  1.233559e-16
## hour               -0.0184473676  4.306750e-17 -2.027034e-17
## minute             -0.0004861354 -1.445673e-17  2.158154e-17
```

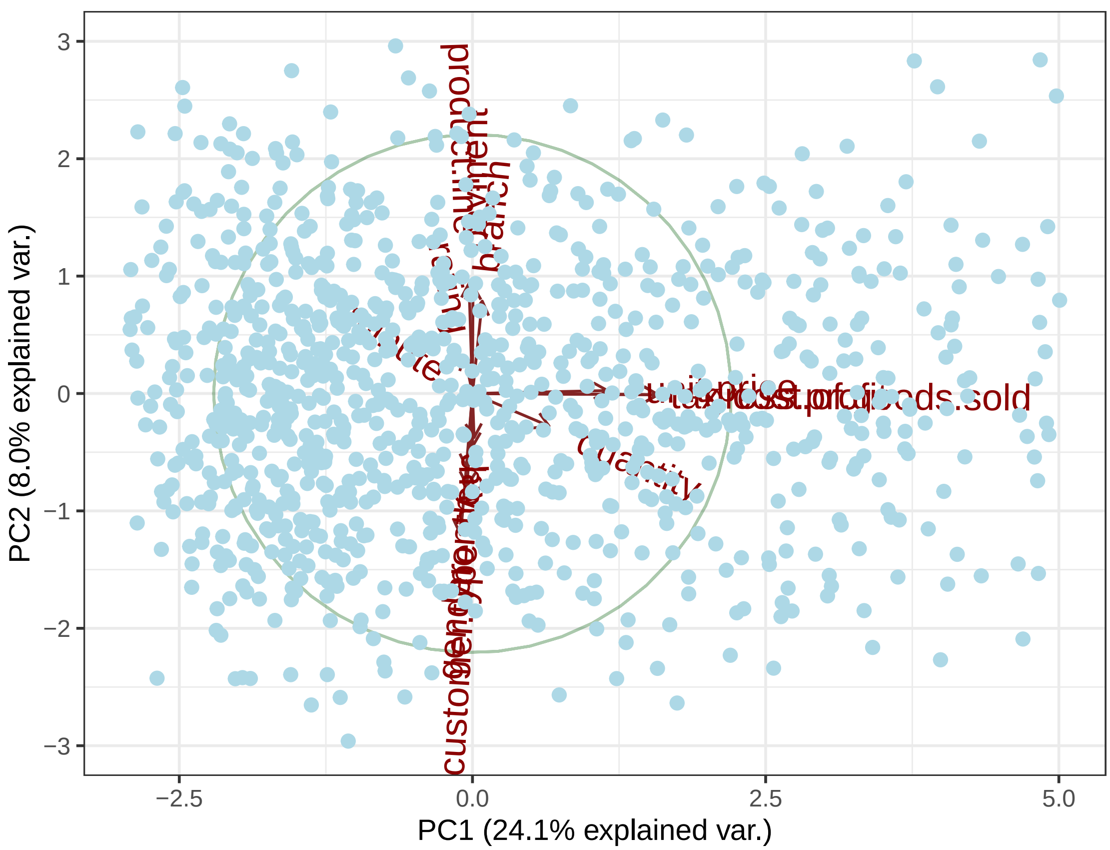
```r
names(salesdf.pca)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```r
summary(salesdf.pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3    PC4     PC5     PC6    PC7
## Standard deviation     1.9024 1.0941 1.08327 1.0514 1.02849 1.00227 0.9965
## Proportion of Variance 0.2413 0.0798 0.07823 0.0737 0.07052 0.06697 0.0662
## Cumulative Proportion  0.2413 0.3211 0.39929 0.4730 0.54351 0.61048 0.6767
##                            PC8    PC9    PC10    PC11    PC12    PC13      PC14
## Standard deviation     0.99476 0.9760 0.93626 0.92005 0.89185 0.62394 4.267e-16
## Proportion of Variance 0.06597 0.0635 0.05844 0.05643 0.05303 0.02595 0.000e+00
## Cumulative Proportion  0.74265 0.8062 0.86459 0.92102 0.97405 1.00000 1.000e+00
##                             PC15
## Standard deviation     1.755e-16
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

```r
# visualization of the PCA

ggbiplot(salesdf.pca, obs.scale = 1, var.scale = 0.5, circle = TRUE, ellipse = TRUE, varname.size = 5) +
  theme_bw() +
  theme(legend.direction = 'horizontal', legend.position = 'top')+
  geom_point(size = 2, color = 'light blue')
```



From the plot, the significant variables affecting our target variable is cost of goods sold, followed by gross profit, the branch, quantity and mode of payment. If we were to create a model we would pick these variables as our top choices because they influence the target variable the most.

6.2 Feature Selection

6.2.1 Filter Methods

```r
# with filter method, we use correlation matrix to filter out the most correlated variables. If two var
# let's create a copy of the dataset to use for feature selection
sales.fs <- sales.cleaned
# step 1: calculating the correlation matrix for the dataset
sales.corr <- cor(sales.fs)
```

```
## Warning in cor(sales.fs): the standard deviation is zero
```

```r
sales.corr
```

```
##                              branch customer.type       gender product.line
## branch                   1.00000000  -0.019607869 -0.056317558   0.03290205
## customer.type           -0.01960787   1.000000000  0.039996160  -0.02510945
## gender                  -0.05631756   0.039996160  1.000000000  -0.06612647
## product.line             0.03290205  -0.025109450 -0.066126475   1.00000000
## unit.price               0.02820244  -0.020237875  0.015444630   0.03842765
## quantity                -0.02001652   0.004744509 -0.061971894  -0.06977620
## tax                      0.04104666  -0.019670283 -0.049450989  -0.01854396
## payment                  0.02055290  -0.069286242 -0.049514182   0.01051098
## cost.of.goods.sold       0.04104666  -0.019670283 -0.049450989  -0.01854396
## gross.margin.percentage          NA            NA           NA           NA
## gross.profit             0.04104666  -0.019670283 -0.049450989  -0.01854396
## rating                   0.01023848   0.018888672  0.004800208   0.02339096
## month                   -0.03530092   0.005972443  0.027533609  -0.04701346
## day                     -0.01308653   0.034124208  0.051156850  -0.02332870
## hour                     0.03300711  -0.018893298  0.084081139   0.03691312
## minute                   0.03837833  -0.012909043  0.009257593  -0.01014963
##                           unit.price     quantity          tax      payment
## branch                   0.028202440 -0.020016524  0.041046665  0.020552896
## customer.type           -0.020237875  0.004744509 -0.019670283 -0.069286242
## gender                   0.015444630 -0.061971894 -0.049450989 -0.049514182
## product.line             0.038427649 -0.069776204 -0.018543956  0.010510982
## unit.price               1.000000000  0.008127624  0.633962089 -0.019637884
## quantity                 0.008127624  1.000000000  0.357573247 -0.029577901
## tax                      0.633962089  0.357573247  1.000000000  0.008823723
## payment                 -0.019637884 -0.029577901  0.008823723  1.000000000
## cost.of.goods.sold       0.633962089  0.357573247  1.000000000  0.008823723
## gross.margin.percentage           NA           NA           NA           NA
## gross.profit             0.633962089  0.357573247  1.000000000  0.008823723
## rating                  -0.008777507  0.017240731 -0.036441705  0.013001094
## month                   -0.027387186  0.020515373 -0.022301340 -0.022555784
## day                      0.057020896 -0.024342312 -0.002514770 -0.028627647
## hour                     0.008242210  0.015210429 -0.002770440 -0.013989800
## minute                  -0.006868818 -0.024797102 -0.027479899 -0.050585696
##                         cost.of.goods.sold gross.margin.percentage gross.profit
## branch                         0.041046665                      NA  0.041046665
## customer.type                 -0.019670283                      NA -0.019670283
## gender                        -0.049450989                      NA -0.049450989
## product.line                  -0.018543956                      NA -0.018543956
## unit.price                     0.633962089                      NA  0.633962089
## quantity                       0.357573247                      NA  0.357573247
## tax                            1.000000000                      NA  1.000000000
```

```
## payment                    0.008823723                      NA  0.008823723
## cost.of.goods.sold         1.000000000                      NA  1.000000000
## gross.margin.percentage             NA                       1           NA
## gross.profit               1.000000000                      NA  1.000000000
## rating                    -0.036441705                      NA -0.036441705
## month                     -0.022301340                      NA -0.022301340
## day                       -0.002514770                      NA -0.002514770
## hour                      -0.002770440                      NA -0.002770440
## minute                    -0.027479899                      NA -0.027479899
##                                 rating        month          day        hour
## branch                     0.010238476 -0.035300925 -0.013086533  0.03300711
## customer.type              0.018888672  0.005972443  0.034124208 -0.01889330
## gender                     0.004800208  0.027533609  0.051156850  0.08408114
## product.line               0.023390962 -0.047013462 -0.023328697  0.03691312
## unit.price                -0.008777507 -0.027387186  0.057020896  0.00824221
## quantity                   0.017240731  0.020515373 -0.024342312  0.01521043
## tax                       -0.036441705 -0.022301340 -0.002514770 -0.00277044
## payment                    0.013001094 -0.022555784 -0.028627647 -0.01398980
## cost.of.goods.sold        -0.036441705 -0.022301340 -0.002514770 -0.00277044
## gross.margin.percentage             NA           NA           NA          NA
## gross.profit              -0.036441705 -0.022301340 -0.002514770 -0.00277044
## rating                     1.000000000 -0.042880374 -0.007075821 -0.03058764
## month                     -0.042880374  1.000000000 -0.118996386  0.04376174
## day                       -0.007075821 -0.118996386  1.000000000  0.02066810
## hour                      -0.030587644  0.043761744  0.020668100  1.00000000
## minute                     0.050558480 -0.006553809  0.012645496 -0.02538363
##                                 minute
## branch                     0.038378328
## customer.type             -0.012909043
## gender                     0.009257593
## product.line              -0.010149626
## unit.price                -0.006868818
## quantity                  -0.024797102
## tax                       -0.027479899
## payment                   -0.050585696
## cost.of.goods.sold        -0.027479899
## gross.margin.percentage             NA
## gross.profit              -0.027479899
## rating                     0.050558480
## month                     -0.006553809
## day                        0.012645496
## hour                      -0.025383629
## minute                     1.000000000
```

We see NA for gross margin percentage. let's delete the column and redo the correlation for the remaining columns.

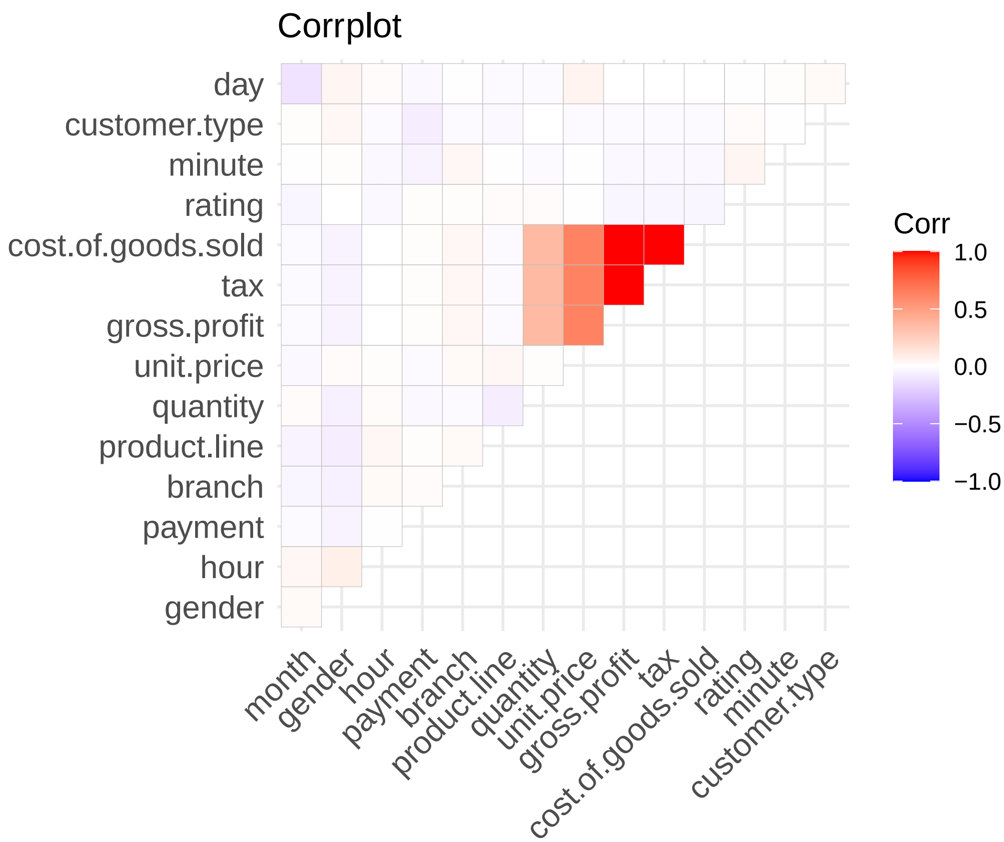
```
sales.fs$gross.margin.percentage <- NULL
salesfs.corr <- cor(sales.fs)
salesfs.corr
```

```
##                            branch customer.type       gender product.line
## branch                 1.00000000  -0.019607869 -0.056317558   0.03290205
## customer.type         -0.01960787   1.000000000  0.039996160  -0.02510945
## gender                -0.05631756   0.039996160  1.000000000  -0.06612647
```

```
## product.line        0.03290205  -0.025109450 -0.066126475   1.00000000
## unit.price          0.02820244  -0.020237875  0.015444630   0.03842765
## quantity           -0.02001652   0.004744509 -0.061971894  -0.06977620
## tax                 0.04104666  -0.019670283 -0.049450989  -0.01854396
## payment             0.02055290  -0.069286242 -0.049514182   0.01051098
## cost.of.goods.sold  0.04104666  -0.019670283 -0.049450989  -0.01854396
## gross.profit        0.04104666  -0.019670283 -0.049450989  -0.01854396
## rating              0.01023848   0.018888672  0.004800208   0.02339096
## month              -0.03530092   0.005972443  0.027533609  -0.04701346
## day                -0.01308653   0.034124208  0.051156850  -0.02332870
## hour                0.03300711  -0.018893298  0.084081139   0.03691312
## minute              0.03837833  -0.012909043  0.009257593  -0.01014963
##                      unit.price     quantity          tax      payment
## branch              0.028202440 -0.020016524  0.041046665  0.020552896
## customer.type      -0.020237875  0.004744509 -0.019670283 -0.069286242
## gender              0.015444630 -0.061971894 -0.049450989 -0.049514182
## product.line        0.038427649 -0.069776204 -0.018543956  0.010510982
## unit.price          1.000000000  0.008127624  0.633962089 -0.019637884
## quantity            0.008127624  1.000000000  0.357573247 -0.029577901
## tax                 0.633962089  0.357573247  1.000000000  0.008823723
## payment            -0.019637884 -0.029577901  0.008823723  1.000000000
## cost.of.goods.sold  0.633962089  0.357573247  1.000000000  0.008823723
## gross.profit        0.633962089  0.357573247  1.000000000  0.008823723
## rating             -0.008777507  0.017240731 -0.036441705  0.013001094
## month              -0.027387186  0.020515373 -0.022301340 -0.022555784
## day                 0.057020896 -0.024342312 -0.002514770 -0.028627647
## hour                0.008242210  0.015210429 -0.002770440 -0.013989800
## minute             -0.006868818 -0.024797102 -0.027479899 -0.050585696
##                    cost.of.goods.sold gross.profit       rating        month
## branch                   0.041046665  0.041046665  0.010238476 -0.035300925
## customer.type           -0.019670283 -0.019670283  0.018888672  0.005972443
## gender                  -0.049450989 -0.049450989  0.004800208  0.027533609
## product.line            -0.018543956 -0.018543956  0.023390962 -0.047013462
## unit.price               0.633962089  0.633962089 -0.008777507 -0.027387186
## quantity                 0.357573247  0.357573247  0.017240731  0.020515373
## tax                      1.000000000  1.000000000 -0.036441705 -0.022301340
## payment                  0.008823723  0.008823723  0.013001094 -0.022555784
## cost.of.goods.sold       1.000000000  1.000000000 -0.036441705 -0.022301340
## gross.profit             1.000000000  1.000000000 -0.036441705 -0.022301340
## rating                  -0.036441705 -0.036441705  1.000000000 -0.042880374
## month                   -0.022301340 -0.022301340 -0.042880374  1.000000000
## day                     -0.002514770 -0.002514770 -0.007075821 -0.118996386
## hour                    -0.002770440 -0.002770440 -0.030587644  0.043761744
## minute                  -0.027479899 -0.027479899  0.050558480 -0.006553809
##                             day         hour       minute
## branch             -0.013086533  0.03300711  0.038378328
## customer.type       0.034124208 -0.01889330 -0.012909043
## gender              0.051156850  0.08408114  0.009257593
## product.line       -0.023328697  0.03691312 -0.010149626
## unit.price          0.057020896  0.00824221 -0.006868818
## quantity           -0.024342312  0.01521043 -0.024797102
## tax                -0.002514770 -0.00277044 -0.027479899
## payment            -0.028627647 -0.01398980 -0.050585696
## cost.of.goods.sold -0.002514770 -0.00277044 -0.027479899
```

```
## gross.profit       -0.002514770 -0.00277044 -0.027479899
## rating             -0.007075821 -0.03058764  0.050558480
## month              -0.118996386  0.04376174 -0.006553809
## day                 1.000000000  0.02066810  0.012645496
## hour                0.020668100  1.00000000 -0.025383629
## minute              0.012645496 -0.02538363  1.000000000
```

```
sales.fs %>% cor %>% ggcorrplot(method = "square",  lab = FALSE, type = "upper", hc.order = TRUE, title
```




```
# finding highly correlated attributes

high.corr <- findCorrelation(salesfs.corr, cutoff = 0.75)
names(sales.fs[, high.corr])
```

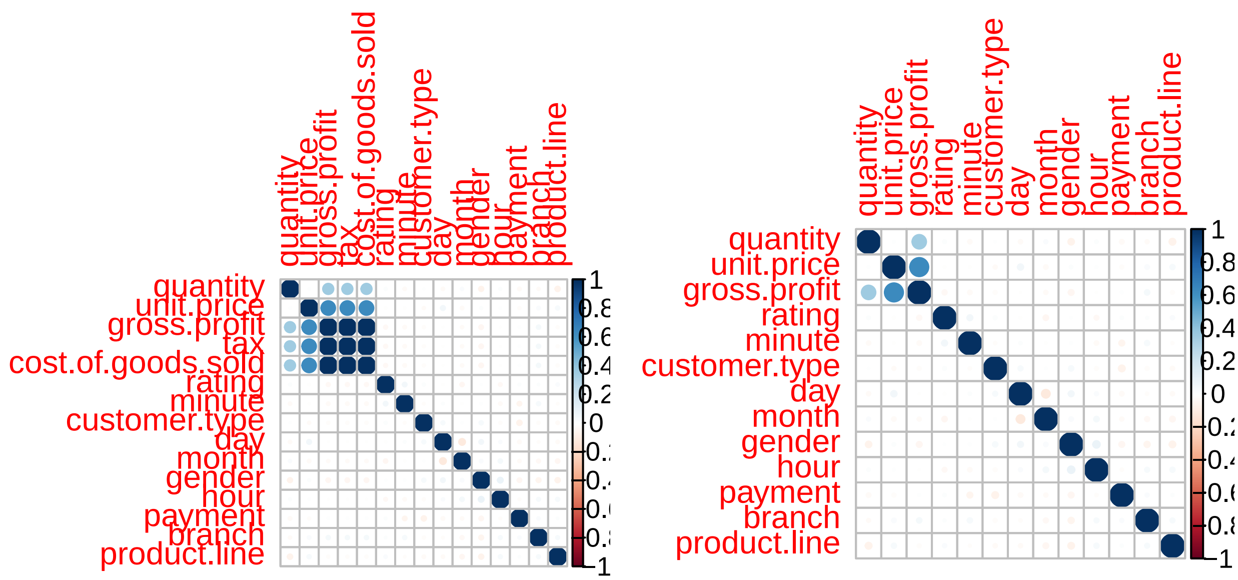
```
## [1] "tax"                "cost.of.goods.sold"
```

```
# removing highly correlated features to remove redundant features

sales.fs2 <- sales.fs[,-high.corr]
head(sales.fs2)
```

```
##   branch customer.type gender product.line unit.price quantity payment
## 1      1             1      1            1      74.69        8       1
## 2      3             2      1            2      15.28        6       2
## 3      1             2      2            3      46.33        8       3
## 4      1             1      2            1      58.22        9       1
## 5      1             2      2            4      86.31        8       1
## 6      3             2      2            2      85.39        8       1
```

```
##   gross.profit rating month day hour minute
## 1      26.1415    9.1     1   5    4      9
## 2       3.8200    9.6     3   8    1     30
## 3      16.2155    7.4     3   3    4     24
## 4      23.2880    8.4     1  27   11     34
## 5      30.2085    5.3     2   8    1     38
## 6      29.8865    4.1     3  25    9     31
```

```
# we can now compare the two correlations, before and after removing redundant variables
par(mfrow = c(1, 2), pty = "s")
corrplot(salesfs.corr,order = "hclust")
corrplot(cor(sales.fs2), order = "hclust")
```



6.2.2 Wrapper Methods

```
# making a copy of df to use for wrapper method

sales.wm <- sales.df
glimpse(sales.wm)
```

```
## Rows: 1,000
## Columns: 15
## $ branch            <dbl> 1, 3, 1, 1, 1, 3, 1, 3, 1, 2, 2, 2, 1, 1, 1, 2, ...
## $ customer.type     <dbl> 1, 2, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, ...
## $ gender            <dbl> 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, ...
## $ product.line      <dbl> 1, 2, 3, 1, 4, 2, 2, 3, 1, 5, 6, 2, 2, 5, 1, 4, ...
## $ unit.price        <dbl> 74.69, 15.28, 46.33, 58.22, 86.31, 85.39, 68.84,...
```

```
## $ quantity           <dbl> 8, 6, 8, 9, 8, 8, 7, 2, 3, 4, 5, 5, 6, 2, 2, 7, ...
## $ tax                <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085, 29.8...
## $ payment            <dbl> 1, 2, 3, 1, 1, 1, 1, 1, 3, 3, 1, 2, 1, 1, 2, 2, ...
## $ cost.of.goods.sold <dbl> 522.83, 76.40, 324.31, 465.76, 604.17, 597.73, 4...
## $ gross.profit       <dbl> 26.1415, 3.8200, 16.2155, 23.2880, 30.2085, 29.8...
## $ rating             <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0, 7.2, 5.9...
## $ month              <dbl> 1, 3, 3, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2, 2, 3, 1, ...
## $ day                <dbl> 5, 8, 3, 27, 8, 25, 25, 24, 10, 20, 6, 9, 12, 7,...
## $ hour               <dbl> 4, 1, 4, 11, 1, 9, 5, 2, 8, 4, 9, 8, 1, 7, 10, 7...
## $ minute             <dbl> 9, 30, 24, 34, 38, 31, 37, 39, 16, 28, 8, 4, 26,...
```

```
# we begin with a sequential forward greedy search
wrapper = clustvarsel(sales.wm, G = 1:15)
wrapper
```

```
## ------------------------------------------------------
## Variable selection for Gaussian model-based clustering
## Stepwise (forward/backward) greedy search
## ------------------------------------------------------
##
##  Variable proposed Type of step  BICclust Model  G   BICdiff Decision
##               hour          Add -4267.096     E 10  901.6667 Accepted
##             branch          Add -6237.760   VEV  4  477.3458 Accepted
##              month          Add -9395.372   EEI  5 -666.9596 Rejected
##             branch       Remove -4267.096     E 10  477.3458 Rejected
##
## Selected subset: hour, branch
```

From the search two variables were selected: hour and branch, the rest were rejected.

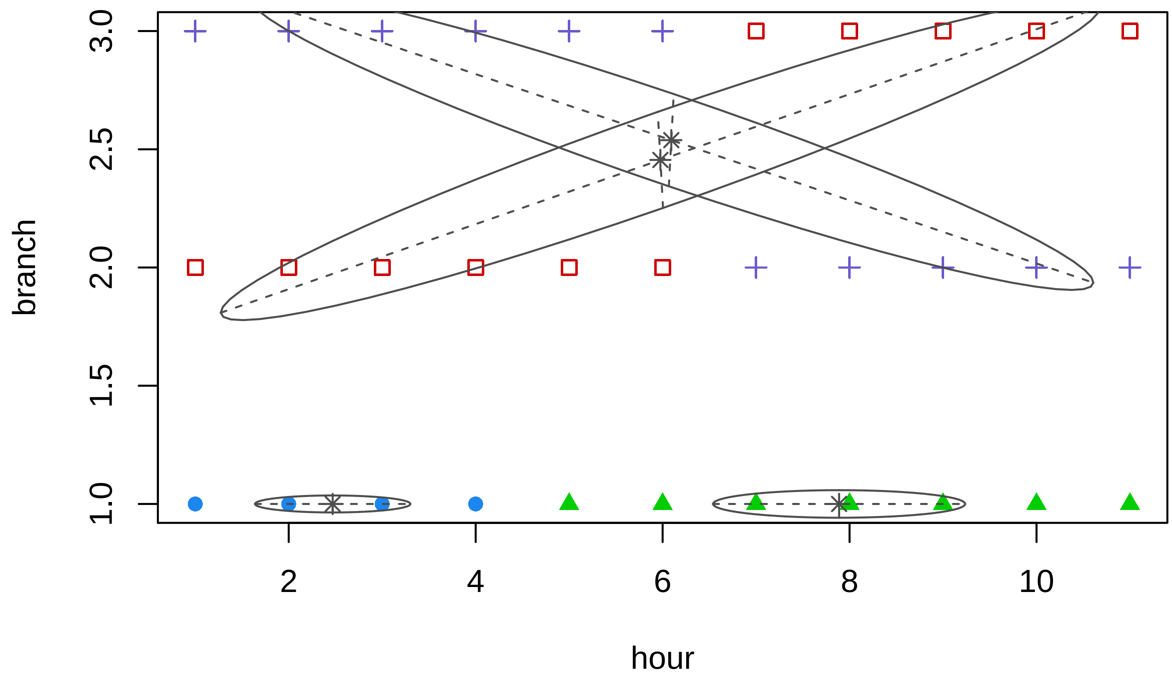
```
# the variables identified to use for clustering are: hour and branch. Let's proceed to the modeling pa

selected = sales.wm[, wrapper$subset] # choosing the selected variables
glimpse(selected)
```

```
## Rows: 1,000
## Columns: 2
## $ hour   <dbl> 4, 1, 4, 11, 1, 9, 5, 2, 8, 4, 9, 8, 1, 7, 10, 7, 2, 1, 9, 6...
## $ branch <dbl> 1, 3, 1, 1, 1, 3, 1, 3, 1, 2, 2, 2, 1, 1, 1, 2, 1, 1, 1, 2, ...
```

```
model = Mclust(selected, G = 1:15) #building the model
summary(model)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 4 components:
##
##  log-likelihood    n df      BIC       ICL
##       -3049.802 1000 20 -6237.76 -6297.175
##
## Clustering table:
##   1   2   3   4
## 137 327 203 333
```

```r
# plotting the model
plot(model,c("classification"))
```



6.2.3 Embedded methods

```r
# embedded method is know to be suitable for very high dimensional dataset. Ours has 15 columms after c
# we first make a copy to use for this implementation part

sales.em <- sales.df
set.seed(2)
em.model <- ewkm(sales.em[1:15], 3, maxiter = 1000)
em.model$size
```

```
## [1] 379 231 390
```

```r
em.model$centers
```

```
##     branch customer.type   gender product.line unit.price quantity       tax
## 1 2.010554      1.490765 1.000000     3.672823   46.72037 5.076517 10.325594
## 2 2.038961      1.467532 1.471861     3.484848   80.65697 6.623377 33.299985
## 3 1.935897      1.525641 2.000000     3.530769   49.57269 4.802564  9.676083
##    payment cost.of.goods.sold gross.profit   rating    month      day     hour
## 1 2.010554           206.5119    10.325594 7.000792 1.992084 14.90765 5.701847
## 2 1.982684           665.9997    33.299985 6.859740 1.948052 15.16883 5.718615
## 3 1.912821           193.5217     9.676083 7.012308 2.020513 15.64615 6.225641
##     minute
## 1 30.98417
```

```
## 2 29.83550
## 3 31.95641
```

```r
names(em.model)
```

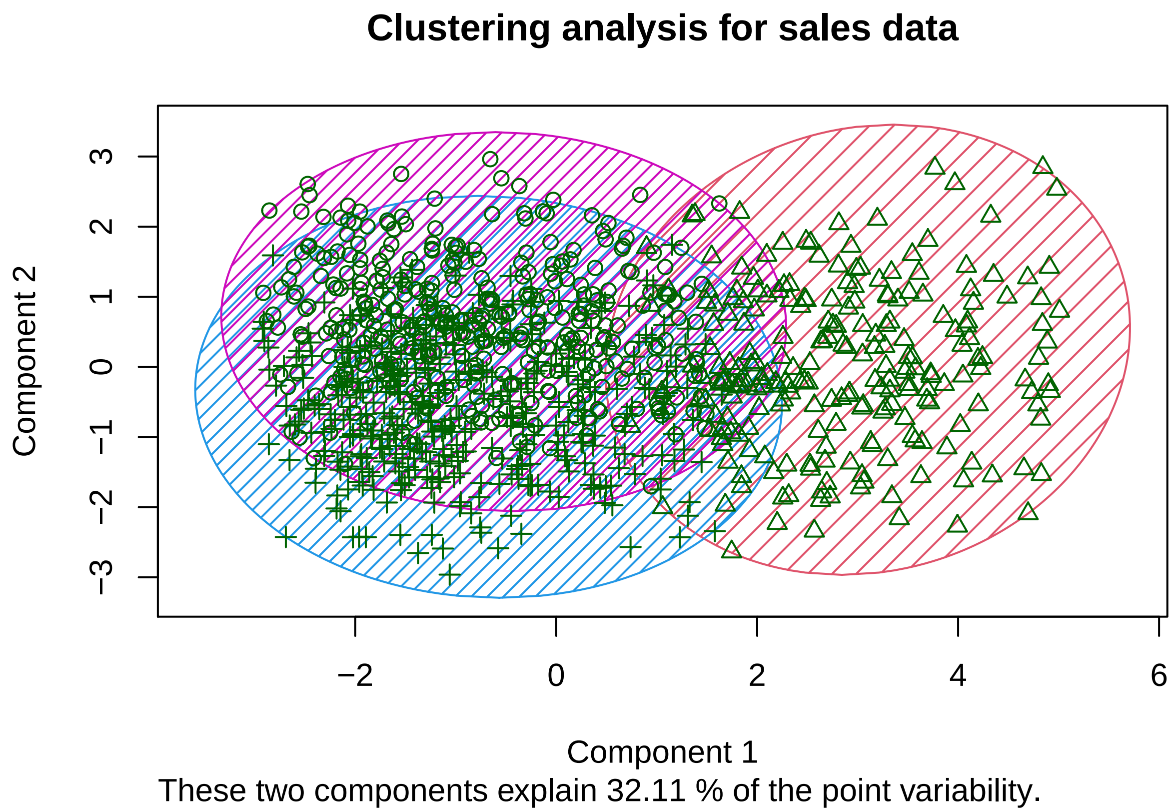
```
##  [1] "cluster"          "centers"          "totss"            "withinss"
##  [5] "tot.withinss"     "betweenss"        "size"             "iterations"
##  [9] "total.iterations" "restarts"         "weights"
```

```r
round(em.model$weights*100,2)
```

```
##   branch customer.type gender product.line unit.price quantity tax payment
## 1      0          0.00  99.99            0          0        0   0       0
## 2      0         51.51  48.48            0          0        0   0       0
## 3      0          0.00  99.99            0          0        0   0       0
##   cost.of.goods.sold gross.profit rating month day hour minute
## 1                  0            0      0     0   0    0      0
## 2                  0            0      0     0   0    0      0
## 3                  0            0      0     0   0    0      0
```

In cluster 1 and 3 gender carry's the most weight, while in cluster 2, customer type and gender carry about the same weight.

```r
# plotting cluster for 1st and second PC
clusplot(sales.em[1:15], em.model$cluster, color = TRUE, shade = TRUE, lines = 1, main = "Clustering ana
```



7. Follow up questions: did we have the right data? Yes we did.