CarreFour Data Analysis in R

Snow

9/7/2021

## Define the question

You are a Data analyst at Carrefour Kenya and are currently undertaking a project that will inform the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax).

## Metric for success

In order to work on the above problem, you need to do the following:

* Define the question- the metric for success, the context,experimental design taken and the appropriateness of the available data to answer the given question.
* Find and deal with outliers, anomalies, and missing data within the dataset.
* Perform univariate and bivariate analysis.
* From your insights provide a conclusion and recommendation.
* Build an associative model and visualize some of the rules
* Create a plot of anomalies using the dataset provided.

## Data Understanding (the context)

Your project has been divided into four parts where you’ll explore a recent marketing dataset by performing various unsupervised learning techniques and later providing recommendations based on your insights. 1. Part 1: Dimensionality Reduction 2. Part 2: Feature Selection 3. Part 3: Association Rules 4. Part 4: Anomaly Detection

In order to work on the above problem, you need to do the following:

* Define the question, the metric for success, the context, experimental design taken and the appropriateness of the available data to answer the given question.
* Find and deal with outliers, anomalies, and missing data within the dataset.
* Perform univariate and bivariate analysis.
* From your insights provide a conclusion and recommendation.
* Build the associative model and inspect the rules.

## Experimental design

1. Import the data to R
2. Perform data exploration
3. Define metrics for success
4. Perform Univariate and Bivariate data Analysis
5. Build an associative model
6. Provide conclusion

library(superml)

## Loading required package: R6

library(naniar)  
library(ggplot2)  
library(Rtsne)  
library(data.table)  
library(ggbiplot)

## Loading required package: plyr

## Loading required package: scales

## Loading required package: grid

library(tibbletime)

##   
## Attaching package: 'tibbletime'

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

#Part 1

df <- fread("http://bit.ly/CarreFourDataset")  
head(df)

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

dim(df)

## [1] 1000 16

## Data cleaning processes.

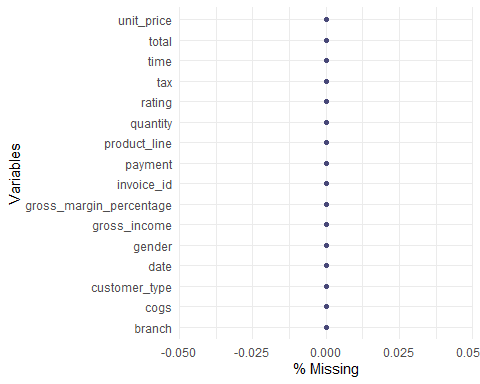
df.copy <- copy(df)

# clean data by changing the names of columns  
# First we need to change the column names to lowercase and remove and replace spaces with an underscore.   
# replace the spaces with underscores using gsub() function  
names(df) <- gsub(" ","\_", names(df))  
  
# lowercase  
names(df) <- tolower(names(df))  
  
# display the column names to confirm the changes  
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"

# check for missing and duplicates  
gg\_miss\_var(df, show\_pct = TRUE)

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please  
## use `guide = "none"` instead.



colSums(is.na(df))

## invoice\_id branch customer\_type   
## 0 0 0   
## gender product\_line unit\_price   
## 0 0 0   
## quantity tax date   
## 0 0 0   
## time payment cogs   
## 0 0 0   
## gross\_margin\_percentage gross\_income rating   
## 0 0 0   
## total   
## 0

From the dataset there is no row with missing data. Now, let’s check for duplicates.

# Duplicates  
dup <- df[duplicated(df),]  
dup

## Empty data.table (0 rows and 16 cols): invoice\_id,branch,customer\_type,gender,product\_line,unit\_price...

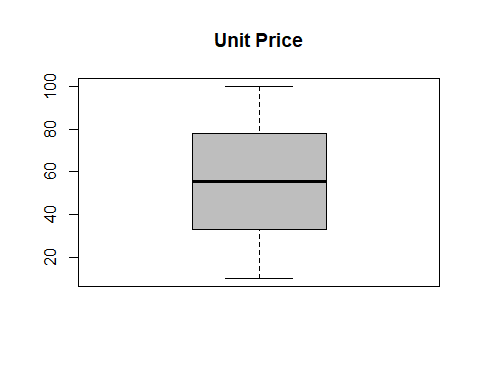
There is also no duplicates in the data.

##Univarate Bivatiate and Multivariate Analysis (EDA)

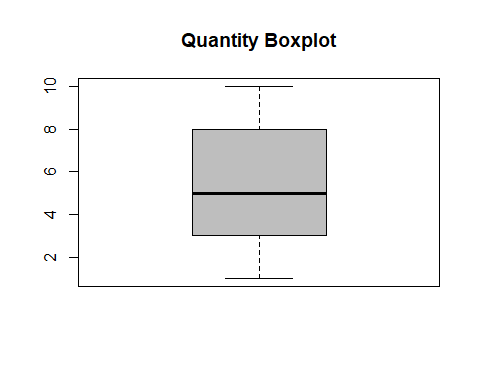
head(df)

## invoice\_id branch customer\_type gender product\_line unit\_price  
## 1: 750-67-8428 A Member Female Health and beauty 74.69  
## 2: 226-31-3081 C Normal Female Electronic accessories 15.28  
## 3: 631-41-3108 A Normal Male Home and lifestyle 46.33  
## 4: 123-19-1176 A Member Male Health and beauty 58.22  
## 5: 373-73-7910 A Normal Male Sports and travel 86.31  
## 6: 699-14-3026 C Normal Male Electronic accessories 85.39  
## quantity tax date time payment cogs gross\_margin\_percentage  
## 1: 7 26.1415 1/5/2019 13:08 Ewallet 522.83 4.761905  
## 2: 5 3.8200 3/8/2019 10:29 Cash 76.40 4.761905  
## 3: 7 16.2155 3/3/2019 13:23 Credit card 324.31 4.761905  
## 4: 8 23.2880 1/27/2019 20:33 Ewallet 465.76 4.761905  
## 5: 7 30.2085 2/8/2019 10:37 Ewallet 604.17 4.761905  
## 6: 7 29.8865 3/25/2019 18:30 Ewallet 597.73 4.761905  
## gross\_income rating total  
## 1: 26.1415 9.1 548.9715  
## 2: 3.8200 9.6 80.2200  
## 3: 16.2155 7.4 340.5255  
## 4: 23.2880 8.4 489.0480  
## 5: 30.2085 5.3 634.3785  
## 6: 29.8865 4.1 627.6165

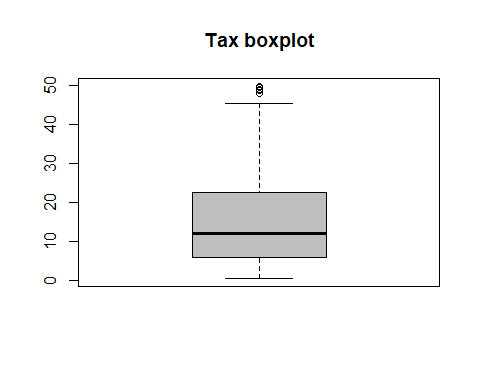
boxplot(df$unit\_price,col='grey', main = 'Unit Price')



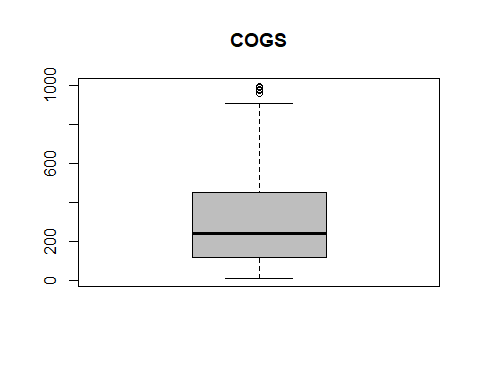
boxplot(df$quantity,col='grey', main = 'Quantity Boxplot')



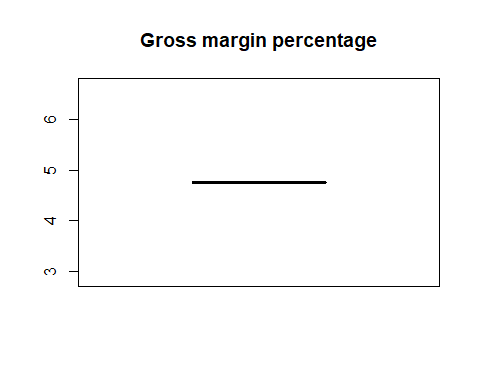
boxplot(df$tax,col='grey', main = 'Tax boxplot')



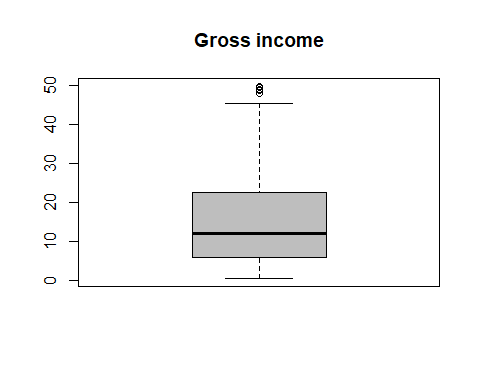
boxplot(df$cogs,col='grey', main = 'COGS')



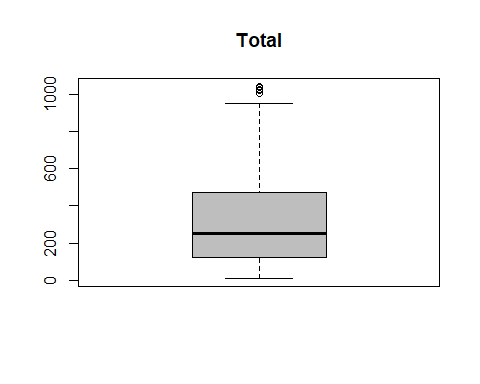
boxplot(df$gross\_margin\_percentage,col='grey', main = 'Gross margin percentage')



boxplot(df$gross\_income,col='grey', main = 'Gross income')



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

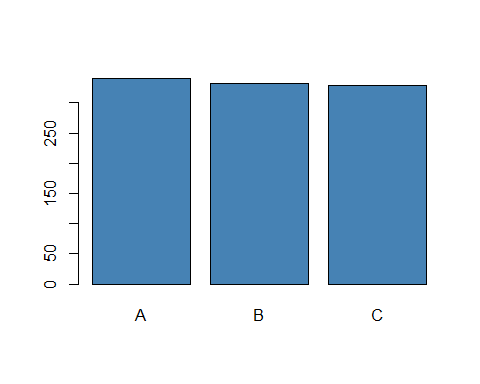
 On the numerical columns, there are a few outliers. However, we will not drop these outliers because they might part of some gooods with high taxes hence the overall price will be higher.

head(df)

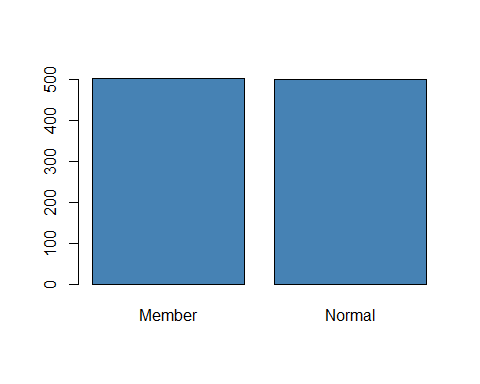
## invoice\_id branch customer\_type gender product\_line unit\_price  
## 1: 750-67-8428 A Member Female Health and beauty 74.69  
## 2: 226-31-3081 C Normal Female Electronic accessories 15.28  
## 3: 631-41-3108 A Normal Male Home and lifestyle 46.33  
## 4: 123-19-1176 A Member Male Health and beauty 58.22  
## 5: 373-73-7910 A Normal Male Sports and travel 86.31  
## 6: 699-14-3026 C Normal Male Electronic accessories 85.39  
## quantity tax date time payment cogs gross\_margin\_percentage  
## 1: 7 26.1415 1/5/2019 13:08 Ewallet 522.83 4.761905  
## 2: 5 3.8200 3/8/2019 10:29 Cash 76.40 4.761905  
## 3: 7 16.2155 3/3/2019 13:23 Credit card 324.31 4.761905  
## 4: 8 23.2880 1/27/2019 20:33 Ewallet 465.76 4.761905  
## 5: 7 30.2085 2/8/2019 10:37 Ewallet 604.17 4.761905  
## 6: 7 29.8865 3/25/2019 18:30 Ewallet 597.73 4.761905  
## gross\_income rating total  
## 1: 26.1415 9.1 548.9715  
## 2: 3.8200 9.6 80.2200  
## 3: 16.2155 7.4 340.5255  
## 4: 23.2880 8.4 489.0480  
## 5: 30.2085 5.3 634.3785  
## 6: 29.8865 4.1 627.6165

The next step would be to find the frequency on the different categorical columns

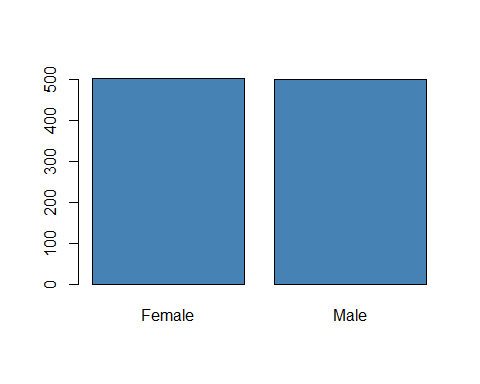
# Frequency of categorical columns  
#Branch , customer\_type, Gender, productline , payment  
branch <- table(df$branch)  
barplot(branch, col = "steelblue")



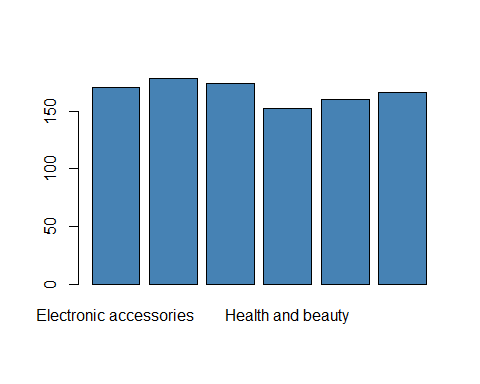
customer\_type\_freq <- table (df$customer\_type)  
barplot(customer\_type\_freq, col = "steelblue")



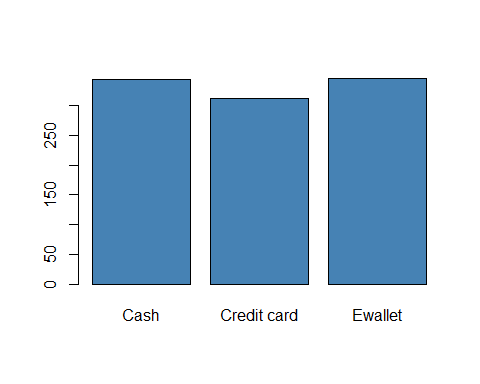
gender <- table(df$gender)  
barplot(gender, col = "steelblue")



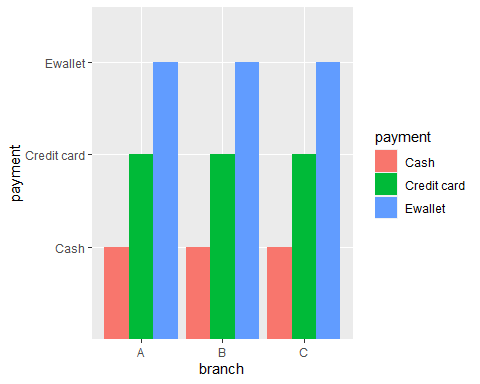
product\_line <- table(df$product\_line)  
barplot(product\_line, col = "steelblue")



payment <- table(df$payment)  
barplot(payment, col = "steelblue")

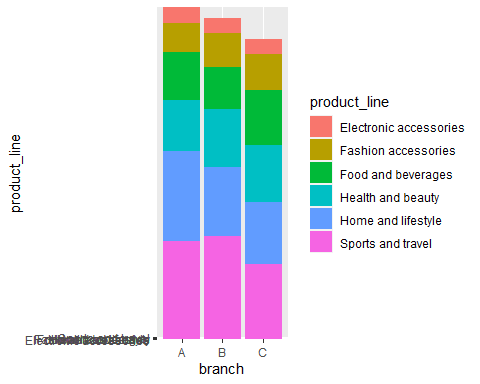
 From the bar plots above we can conclude that: - The data is collected on Branches A, B and C equally. - The information collected was half from the members and half from the normal customers. - The gender was equally balances in the data. - Most people paid their bills with E wallet and cash rather than Credit card

ggplot(df, aes(fill=payment, y= payment, x=branch)) +   
 geom\_bar(position="dodge", stat="identity")



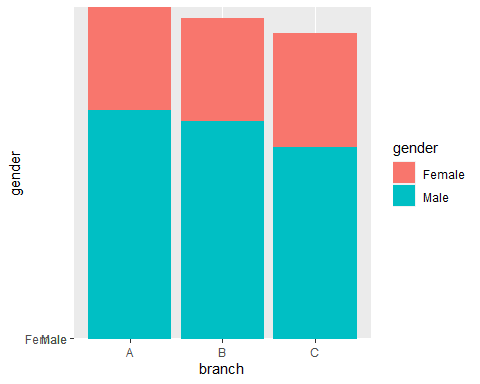
From the data, Ewallet payments are the most popular in all the three branches.

ggplot(df, aes(fill=product\_line, y= product\_line, x=branch)) +   
 geom\_bar(position="stack", stat="identity")



From the plot, Branch B sells more sports and travel goods than the other branches. Branch A sells more home and lifestyle goods than the other branches. Therefore, the marketing team should stack these branches with the product with which they sell more.

ggplot(df, aes(fill=gender, y= gender, x=branch)) +   
 geom\_bar(position="stack", stat="identity")

 There are more males in the Carrefour branches than the females. This is not what many people assume as many people erroneously think that there are usually more females doing shopping.

Measures of central tendency for the numerical columns

# numerical columns.   
num\_col <- unlist(lapply(df, is.numeric))  
  
  
df\_num <- subset(df, select = num\_col)  
  
head (df\_num)

## 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

#Getting the measures of dispersion in the numerical columns.   
  
summary\_stats <- data.frame(  
 Mean = apply(df\_num, 2, mean),   
 Median = apply(df\_num, 2, median),   
 Min = apply(df\_num, 2, min),   
 Max = apply(df\_num, 2, max))   
summary\_stats

## Mean Median Min Max  
## unit\_price 55.672130 55.230000 10.080000 99.960000  
## quantity 5.510000 5.000000 1.000000 10.000000  
## tax 15.379369 12.088000 0.508500 49.650000  
## cogs 307.587380 241.760000 10.170000 993.000000  
## gross\_margin\_percentage 4.761905 4.761905 4.761905 4.761905  
## gross\_income 15.379369 12.088000 0.508500 49.650000  
## rating 6.972700 7.000000 4.000000 10.000000  
## total 322.966749 253.848000 10.678500 1042.650000

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

# Mode  
mode.unit\_price <- getmode(df$unit\_price)  
mode.unit\_price

## [1] 83.77

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

## [1] 10

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

## [1] 39.48

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

## [1] 789.6

mode.gross\_income <- getmode(df$gross\_income)  
mode.gross\_income

## [1] 39.48

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

## [1] 6

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

## [1] 829.08

# Label Encoder  
#Branch , customer\_type, Gender, productline , payment  
lbl <- LabelEncoder$new()  
lbl$fit(df$branch)  
df$branch <- lbl$fit\_transform(df$branch)  
  
lbl$fit(df$customer\_type)  
df$customer\_type <- lbl$fit\_transform(df$customer\_type)  
  
lbl$fit(df$gender)  
df$gender <- lbl$fit\_transform(df$gender)  
  
lbl$fit(df$product\_line)  
df$product\_line <- lbl$fit\_transform(df$product\_line)  
  
lbl$fit(df$payment)  
df$payment <- lbl$fit\_transform(df$payment)

str(df)

## Classes 'data.table' and 'data.frame': 1000 obs. of 16 variables:  
## $ invoice\_id : chr "750-67-8428" "226-31-3081" "631-41-3108" "123-19-1176" ...  
## $ branch : num 0 1 0 0 0 1 0 1 0 2 ...  
## $ customer\_type : num 0 1 1 0 1 1 0 1 0 0 ...  
## $ gender : num 0 0 1 1 1 1 0 0 0 0 ...  
## $ product\_line : num 0 1 2 0 3 1 1 2 0 4 ...  
## $ unit\_price : num 74.7 15.3 46.3 58.2 86.3 ...  
## $ quantity : int 7 5 7 8 7 7 6 10 2 3 ...  
## $ tax : num 26.14 3.82 16.22 23.29 30.21 ...  
## $ date : chr "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...  
## $ time : chr "13:08" "10:29" "13:23" "20:33" ...  
## $ payment : num 0 1 2 0 0 0 0 0 2 2 ...  
## $ cogs : num 522.8 76.4 324.3 465.8 604.2 ...  
## $ gross\_margin\_percentage: num 4.76 4.76 4.76 4.76 4.76 ...  
## $ gross\_income : num 26.14 3.82 16.22 23.29 30.21 ...  
## $ rating : num 9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...  
## $ total : num 549 80.2 340.5 489 634.4 ...  
## - attr(\*, ".internal.selfref")=<externalptr>

# Since the gross margin percentage has only one value we can drop the column.   
table(df$gross\_margin\_percentage)

##   
## 4.761904762   
## 1000

df$gross\_margin\_percentage <- NULL

# Drop the categorcal columns   
df$invoice\_id <- NULL  
df$date <- NULL  
df$time <- NULL

# Separate the data   
df.x <- df[ , 1:11]  
df.y <- df[, 12]

head(df.x)

## branch customer\_type gender product\_line unit\_price quantity tax payment  
## 1: 0 0 0 0 74.69 7 26.1415 0  
## 2: 1 1 0 1 15.28 5 3.8200 1  
## 3: 0 1 1 2 46.33 7 16.2155 2  
## 4: 0 0 1 0 58.22 8 23.2880 0  
## 5: 0 1 1 3 86.31 7 30.2085 0  
## 6: 1 1 1 1 85.39 7 29.8865 0  
## cogs gross\_income rating  
## 1: 522.83 26.1415 9.1  
## 2: 76.40 3.8200 9.6  
## 3: 324.31 16.2155 7.4  
## 4: 465.76 23.2880 8.4  
## 5: 604.17 30.2085 5.3  
## 6: 597.73 29.8865 4.1

head(df.y)

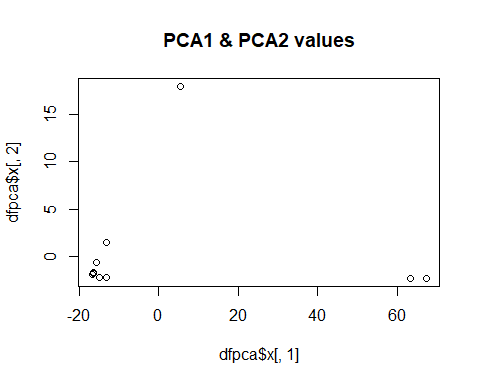
## total  
## 1: 548.9715  
## 2: 80.2200  
## 3: 340.5255  
## 4: 489.0480  
## 5: 634.3785  
## 6: 627.6165

# perform tsne  
tsne = Rtsne(df.x, dims = 2, perplexity = 30)

#visualize TSNE  
  
df.tsne = data.frame(tsne$Y)   
ggplot(df.tsne, aes(x=X1, y=X2)) + geom\_point(size=2)

 ## Performing the PCA

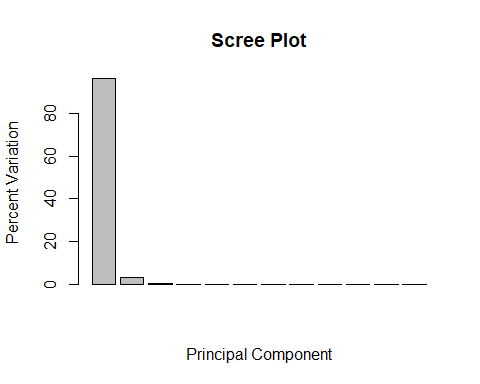
# Run the PCA on the df  
dfpca <- prcomp(t(df),center = TRUE, scale=TRUE)   
  
## plot pc1 and pc2  
plot(dfpca$x[,1], dfpca$x[,2], main = "PCA1 & PCA2 values")



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

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

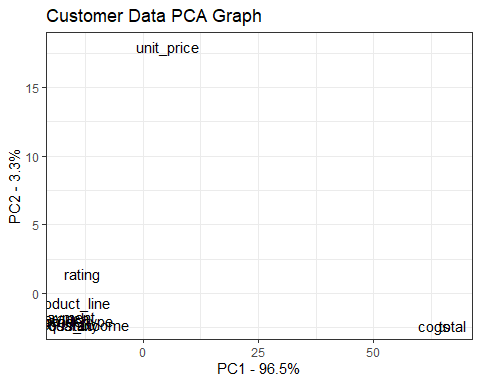
## make a scree plot  
pca.var <- dfpca$sdev^2  
pca.var.per <- round(pca.var/sum(pca.var)\*100, 1)  
  
barplot(pca.var.per, main="Scree Plot", xlab="Principal Component", ylab="Percent Variation")



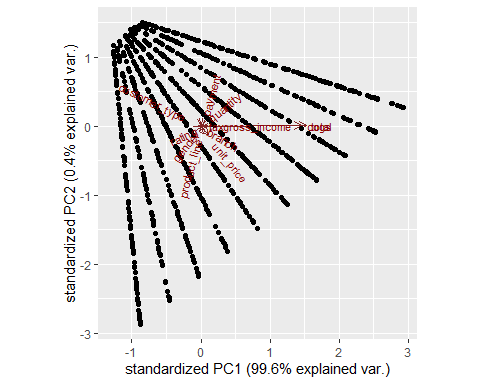
## plot that shows the PCs and the variation:  
  
  
pca.data <- data.frame(Sample=rownames(dfpca$x),  
 X=dfpca$x[,1],  
 Y=dfpca$x[,2])  
pca.data

## Sample X Y  
## branch branch -16.460925 -1.774218  
## customer\_type customer\_type -16.728657 -1.974943  
## gender gender -16.727800 -1.955175  
## product\_line product\_line -15.501089 -0.625429  
## unit\_price unit\_price 5.501295 17.977265  
## quantity quantity -14.979897 -2.249242  
## tax tax -13.006234 -2.255524  
## payment payment -16.446861 -1.686001  
## cogs cogs 63.189817 -2.333115  
## gross\_income gross\_income -13.006234 -2.255524  
## rating rating -13.033551 1.469104  
## total total 67.200135 -2.337199

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

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

library(ggbiplot)  
ggbiplot (prcomp(df))



# Part 2: Feature Selection

## using the filter method.

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

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

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

## [1] 7 9 10

#names (df[,highlyCorrelated])

correlationMatrix

## branch customer\_type gender product\_line unit\_price  
## branch 1.000000000 -0.004899261 -0.012218875 0.01257525 0.013763477  
## customer\_type -0.004899261 1.000000000 0.039996160 -0.02510945 -0.020237875  
## gender -0.012218875 0.039996160 1.000000000 -0.06612647 0.015444630  
## product\_line 0.012575246 -0.025109450 -0.066126475 1.00000000 0.038427649  
## unit\_price 0.013763477 -0.020237875 0.015444630 0.03842765 1.000000000  
## quantity 0.002120920 -0.016762706 -0.074258307 -0.06251471 0.010777564  
## tax 0.012811933 -0.019670283 -0.049450989 -0.01854396 0.633962089  
## payment 0.026725563 -0.069286242 -0.049514182 0.01051098 -0.019637884  
## cogs 0.012811933 -0.019670283 -0.049450989 -0.01854396 0.633962089  
## gross\_income 0.012811933 -0.019670283 -0.049450989 -0.01854396 0.633962089  
## rating -0.049585348 0.018888672 0.004800208 0.02339096 -0.008777507  
## total 0.012811933 -0.019670283 -0.049450989 -0.01854396 0.633962089  
## quantity tax payment cogs gross\_income  
## branch 0.002120920 0.012811933 0.026725563 0.012811933 0.012811933  
## customer\_type -0.016762706 -0.019670283 -0.069286242 -0.019670283 -0.019670283  
## gender -0.074258307 -0.049450989 -0.049514182 -0.049450989 -0.049450989  
## product\_line -0.062514713 -0.018543956 0.010510982 -0.018543956 -0.018543956  
## unit\_price 0.010777564 0.633962089 -0.019637884 0.633962089 0.633962089  
## quantity 1.000000000 0.705510186 0.007333388 0.705510186 0.705510186  
## tax 0.705510186 1.000000000 0.008823723 1.000000000 1.000000000  
## payment 0.007333388 0.008823723 1.000000000 0.008823723 0.008823723  
## cogs 0.705510186 1.000000000 0.008823723 1.000000000 1.000000000  
## gross\_income 0.705510186 1.000000000 0.008823723 1.000000000 1.000000000  
## rating -0.015814905 -0.036441705 0.013001094 -0.036441705 -0.036441705  
## total 0.705510186 1.000000000 0.008823723 1.000000000 1.000000000  
## rating total  
## branch -0.049585348 0.012811933  
## customer\_type 0.018888672 -0.019670283  
## gender 0.004800208 -0.049450989  
## product\_line 0.023390962 -0.018543956  
## unit\_price -0.008777507 0.633962089  
## quantity -0.015814905 0.705510186  
## tax -0.036441705 1.000000000  
## payment 0.013001094 0.008823723  
## cogs -0.036441705 1.000000000  
## gross\_income -0.036441705 1.000000000  
## rating 1.000000000 -0.036441705  
## total -0.036441705 1.000000000

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

## [1] "tax"

names (df[, 9])

## [1] "cogs"

names (df[, 11])

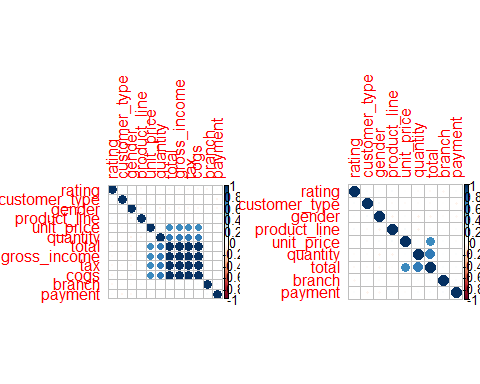
## [1] "rating"

# Next step is removing the variables with high correlation   
df\_low <- df[-highlyCorrelated]  
df\_low$tax <- NULL  
df\_low$cogs <- NULL  
df\_low$gross\_income <- NULL

cor2 <- cor(df\_low)  
cor2

## branch customer\_type gender product\_line unit\_price  
## branch 1.000000000 -0.006113857 -0.013460802 0.008640181 0.013551891  
## customer\_type -0.006113857 1.000000000 0.037110365 -0.026797451 -0.020544234  
## gender -0.013460802 0.037110365 1.000000000 -0.067954892 0.015205909  
## product\_line 0.008640181 -0.026797451 -0.067954892 1.000000000 0.037893893  
## unit\_price 0.013551891 -0.020544234 0.015205909 0.037893893 1.000000000  
## quantity 0.001930628 -0.018705894 -0.076351656 -0.063649293 0.009800802  
## payment 0.025373513 -0.068185247 -0.048336870 0.010315646 -0.018116773  
## rating -0.049616876 0.017746989 0.003631188 0.023536164 -0.008367916  
## total 0.012931022 -0.020884334 -0.050733456 -0.019186236 0.633734080  
## quantity payment rating total  
## branch 0.001930628 0.02537351 -0.049616876 0.01293102  
## customer\_type -0.018705894 -0.06818525 0.017746989 -0.02088433  
## gender -0.076351656 -0.04833687 0.003631188 -0.05073346  
## product\_line -0.063649293 0.01031565 0.023536164 -0.01918624  
## unit\_price 0.009800802 -0.01811677 -0.008367916 0.63373408  
## quantity 1.000000000 0.01020392 -0.016105001 0.70504027  
## payment 0.010203918 1.00000000 0.012852398 0.01146344  
## rating -0.016105001 0.01285240 1.000000000 -0.03642915  
## total 0.705040267 0.01146344 -0.036429151 1.00000000

# Performing our graphical comparison  
# ---  
#   
  
library(stats)  
par(mfrow = c(1, 2))  
corrplot(correlationMatrix, order = "hclust")  
  
corrplot(cor(df\_low), order = "hclust")

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

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

## wrapper method

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

# Sequential forward greedy search (default)  
#  
out = clustvarsel(df\_low, G = 1:5)  
out

## ------------------------------------------------------   
## Variable selection for Gaussian model-based clustering  
## Stepwise (forward/backward) greedy search  
## ------------------------------------------------------   
##   
## Variable proposed Type of step BICclust Model G BICdiff Decision  
## total Add -13434.37 V 4 385.9196 Accepted  
## quantity Add -17219.45 VEE 5 518.5975 Accepted  
## unit\_price Add -22299.88 EVV 5 2796.2580 Accepted  
## unit\_price Remove -17219.45 VEE 5 2796.2580 Rejected  
## rating Add -26489.95 EVV 5 -266.0309 Rejected  
## unit\_price Remove -17219.45 VEE 5 2796.2580 Rejected  
##   
## Selected subset: total, quantity, unit\_price

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

## Embended methods

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

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

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

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

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

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

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

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

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

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

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

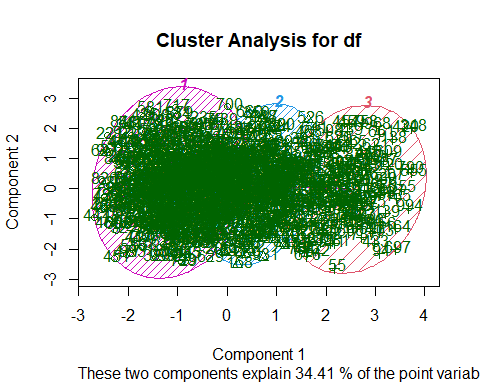
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter  
  
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter  
  
## Warning in segments(lx1, ly1, lx2, ly2, ...): "cor" is not a graphical parameter  
  
## Warning in segments(lx1, ly1, lx2,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

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

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

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

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



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

## branch customer\_type gender product\_line unit\_price quantity payment rating  
## 1 0 45.15 54.84 0 0 0 0 0  
## 2 0 43.39 56.60 0 0 0 0 0  
## 3 0 50.00 50.00 0 0 0 0 0  
## total  
## 1 0  
## 2 0  
## 3 0

# Part 3

library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

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

path <- "http://bit.ly/SupermarketDatasetII"  
  
Transactions<-read.transactions(path, sep = ",")

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

Transactions

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

# verifying the object class  
class(Transactions)

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

# Previewing our first 5 transactions  
inspect(Transactions[1:5])

## 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}

# preview the items that make up our dataset,  
# alternatively we can do the following  
# ---  
#   
items<-as.data.frame(itemLabels(Transactions))  
colnames(items) <- "Item"  
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

# Generating a summary of the transaction dataset  
# ---  
# This would give us some information such as the most purchased items,   
# distribution of the item sets (no. of items purchased in each transaction), etc.  
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

In the dataset, the most frequently bought item is Mineral water followed by eggs.

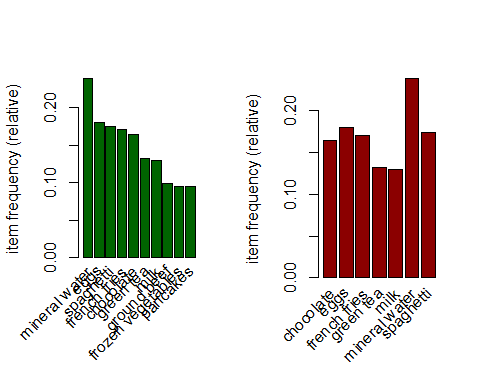
# Exploring the frequency of some articles  
   
itemFrequency(Transactions[, 8:10],type = "absolute")

## black tea blueberries body spray   
## 107 69 86

round(itemFrequency(Transactions[, 8:10],type = "relative")\*100,2)

## black tea blueberries body spray   
## 1.43 0.92 1.15

# Producing a chart of frequencies and filtering   
# to consider only items with a minimum percentage   
# of support/ considering a top x of items  
# ---  
# Displaying top 10 most common items in the transactions dataset   
# and the items whose relative importance is at least 10%  
#   
par(mfrow = c(1, 2))  
  
# plot the frequency of items  
itemFrequencyPlot(Transactions, topN = 10,col="darkgreen")  
itemFrequencyPlot(Transactions, support = 0.1,col="darkred")



# Building a model based on association rules   
# We use Min Support as 0.001 and confidence as 0.8  
rules <- apriori (Transactions, parameter = list(supp = 0.001, conf = 0.8))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.001 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 [0.00s].  
## sorting and recoding items ... [116 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 6 done [0.01s].  
## writing ... [74 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

rules

## set of 74 rules

Using a confidence level of 0.80 and support of 0.001 we have a model with 74 rules. An increase in minimum support will result in a decrease in the number of rules by the model. However, a slight decrease in the confidence level will result in a huge increase in the rules created by the models.

# Lets get more information on the rules formed  
# More statistical information such as support, lift and confidence is also provided.  
# ---  
#   
summary(rules)

## set of 74 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 3 4 5 6   
## 15 42 16 1   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.000 4.000 4.000 4.041 4.000 6.000   
##   
## summary of quality measures:  
## support confidence coverage lift   
## Min. :0.001067 Min. :0.8000 Min. :0.001067 Min. : 3.356   
## 1st Qu.:0.001067 1st Qu.:0.8000 1st Qu.:0.001333 1st Qu.: 3.432   
## Median :0.001133 Median :0.8333 Median :0.001333 Median : 3.795   
## Mean :0.001256 Mean :0.8504 Mean :0.001479 Mean : 4.823   
## 3rd Qu.:0.001333 3rd Qu.:0.8889 3rd Qu.:0.001600 3rd Qu.: 4.877   
## Max. :0.002533 Max. :1.0000 Max. :0.002666 Max. :12.722   
## count   
## Min. : 8.000   
## 1st Qu.: 8.000   
## Median : 8.500   
## Mean : 9.419   
## 3rd Qu.:10.000   
## Max. :19.000   
##   
## mining info:  
## data ntransactions support confidence  
## Transactions 7501 0.001 0.8

The set of 74 rules has a maximum rule length of 6 and a minimum of 3.

# lets take a peek at the first 5 rules of the associative model formed.   
  
inspect(rules[1:5])

## lhs rhs support confidence  
## [1] {frozen smoothie,spinach} => {mineral water} 0.001066524 0.8888889   
## [2] {bacon,pancakes} => {spaghetti} 0.001733102 0.8125000   
## [3] {nonfat milk,turkey} => {mineral water} 0.001199840 0.8181818   
## [4] {ground beef,nonfat milk} => {mineral water} 0.001599787 0.8571429   
## [5] {mushroom cream sauce,pasta} => {escalope} 0.002532996 0.9500000   
## coverage lift count  
## [1] 0.001199840 3.729058 8   
## [2] 0.002133049 4.666587 13   
## [3] 0.001466471 3.432428 9   
## [4] 0.001866418 3.595877 12   
## [5] 0.002666311 11.976387 19

The interpretation of this will require the understanding of several words. - Support -> How popular an itemset is, as measured by the proportion of transactions in which an itemset appears. - Confidence -> How often one item A appears whenever another item B appears in a transaction. This is usually a conditional probability. - Lift -> A rule with a lift of > 1 it would imply that those two occurrences are dependent on one another and useful for predicting.

Thus in the 5th rule with a confidence level ~ 0.95 means that it is very likely that these three items are bought together by every customer.

# So lets sort the rules by the conficence levels to see the items are mostly bought together  
  
  
rules<-sort(rules, by="confidence", decreasing=TRUE)  
inspect(rules[1:5])

## lhs rhs support   
## [1] {french fries,mushroom cream sauce,pasta} => {escalope} 0.001066524  
## [2] {ground beef,light cream,olive oil} => {mineral water} 0.001199840  
## [3] {cake,meatballs,mineral water} => {milk} 0.001066524  
## [4] {cake,olive oil,shrimp} => {mineral water} 0.001199840  
## [5] {mushroom cream sauce,pasta} => {escalope} 0.002532996  
## confidence coverage lift count  
## [1] 1.00 0.001066524 12.606723 8   
## [2] 1.00 0.001199840 4.195190 9   
## [3] 1.00 0.001066524 7.717078 8   
## [4] 1.00 0.001199840 4.195190 9   
## [5] 0.95 0.002666311 11.976387 19

The following rules with a confidence level of 1 means that the items are almost always bought in that combination. Therefore, the marketing division would have to find a way to create promotions on these items. For instance, a promotion campaign would be like buy french fries and get 50 percent off on Mushroom cream sauce.

# Part 4: Anomaly Detection

# Installing anomalize package  
# ---  
#   
#install.packages("anomalize")

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

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

## v tibble 3.1.4 v dplyr 1.0.7  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 2.0.1 v forcats 0.5.1  
## v purrr 0.3.4

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::arrange() masks plyr::arrange()  
## x dplyr::between() masks data.table::between()  
## x readr::col\_factor() masks scales::col\_factor()  
## x purrr::compact() masks plyr::compact()  
## x dplyr::count() masks plyr::count()  
## x purrr::discard() masks scales::discard()  
## x tidyr::expand() masks Matrix::expand()  
## x dplyr::failwith() masks plyr::failwith()  
## x dplyr::filter() masks tibbletime::filter(), stats::filter()  
## x dplyr::first() masks data.table::first()  
## x dplyr::id() masks plyr::id()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::last() masks data.table::last()  
## x latticeExtra::layer() masks ggplot2::layer()  
## x purrr::lift() masks caret::lift()  
## x purrr::map() masks mclust::map()  
## x dplyr::mutate() masks plyr::mutate()  
## x tidyr::pack() masks Matrix::pack()  
## x dplyr::recode() masks arules::recode()  
## x dplyr::rename() masks plyr::rename()  
## x dplyr::summarise() masks plyr::summarise()  
## x dplyr::summarize() masks plyr::summarize()  
## x purrr::transpose() masks data.table::transpose()  
## x tidyr::unpack() masks Matrix::unpack()

library(anomalize)

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

# load data and convert it to as\_tbl\_time  
anom <- read.csv('http://bit.ly/CarreFourSalesDataset')  
head(anom)

## Date Sales  
## 1 1/5/2019 548.9715  
## 2 3/8/2019 80.2200  
## 3 3/3/2019 340.5255  
## 4 1/27/2019 489.0480  
## 5 2/8/2019 634.3785  
## 6 3/25/2019 627.6165

First we have to format the Date column as date attribute.

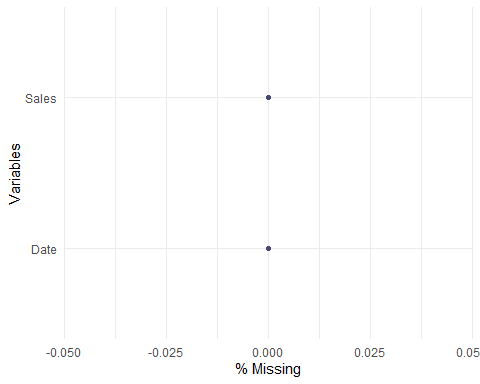
# conversion to date  
anom$Date <- as.Date(anom$Date , format = "%m/%d/%y")  
dim(anom)

## [1] 1000 2

For the Carrefour sales data, there are 1000 rows and 2 columns

library(naniar)  
gg\_miss\_var(anom, show\_pct = TRUE)

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please  
## use `guide = "none"` instead.



colSums(is.na(anom))

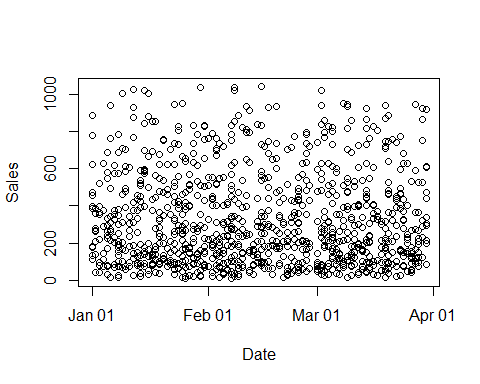
## Date Sales   
## 0 0

There are no missing values in the sales Data First lets convert the df to a different format.

anomX <- as\_tbl\_time(anom, Date)  
class(anomX)

## [1] "tbl\_time" "tbl\_df" "tbl" "data.frame"

plot (anomX)



#install.packages("devtools")  
#devtools::install\_github("twitter/AnomalyDetection")  
library(AnomalyDetection)

sales\_an <- AnomalyDetectionVec (x = anomX$Sales,period = 3 , direction= "both", plot = TRUE)



# Anomalize   
  
#anomX %>%  
# time\_decompose(dates) %>%  
# anomalize(remainder) %>%  
# time\_recompose() %>%  
# plot\_anomalies(time\_recomposed = TRUE, ncol = 3, alpha\_dots = 0.5)

# Conclusions

The data provided was accurate and more than sufficient to perform all the analysis that was initially intended for the project. The marketing team will find insight and leads on various topics such as: - product distribution. - marketing strategies and much more