Carrefour Kenya Data Analysis

Stacy Mwende

3/19/2020

## Context

You are a Data analyst at Carrefour Kenya and are currently undertaking a project that will inform the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax). Your project has been divided into three parts where you’ll explore a recent marketing dataset by performing various unsupervised learning techniques and later providing recommendations based on your insights.

Part 1: Dimensionality Reduction

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

Part 2: Feature Selection

This section requires you to perform feature selection through the use of the unsupervised learning methods learned earlier this week. You will be required to perform your analysis and provide insights on the features that contribute the most information to the dataset.

Part 3: Association Rules

This section will require that you create association rules that will allow you to identify relationships between variables in the dataset. You are provided with a separate dataset that comprises groups of items that will be associated with others. Just like in the other sections, you will also be required to provide insights for your analysis.

Part 4: Anomaly Detection

You have also been requested to check whether there are any anomalies in the given sales dataset. The objective of this task being fraud detection.

## Defining the question

Based on project data provided,inform the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax)

## Experimental design

Business understanding Data understanding Analysis Conclusion Recommendation

## Loading the libraries required for analysis

#install.packages("Rtsne")  
#library(Rtsne)  
library(lattice)  
#install.packages("dplyr")  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

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

#install.packages("tidyverse")  
library(tidyverse)

## -- Attaching packages -------------------------------------------------------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.1 v readr 1.3.1  
## v tibble 2.1.1 v purrr 0.3.3  
## v tidyr 0.8.3 v stringr 1.4.0  
## v ggplot2 3.1.1 v forcats 0.4.0

## -- Conflicts ----------------------------------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

#install.packages("ggplot2")  
library(ggplot2)  
#install.packages("devtools",dependencies=TRUE)  
library(devtools) #Load devtools before running ggbiplot otherwise will encounter install\_github error  
#install\_github("vqv/ggbiplot", force = TRUE) #For plotting PCA  
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 object is masked from 'package:purrr':  
##   
## compact

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

## 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

# Create a working directory  
path <- "C:/Users/comp5/Downloads/R\_Program"  
data <- file.path(path, "Supermarket\_Dataset\_1 - Sales Data.csv")  
  
# Reading the dataset  
Data1 <- read.csv(data)  
head(Data1)

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

## Exploring the dataset

**Number of rows and Columns**

dim(Data1)

## [1] 1000 16

We have 1000 entries and 16 columns

**Get the datatypes**

sapply(Data1, class)

## Invoice.ID Branch Customer.type   
## "factor" "factor" "factor"   
## Gender Product.line Unit.price   
## "factor" "factor" "numeric"   
## Quantity Tax Date   
## "integer" "numeric" "factor"   
## Time Payment cogs   
## "factor" "factor" "numeric"   
## gross.margin.percentage gross.income Rating   
## "numeric" "numeric" "numeric"   
## Total   
## "numeric"

**Check for null values**

sum(is.na(Data1))

## [1] 0

There no null values in the dataset

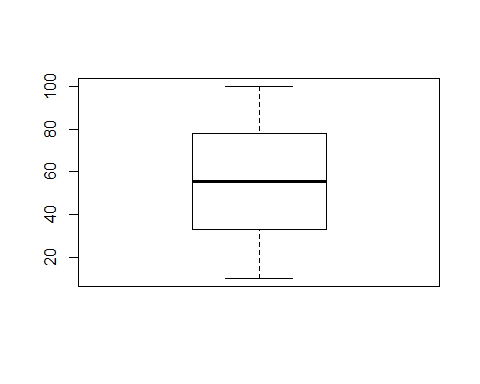
**Check for duplicates**

anyDuplicated(Data1)

## [1] 0

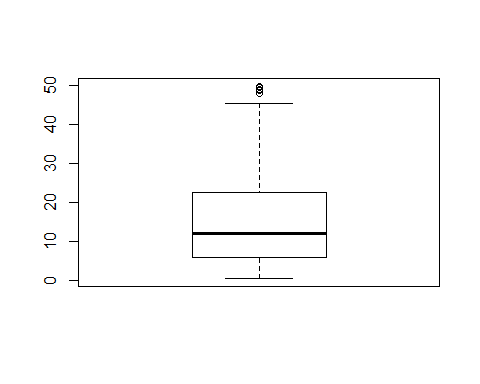
**Check for outliers** This is done on the numeric variables in the provided dataset.

#Unit Price  
boxplot(Data1$Unit.price)



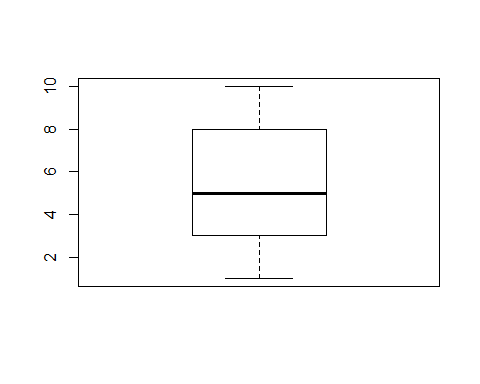
There are no outliers in the unit price dataset.

#Tax Column  
boxplot(Data1$Tax)



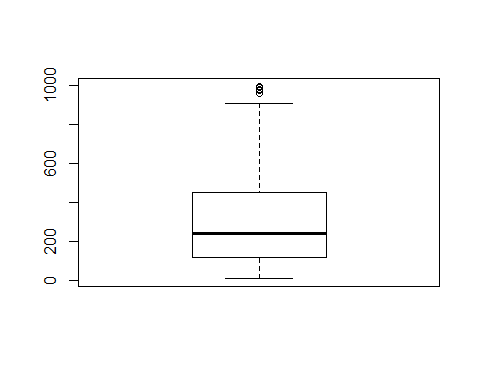
There are some outliers in the tax column which is allowed because different products have different tax rates based on the price of the product

#Quantity Column  
boxplot(Data1$Quantity)



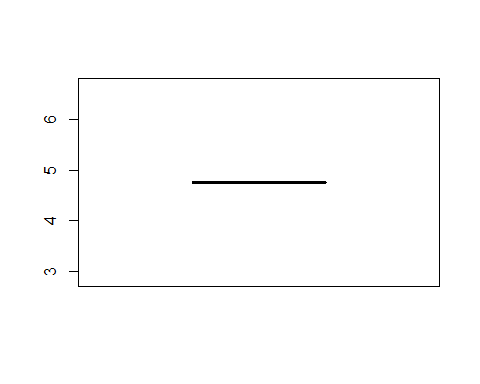
No outliers in the quantity column

#Cogs Column  
boxplot(Data1$cogs)



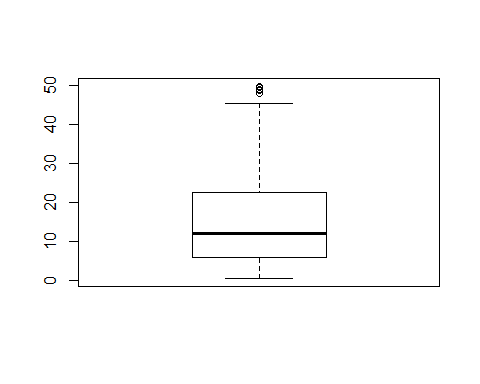
Cogs refers to the price of the item bought (Unit Price \* Quantity) There are some outliers in the specified column which comes in due to the different price rates of the different items sold.

# gross.margin.percentage Column  
boxplot(Data1$gross.margin.percentage)

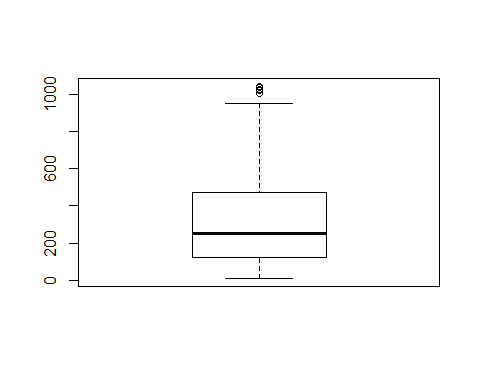


The above visulaization appeared like that due to the fact that the value provided was same all through the rest entries

#Gross income Column  
boxplot(Data1$gross.income)

 We observe some outliers in the gross income column which is explained by the different unit prices for the various products

#Total Column  
boxplot(Data1$Total)

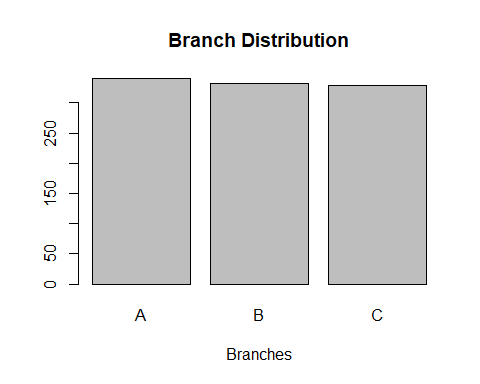


## Get unique entries in the non-numeric columns

# Branch column  
unique(Data1$Branch)

## [1] A C B  
## Levels: A B C

Branch <- Data1$Branch  
Branch\_freq <- table(Branch)  
barplot(Branch\_freq, main = "Branch Distribution", xlab = "Branches")

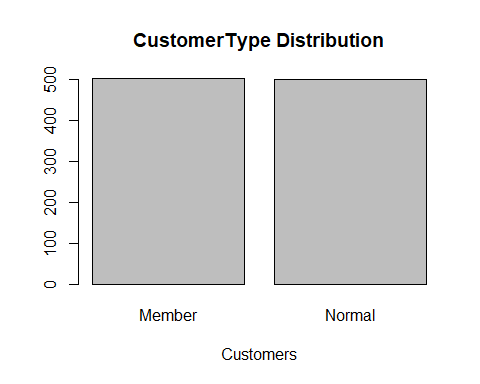


Branch B and C have almost the same count with a slight difference in branch A

# CustomerType column  
unique(Data1$Customer.type)

## [1] Member Normal  
## Levels: Member Normal

customer <- Data1$Customer.type  
customer\_freq <- table(customer)  
barplot(customer\_freq, main = "CustomerType Distribution", xlab = "Customers")

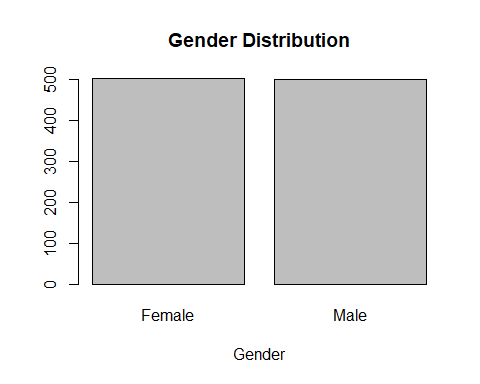


Walk-ins of both Member and Normal customers was same.

# Gender column  
unique(Data1$Gender)

## [1] Female Male   
## Levels: Female Male

Gender <- Data1$Gender  
Gender\_freq <- table(Gender)  
barplot(Gender\_freq, main = "Gender Distribution", xlab = "Gender")



# Branch column  
unique(Data1$Product.line)

## [1] Health and beauty Electronic accessories Home and lifestyle   
## [4] Sports and travel Food and beverages Fashion accessories   
## 6 Levels: Electronic accessories ... Sports and travel

product <- Data1$Product.line  
product\_freq <- table(product)  
product\_freq

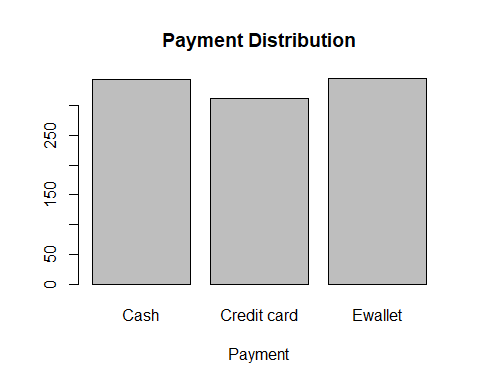
## product  
## Electronic accessories Fashion accessories Food and beverages   
## 170 178 174   
## Health and beauty Home and lifestyle Sports and travel   
## 152 160 166

Most customers where more on Fashion accessories followed by Food and beverages, Electronics, Sports and travel, Home and lifestyle , and Health and beauty respectively.

# Branch column  
unique(Data1$Payment)

## [1] Ewallet Cash Credit card  
## Levels: Cash Credit card Ewallet

payment <- Data1$Payment  
payment\_freq <- table(payment)  
barplot(payment\_freq, main = "Payment Distribution", xlab = "Payment")



Cash and Ewallet hold the same count of about 350 with credit card having a count of 300

#Encode the categorical columns to integers We prefer interger over numeric because interger can’t take decimals which is what we need for the specific columns.

# Branch column  
Data1$Branch<-as.integer(Data1$Branch)  
# Customer Type column  
Data1$Customer.type<-as.integer(Data1$Customer.type)  
# Gender column  
Data1$Gender<-as.integer(Data1$Gender)  
# Product.line column  
Data1$Product.line<-as.integer(Data1$Product.line)  
#Payment column  
Data1$Payment<-as.integer(Data1$Payment)

#install.packages("lubridate") #Date split package  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:plyr':  
##   
## here

## The following object is masked from 'package:base':  
##   
## date

# Convert to date datatype first then split thereafter  
Data1$Date <- as.Date(Data1$Date, "%m/%d/%Y")  
Data1$year <- year(ymd(Data1$Date))  
Data1$month <- month(ymd(Data1$Date))   
Data1$day <- day(ymd(Data1$Date))

## Dimensionality reduction

**Apply PCA Algorithm**

Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It’s often used to make data easy to explore and visualize.

#Extract numerical and integer columns only   
Data1\_df <- select\_if(Data1,is.numeric)  
str(Data1\_df)

## 'data.frame': 1000 obs. of 16 variables:  
## $ Branch : int 1 3 1 1 1 3 1 3 1 2 ...  
## $ Customer.type : int 1 2 2 1 2 2 1 2 1 1 ...  
## $ Gender : int 1 1 2 2 2 2 1 1 1 1 ...  
## $ Product.line : int 4 1 5 4 6 1 1 5 4 3 ...  
## $ Unit.price : num 74.7 15.3 46.3 58.2 86.3 ...  
## $ Quantity : int 7 5 7 8 7 7 6 10 2 3 ...  
## $ Tax : num 26.14 3.82 16.22 23.29 30.21 ...  
## $ Payment : int 3 1 2 3 3 3 3 3 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 ...  
## $ year : num 2019 2019 2019 2019 2019 ...  
## $ month : num 1 3 3 1 2 3 2 2 1 2 ...  
## $ day : int 5 8 3 27 8 25 25 24 10 20 ...

# Remove the constant/zero column to avaoid an error which comes about by including them  
names(Data1\_df[, sapply(Data1\_df, function(v) var(v, na.rm=TRUE)==0)])

## [1] "gross.margin.percentage" "year"

# Drop the columns as they result to error "stop("cannot rescale a constant/zero column to unit variance")"  
Data1\_df <- subset(Data1\_df, select = -c(gross.margin.percentage, year))

#The prcomp() function also provides the facility to compute standard deviation of each principal component  
# We then pass df to the prcomp(). We also set two arguments, center and scale,   
# to be TRUE then preview our object with summary  
# ---  
#   
Data1\_pca <- prcomp(Data1\_df, center = TRUE, scale. = TRUE)  
summary(Data1\_pca)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6  
## Standard deviation 2.2203 1.08227 1.06969 1.02580 1.00895 0.99359  
## Proportion of Variance 0.3521 0.08367 0.08173 0.07516 0.07271 0.07052  
## Cumulative Proportion 0.3521 0.43578 0.51751 0.59267 0.66538 0.73590  
## PC7 PC8 PC9 PC10 PC11 PC12  
## Standard deviation 0.97647 0.96596 0.94903 0.9057 0.29961 1.824e-16  
## Proportion of Variance 0.06811 0.06665 0.06433 0.0586 0.00641 0.000e+00  
## Cumulative Proportion 0.80401 0.87066 0.93499 0.9936 1.00000 1.000e+00  
## PC13 PC14  
## Standard deviation 1.518e-16 4.039e-18  
## Proportion of Variance 0.000e+00 0.000e+00  
## Cumulative Proportion 1.000e+00 1.000e+00

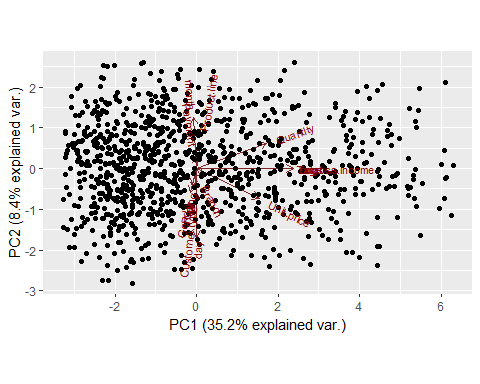
A principal component is a normalized linear combination of the original predictors in a data set. We have a total of 14 components since we removed 2 columns from the dataset PC1 captures the maximum variance in the data set which we observe in the dataset(35%) As we move across the PCA results above, the variability decreases as you move to the last component.

# Calling str() to have a look at your PCA object  
#   
str(Data1\_pca)

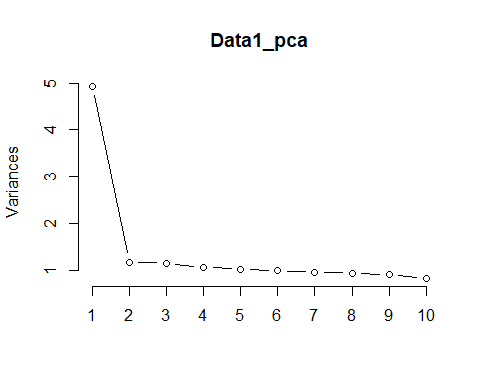
## List of 5  
## $ sdev : num [1:14] 2.22 1.08 1.07 1.03 1.01 ...  
## $ rotation: num [1:14, 1:14] 0.0226 -0.0126 -0.0283 0.0175 0.2912 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : chr [1:14] "Branch" "Customer.type" "Gender" "Product.line" ...  
## .. ..$ : chr [1:14] "PC1" "PC2" "PC3" "PC4" ...  
## $ center : Named num [1:14] 1.99 1.5 1.5 3.45 55.67 ...  
## ..- attr(\*, "names")= chr [1:14] "Branch" "Customer.type" "Gender" "Product.line" ...  
## $ scale : Named num [1:14] 0.818 0.5 0.5 1.715 26.495 ...  
## ..- attr(\*, "names")= chr [1:14] "Branch" "Customer.type" "Gender" "Product.line" ...  
## $ x : num [1:1000, 1:14] 2.03 -2.291 0.118 1.471 2.745 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : chr [1:1000] "1" "2" "3" "4" ...  
## .. ..$ : chr [1:14] "PC1" "PC2" "PC3" "PC4" ...  
## - attr(\*, "class")= chr "prcomp"

From the analysis above we see the center point (scale), standard deviation(sdev) of each principal component. We also see the relationship (correlation or anticorrelation, etc) between the initial variables and the principal components ($rotation).

ggbiplot(Data1\_pca, labels=rownames(Data1\_pca),ellipse = TRUE,obs.scale=1,var.scale=1)



plot(Data1\_pca, type="l")



The plot above shows that 10 components contribute most in the variance observed in the dataset provided with PC1 contributing the most.

## Feature Selection

Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. This becomes even more important when the number of features are very large. We can use the findCorrelation function included in the caret package to create a subset of variabes. This function would allows us to remove redundancy by correlation using the given dataset. It would search through a correlation matrix and return a vector of integers corresponding to the columns, to remove or reduce pair-wise correlations

suppressWarnings(  
 suppressMessages(if  
 (!require(caret, quietly=TRUE))  
 install.packages("caret")))  
library(caret)  
  
suppressWarnings(  
 suppressMessages(if  
 (!require(corrplot, quietly=TRUE))  
 install.packages("corrplot")))  
library(corrplot)  
  
# Calculating the correlation matrix  
#  
correlationMatrix <- cor(Data1\_df)  
#  
# Find attributes that are highly correlated  
#   
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75)  
#  
# Highly correlated attributes  
#   
highlyCorrelated

## [1] 7 9 10

names(Data1\_df[,highlyCorrelated])

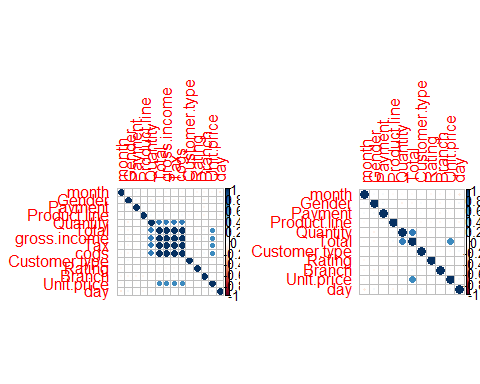
## [1] "Tax" "cogs" "gross.income"

These are the highly correlated features in the dataset

# We can remove the variables with a higher correlation to remove redudancy.  
Data2\_df <- Data1\_df[-highlyCorrelated]  
colnames(Data2\_df)

## [1] "Branch" "Customer.type" "Gender" "Product.line"   
## [5] "Unit.price" "Quantity" "Payment" "Rating"   
## [9] "Total" "month" "day"

# Performing our graphical comparison  
#   
par(mfrow = c(1, 2))  
corrplot(correlationMatrix, order = "hclust")  
corrplot(cor(Data2\_df), order = "hclust")



## Association rules

Association rules are algorithms used to find relationships between items using point-of-sale systems. These rules are used to predict the likelihood of different products being purchased together.

suppressWarnings(  
 suppressMessages(if  
 (!require(arules, quietly=TRUE))  
 install.packages("arules")))  
library(arules)  
  
library(arulesViz)

# Loading the dataset  
#  
# We will use read.transactions fuction which will load data from comma-separated files and convert them to the class transactions, which is the kind of data that we will require while working with models of association rules  
  
Rules\_df <- read.transactions("Supermarket\_Sales\_Dataset II.csv",sep=",")  
Rules\_df

path <- "C:/Users/comp5/Downloads/R\_Program"  
data <- file.path(path, "Supermarket\_Sales\_Dataset II.csv")  
Rules\_df <- read.csv(data)  
head(Rules\_df )

## shrimp almonds avocado vegetables.mix green.grapes  
## 1 burgers meatballs eggs   
## 2 chutney   
## 3 turkey avocado   
## 4 mineral water milk energy bar whole wheat rice green tea  
## 5 low fat yogurt   
## 6 whole wheat pasta french fries   
## whole.weat.flour yams cottage.cheese energy.drink tomato.juice  
## 1   
## 2   
## 3   
## 4   
## 5   
## 6   
## low.fat.yogurt green.tea honey salad mineral.water salmon  
## 1   
## 2   
## 3   
## 4   
## 5   
## 6   
## antioxydant.juice frozen.smoothie spinach olive.oil  
## 1 NA  
## 2 NA  
## 3 NA  
## 4 NA  
## 5 NA  
## 6 NA

#Check for number of rows and columns  
dim(Rules\_df)

## [1] 7500 20

The data has 7500 entries and 20 columns

# Look at the data structure of the dataset  
str(Rules\_df)

## 'data.frame': 7500 obs. of 20 variables:  
## $ shrimp : Factor w/ 115 levels "almonds","antioxydant juice",..: 15 27 108 72 65 112 98 49 43 37 ...  
## $ almonds : Factor w/ 118 levels "","almonds","antioxydant juice",..: 69 1 5 71 1 43 63 99 1 85 ...  
## $ avocado : Factor w/ 116 levels "","almonds","antioxydant juice",..: 36 1 1 37 1 1 93 53 1 1 ...  
## $ vegetables.mix : Factor w/ 115 levels "","almonds","antioxydant juice",..: 1 1 1 112 1 1 1 1 1 1 ...  
## $ green.grapes : Factor w/ 111 levels "","almonds","antioxydant juice",..: 1 1 1 51 1 1 1 1 1 1 ...  
## $ whole.weat.flour : Factor w/ 107 levels "","almonds","antioxydant juice",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ yams : Factor w/ 103 levels "","almonds","antioxydant juice",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ cottage.cheese : Factor w/ 99 levels ""," asparagus",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ energy.drink : Factor w/ 89 levels "","almonds","antioxydant juice",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ tomato.juice : Factor w/ 81 levels "","asparagus",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ low.fat.yogurt : Factor w/ 67 levels "","asparagus",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ green.tea : Factor w/ 51 levels "","blueberries",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ honey : Factor w/ 43 levels "","asparagus",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ salad : Factor w/ 29 levels "","babies food",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ mineral.water : Factor w/ 19 levels "","candy bars",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ salmon : Factor w/ 8 levels "","antioxydant juice",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ antioxydant.juice: Factor w/ 3 levels "","french fries",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ frozen.smoothie : Factor w/ 3 levels "","protein bar",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ spinach : Factor w/ 3 levels "","cereals","mayonnaise": 1 1 1 1 1 1 1 1 1 1 ...  
## $ olive.oil : logi NA NA NA NA NA NA ...

All of the 19 items are in factor datatype with Olive.oil appearing in logic form

# Check for null values per column in the dataset  
colSums(is.na(Rules\_df))

## shrimp almonds avocado vegetables.mix   
## 0 0 0 0   
## green.grapes whole.weat.flour yams cottage.cheese   
## 0 0 0 0   
## energy.drink tomato.juice low.fat.yogurt green.tea   
## 0 0 0 0   
## honey salad mineral.water salmon   
## 0 0 0 0   
## antioxydant.juice frozen.smoothie spinach olive.oil   
## 0 0 0 7500

Olive.oil is the only item with no entries whic means that the item was not purchased. Since all the data for the specific column is missing, we delete it.

# Drop olive oil column from the dataframe  
#  
Rules\_df$olive.oil <- NULL  
colnames(Rules\_df)

## [1] "shrimp" "almonds" "avocado"   
## [4] "vegetables.mix" "green.grapes" "whole.weat.flour"   
## [7] "yams" "cottage.cheese" "energy.drink"   
## [10] "tomato.juice" "low.fat.yogurt" "green.tea"   
## [13] "honey" "salad" "mineral.water"   
## [16] "salmon" "antioxydant.juice" "frozen.smoothie"   
## [19] "spinach"

We have a total of 19 column to work with.

# Do a summary to see how different items were purchased  
summary(Rules\_df)

## shrimp almonds avocado   
## mineral water : 577 :1754 :3112   
## burgers : 576 mineral water: 484 mineral water: 375   
## turkey : 458 spaghetti : 411 spaghetti : 279   
## chocolate : 391 eggs : 302 eggs : 225   
## frozen vegetables: 373 ground beef : 291 milk : 213   
## spaghetti : 354 french fries : 243 french fries : 180   
## (Other) :4771 (Other) :4015 (Other) :3116   
## vegetables.mix green.grapes whole.weat.flour  
## :4156 :4972 :5637   
## mineral water: 201 green tea : 153 french fries: 107   
## eggs : 181 eggs : 134 eggs : 102   
## french fries : 174 french fries: 130 green tea : 100   
## spaghetti : 167 chocolate : 115 chocolate : 71   
## milk : 149 milk : 114 pancakes : 69   
## (Other) :2472 (Other) :1882 (Other) :1414   
## yams cottage.cheese energy.drink   
## :6132 :6520 :6847   
## green tea : 96 green tea : 67 green tea : 57   
## french fries : 81 pancakes : 44 low fat yogurt : 38   
## pancakes : 69 low fat yogurt: 43 frozen smoothie: 35   
## eggs : 59 french fries : 40 french fries : 34   
## low fat yogurt: 55 chocolate : 38 fresh bread : 28   
## (Other) :1008 (Other) : 748 (Other) : 461   
## tomato.juice low.fat.yogurt green.tea   
## :7106 :7245 :7347   
## green tea : 31 low fat yogurt: 21 green tea : 14   
## french fries : 19 green tea : 20 french fries : 10   
## low fat yogurt: 17 fresh bread : 14 frozen smoothie: 10   
## tomato juice : 16 french fries : 12 low fat yogurt : 9   
## pancakes : 14 light mayo : 9 fresh bread : 7   
## (Other) : 297 (Other) : 179 (Other) : 103   
## honey salad mineral.water   
## :7414 :7454 :7476   
## green tea : 8 green tea : 4 magazines : 3   
## fresh bread : 6 french fries : 3 fresh bread : 2   
## low fat yogurt: 6 frozen smoothie: 3 green tea : 2   
## escalope : 4 cottage cheese : 2 low fat yogurt: 2   
## french fries : 4 eggplant : 2 pancakes : 2   
## (Other) : 58 (Other) : 32 (Other) : 13   
## salmon antioxydant.juice frozen.smoothie  
## :7493 :7497 :7497   
## antioxydant juice: 1 french fries : 1 protein bar: 2   
## cake : 1 frozen smoothie: 2 spinach : 1   
## chocolate : 1   
## frozen smoothie : 1   
## magazines : 1   
## (Other) : 2   
## spinach   
## :7498   
## cereals : 1   
## mayonnaise: 1   
##   
##   
##   
##

# Plot a frequency plot to see what items topped the list with more purchases  
Plot\_data <- as(Rules\_df, "transactions")  
# plot item frequency  
itemFrequencyPlot(Plot\_data,topN=20,type="absolute", col="darkgreen")

We observe sales of over 7000 for a couple of items such as Spinach,antioxydant juice, frozen smoothie, salmon, mineral water, salad,honey and green tea respectively. Shrimp mineral water and shrimp burgers were at the bottom of the list regarding the sales.

# Build a model based on association rules using the apriori function   
# We use Min Support as 0.001 and confidence as 0.8  
#   
rules <- apriori(Rules\_df, parameter = list(supp = 0.5, conf = 0.8, target = "rules",minlen=2))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.5 2  
## maxlen target ext  
## 10 rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 3750   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[1280 item(s), 7500 transaction(s)] done [0.06s].  
## sorting and recoding items ... [16 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 6 7 8 9 10

## Warning in apriori(Rules\_df, parameter = list(supp = 0.5, conf = 0.8,  
## target = "rules", : Mining stopped (maxlen reached). Only patterns up to a  
## length of 10 returned!

## done [0.02s].  
## writing ... [425218 rule(s)] done [0.14s].  
## creating S4 object ... done [0.28s].

# Get a summary of the rules  
summary(rules)

## set of 425218 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 2 3 4 5 6 7 8 9 10   
## 204 1478 6576 20134 45002 75943 98616 99417 77848   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 7.000 8.000 7.986 9.000 10.000   
##   
## summary of quality measures:  
## support confidence lift count   
## Min. :0.5541 Min. :0.8108 Min. :1.000 Min. :4156   
## 1st Qu.:0.5541 1st Qu.:1.0000 1st Qu.:1.001 1st Qu.:4156   
## Median :0.6629 Median :1.0000 Median :1.021 Median :4972   
## Mean :0.6455 Mean :0.9882 Mean :1.095 Mean :4841   
## 3rd Qu.:0.7516 3rd Qu.:1.0000 3rd Qu.:1.150 3rd Qu.:5637   
## Max. :0.9996 Max. :1.0000 Max. :1.508 Max. :7497   
##   
## mining info:  
## data ntransactions support confidence  
## Rules\_df 7500 0.5 0.8

# Observing rules built in our model i.e. first 10 model rules  
#   
inspect(rules[1:10])

## lhs rhs support confidence  
## [1] {vegetables.mix=} => {green.grapes=} 0.5541333 1.0000000   
## [2] {green.grapes=} => {vegetables.mix=} 0.5541333 0.8358809   
## [3] {vegetables.mix=} => {whole.weat.flour=} 0.5541333 1.0000000   
## [4] {vegetables.mix=} => {yams=} 0.5541333 1.0000000   
## [5] {vegetables.mix=} => {cottage.cheese=} 0.5541333 1.0000000   
## [6] {vegetables.mix=} => {energy.drink=} 0.5541333 1.0000000   
## [7] {vegetables.mix=} => {tomato.juice=} 0.5541333 1.0000000   
## [8] {vegetables.mix=} => {low.fat.yogurt=} 0.5541333 1.0000000   
## [9] {vegetables.mix=} => {green.tea=} 0.5541333 1.0000000   
## [10] {vegetables.mix=} => {honey=} 0.5541333 1.0000000   
## lift count  
## [1] 1.508447 4156   
## [2] 1.508447 4156   
## [3] 1.330495 4156   
## [4] 1.223092 4156   
## [5] 1.150307 4156   
## [6] 1.095370 4156   
## [7] 1.055446 4156   
## [8] 1.035197 4156   
## [9] 1.020825 4156   
## [10] 1.011600 4156

**Interpretation of the above results**

If a customer gets green.grapes, the possibility of picking vegetable mix is 84% A customer who picks green.grapes, energy drink is 83% more likely to pick vegetable mix

# We can order the rules by either confidence or support then view top 10 rules again  
rules<-sort(rules, by="confidence", decreasing=TRUE)  
inspect(rules[1:10])

## lhs rhs support confidence  
## [1] {vegetables.mix=} => {green.grapes=} 0.5541333 1   
## [2] {vegetables.mix=} => {whole.weat.flour=} 0.5541333 1   
## [3] {vegetables.mix=} => {yams=} 0.5541333 1   
## [4] {vegetables.mix=} => {cottage.cheese=} 0.5541333 1   
## [5] {vegetables.mix=} => {energy.drink=} 0.5541333 1   
## [6] {vegetables.mix=} => {tomato.juice=} 0.5541333 1   
## [7] {vegetables.mix=} => {low.fat.yogurt=} 0.5541333 1   
## [8] {vegetables.mix=} => {green.tea=} 0.5541333 1   
## [9] {vegetables.mix=} => {honey=} 0.5541333 1   
## [10] {vegetables.mix=} => {salad=} 0.5541333 1   
## lift count  
## [1] 1.508447 4156   
## [2] 1.330495 4156   
## [3] 1.223092 4156   
## [4] 1.150307 4156   
## [5] 1.095370 4156   
## [6] 1.055446 4156   
## [7] 1.035197 4156   
## [8] 1.020825 4156   
## [9] 1.011600 4156   
## [10] 1.006171 4156

There’s is 100% confidence that if a customer get vegetable.mix, he is likely to get green.grapes

## Anomaly Detection

An anomaly is a deviation from the norm, strange condition, situation or quality, an incongruity or inconsistency

#Loading packages  
#  
install.packages("anomalize")

## Installing package into 'C:/Users/comp5/Documents/R/win-library/3.5'  
## (as 'lib' is unspecified)

## package 'anomalize' successfully unpacked and MD5 sums checked  
##   
## The downloaded binary packages are in  
## C:\Users\comp5\AppData\Local\Temp\RtmpwP4nGs\downloaded\_packages

library(anomalize)  
  
library(factoextra)

## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ

suppressWarnings(  
 suppressMessages(if  
 (!require(tibble, quietly=TRUE))  
 install.packages("tibble")))  
library(tibble)  
  
  
suppressWarnings(  
 suppressMessages(if  
 (!require(dplyr, quietly=TRUE))  
 install.packages("dplyr")))  
library(dplyr)  
  
suppressWarnings(  
 suppressMessages(if  
 (!require(tibbletime, quietly=TRUE))  
 install.packages("tibbletime")))  
  
library(tibbletime)  
  
suppressWarnings(  
 suppressMessages(if  
 (!require(tidyverse, quietly=TRUE))  
 install.packages("tidyverse")))  
library(tidyverse)

#loading the dataset  
path <- "C:/Users/comp5/Downloads/R\_Program"  
data <- file.path(path, "Supermarket\_Sales\_Forecasting - Sales.csv")  
Anoma\_df <- read.csv(data)  
head(Anoma\_df)

## 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

# Check the data types of the 2 columns  
sapply(Anoma\_df, class)

## Date Sales   
## "factor" "numeric"

# convert Date datatype from factor to Date  
#  
Anoma\_df$Date <- as.Date(Anoma\_df$Date )  
sapply(Anoma\_df, class)

## Date Sales   
## "Date" "numeric"

#Convert the data frame to tibble  
#  
Anoma\_tb <- as\_tibble(Anoma\_df)  
head(Anoma\_tb)

## # A tibble: 6 x 2  
## Date Sales  
## <date> <dbl>  
## 1 0001-05-20 549.   
## 2 0003-08-20 80.2  
## 3 0003-03-20 341.   
## 4 NA 489.   
## 5 0002-08-20 634.   
## 6 NA 628.

Anoma\_tb <- Anoma\_tb %>%  
 tibbletime::as\_tbl\_time(index = Date)

Anoma\_tb %>%  
 time\_decompose(Date) %>%  
 anomalize(remainder) %>%  
 time\_recompose() %>%  
 plot\_anomalies(time\_recomposed = TRUE, ncol = 3, alpha\_dots = 0.5)