# R- Unsupervised learning- IP

Sharon Maswai

March 5, 2020

## Understanding the Question.

As a Data analyst at Carrefour Kenyayou are working on a project aimed at helping inform the strategies made by the marketing team. The project is divided into the following parts.

### Part 1: Dimensionality Reduction

Using PCA change reduce the dimension of the dataset to prinicipal components only. Data:http://bit.ly/CarreFourDataset

### Part 2: Feature Selection

Using unsupervised ;earning methods, perform feature selction to selection the most relevant variables based on correlation. Data: http://bit.ly/CarreFourDataset

### Part 3: Association Rules

Using association rules identify relationships between variables in the dataset and give insights for analysis done. Data: http://bit.ly/SupermarketDatasetII

### Part 4: Anormaly Detection

To detect where there are any anormalies in the records of the dataset. Data: http://bit.ly/CarreFourSalesDataset

#### Metric for success

- Basket analysis
- Anomaly Identification.

#### Data appropriateness

### Loading libraries

```r
# Importing the necessary R libraries
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

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

```r
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

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

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

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```r
options(warn = -1)


library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'

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

##
## Attaching package: 'arules'

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

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

```
library(arulesViz)
```

```
## Loading required package: grid

## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```
#library(grid)
#theme_set(theme_bw())
#options(warn = -1)
```

# Part One: PCA

## Loading Data

```
sales <- read.csv("~/Downloads/Supermarket_Dataset_1 - Sales Data (1).csv")
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
```

## Data Understanding

### 1. Data types

```
sapply(sales, 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"
```

```
dim(sales)
```

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

## Data cleaning

### 1. Checking and handling 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                    cogs
##                       0                       0                       0
```

```
## gross.margin.percentage            gross.income                  Rating
##                       0                       0                       0
##                   Total
##                       0
```

```
sales = na.omit(sales)

#confirming the drop of 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                    cogs
##                       0                       0                       0
## gross.margin.percentage            gross.income                  Rating
##                       0                       0                       0
##                   Total
##                       0
```

```
#remaining rows
nrow(sales)
```

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

**2. Handling duplicate values**

```
#Find the duplicated rows in the dataset

duplicates = sales[duplicated(sales),]

head(duplicates)
```

```
##  [1] Invoice.ID              Branch                  Customer.type
##  [4] Gender                  Product.line            Unit.price
##  [7] Quantity                Tax                     Date
## [10] Time                    Payment                 cogs
## [13] gross.margin.percentage gross.income            Rating
## [16] Total
## <0 rows> (or 0-length row.names)
```

```
#Remove the duplicated rows
sales = distinct(sales)
#emoving duplicated rows
nrow(sales)
```

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

### Confirminng dimensions

```
dim(sales)
```

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

### PCA application

```
# Convert categorical data into numerical

sales$Branch_Num<-as.integer(as.factor(sales$Branch))

sales$Customer_Type_Num<-as.integer(as.factor(sales$Customer.type))

sales$Gender_Numc<-as.integer(as.factor(sales$Gender))

sales$Product_Line_Num<-as.integer(as.factor(sales$Product.line))

sales$Payment_Num<-as.integer(as.factor(sales$Payment))
```

```
 #Split date year, month and day.
# Convert to date datatype first then split thereafter
```

```
sales$hour = format(strptime(sales$Time,"%H:%M"),'%H')
sales$minute = format(strptime(sales$Time,"%H:%M"),'%M')
```

```
#confirming data types
sapply(sales, 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               Branch_Num       Customer_Type_Num
##                "numeric"                "integer"                "integer"
##              Gender_Numc         Product_Line_Num              Payment_Num
##                "integer"                "integer"                "integer"
##                     hour                   minute
##              "character"              "character"
```

```
#num=sales[, c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)]

#normalize <- function(x){
 # return ((x-min(x)) / (max(x)-min(x)))
```

```
#}
#columns = c("Invoice.ID","Branch","Customer.type","Gender","Product.line","Unit.price", "Quantity", "T
#for (i in columns){
   # num[, i] = normalize(num[, i])
#}
```

```
#Subsetting numerical columns
data_num <- select_if(sales,is.numeric)
str(data_num)
```

```
## 'data.frame':    1000 obs. of  13 variables:
##  $ 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 ...
##  $ 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 ...
##  $ Branch_Num             : int  1 3 1 1 1 3 1 3 1 2 ...
##  $ Customer_Type_Num      : int  1 2 2 1 2 2 1 2 1 1 ...
##  $ Gender_Numc            : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ Product_Line_Num       : int  4 1 5 4 6 1 1 5 4 3 ...
##  $ Payment_Num            : int  3 1 2 3 3 3 3 3 2 2 ...
```

```
names(data_num[, sapply(data_num, function(v) var(v, na.rm=TRUE)==0)])
```

```
## NULL
```

```
data_num <- subset(data_num, select = -c(gross.margin.percentage))
#checking dimensions
dim(data_num)
```

```
## [1] 1000   12
```

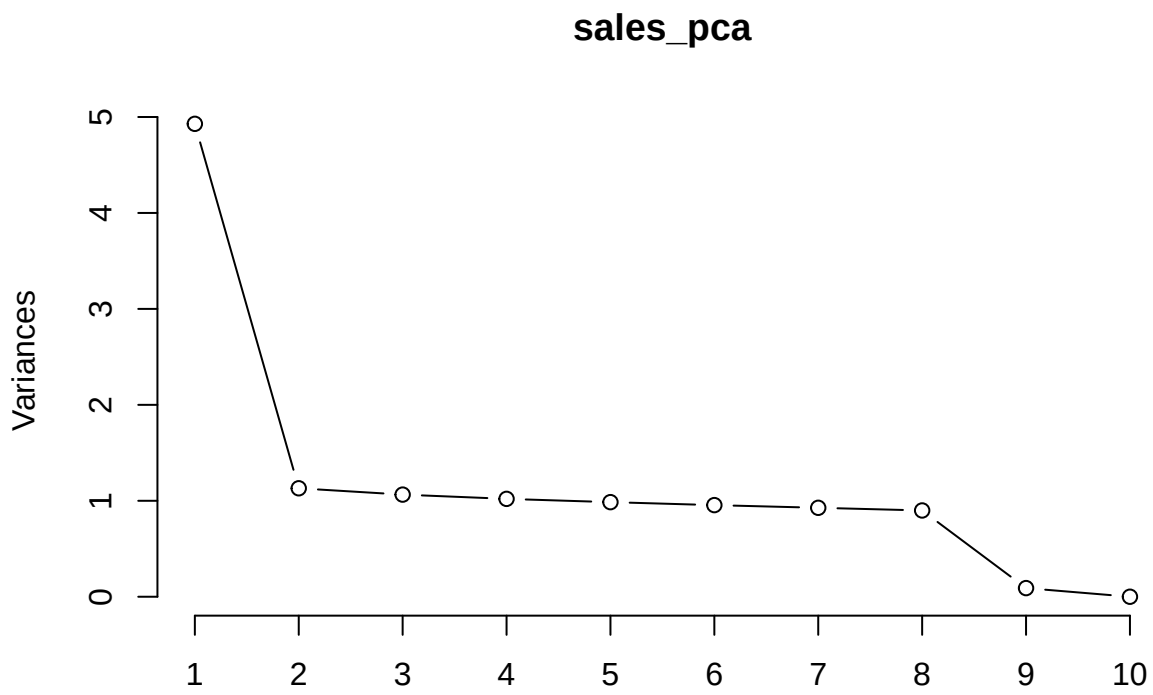
```
sales_pca <- prcomp(data_num, center = TRUE,scale. = TRUE)

summary(sales_pca)
```

```
## Importance of components:
##                           PC1     PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     2.2201 1.06317 1.0317 1.0099 0.99289 0.97714 0.96273
## Proportion of Variance 0.4107 0.09419 0.0887 0.0850 0.08215 0.07957 0.07724
## Cumulative Proportion  0.4107 0.50493 0.5936 0.6786 0.76078 0.84035 0.91758
##                           PC8     PC9      PC10      PC11      PC12
## Standard deviation     0.94823 0.29977 1.956e-16 1.395e-16 1.883e-17
## Proportion of Variance 0.07493 0.00749 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  0.99251 1.00000 1.000e+00 1.000e+00 1.000e+00
```

**Conclusions:**

- There are 12 pricipal components in this dataset ans each of these explain the total variance.
- 41% of the variance can be explained by PC1.

```
plot(sales_pca, type="l")
```




## Part two: Feature Selection

```
#Checking corelation
corr_sales <- cor(data_num)
```

```
#subsetting highly corelated features
Corr_high<- findCorrelation(corr_sales, cutoff=0.70)
```

```
Corr_high
```

```
## [1] 4 7 3 5
```

```
names(data_num[,Corr_high])
```

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

```
New_sales<-data_num[-Corr_high]

names(New_sales)
```

```
## [1] "Unit.price"        "Quantity"          "Rating"
## [4] "Branch_Num"        "Customer_Type_Num" "Gender_Numc"
## [7] "Product_Line_Num"  "Payment_Num"
```

# Part three: Association rules

```
sales <- read.csv("~/Downloads/Supermarket_Sales_Dataset II.csv")
head(sales)
```

```
##              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 low.fat.yogurt
## 1
## 2
## 3
## 4
## 5
## 6
##   green.tea honey salad mineral.water salmon antioxydant.juice frozen.smoothie
## 1
## 2
## 3
## 4
## 5
## 6
##   spinach olive.oil
## 1                NA
## 2                NA
## 3                NA
## 4                NA
## 5                NA
## 6                NA
```

# Data Understanding

### 1. Data types

```
sapply(sales, class)
```

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

```
dim(sales)
```

```
## [1] 7500   20
```

#Data cleaning ### 1. Checking and handling missing values

```
colSums(is.na(sales))
```

```
##            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
```

### 2. Handling duplicate values

```
#Remove the duplicated rows
sales = distinct(sales)
#emoving duplicated rows
nrow(sales)
```

```
## [1] 5175
```

```
summary(sales)
```

```
##                shrimp                   almonds              avocado
##  burgers          : 511   mineral water    : 451               : 867
##  turkey           : 420   spaghetti        : 377   mineral water: 372
##  mineral water    : 377   ground beef      : 287   spaghetti    : 278
##  frozen vegetables: 333   frozen vegetables: 231   eggs         : 217
##  shrimp           : 300   eggs             : 211   milk         : 209
##  spaghetti        : 272   chocolate        : 197   chocolate    : 171
##  (Other)          :2962   (Other)          :3421   (Other)      :3061
##        vegetables.mix       green.grapes      whole.weat.flour
##               :1843               :2647               :3312
##  mineral water: 201   green tea   : 153   french fries: 107
##  eggs         : 181   eggs        : 134   eggs        : 102
##  french fries : 173   french fries: 130   green tea   : 100
##  spaghetti    : 166   chocolate   : 115   chocolate   :  71
##  milk         : 149   milk        : 114   pancakes    :  69
##  (Other)      :2462   (Other)     :1882   (Other)     :1414
##              yams             cottage.cheese          energy.drink
##                :3807                 :4195                 :4522
##  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
##                :4781                 :4920                 :5022
##  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
##                :5089                 :5129                 :5151
##  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
##                   :5168                  :5172               :5172
##  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     olive.oil
##           :5173   Mode:logical
##  cereals   :   1   NA's:5175
##  mayonnaise:   1
##
##
```

```
##
##
```

**Showing the head of dataset**

```
head(sales)
```

```
##              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 low.fat.yogurt
## 1
## 2
## 3
## 4
## 5
## 6
##   green.tea honey salad mineral.water salmon antioxydant.juice frozen.smoothie
## 1
## 2
## 3
## 4
## 5
## 6
##   spinach olive.oil
## 1                NA
## 2                NA
## 3                NA
## 4                NA
## 5                NA
## 6                NA
```

**Applying the rules**

```
a_rules <- apriori(sales, 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: 2587
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1280 item(s), 5175 transaction(s)] done [0.02s].
## sorting and recoding items ... [15 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 done [0.01s].
## writing ... [203730 rule(s)] done [0.03s].
## creating S4 object  ... done [0.05s].
```

### Details on rules

```
summary(a_rules)
```

```
## set of 203730 rules
##
## rule length distribution (lhs + rhs):sizes
##     2     3     4     5     6     7     8     9    10
##   174  1163  4756 13309 26984 40908 47136 41502 27798
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   7.000   8.000   7.688   9.000  10.000
##
## summary of quality measures:
##     support         confidence          lift           count
##  Min.   :0.5115   Min.   :0.8109   Min.   :1.000   Min.   :2647
##  1st Qu.:0.5115   1st Qu.:1.0000   1st Qu.:1.001   1st Qu.:2647
##  Median :0.6400   Median :1.0000   Median :1.017   Median :3312
##  Mean   :0.6155   Mean   :0.9943   Mean   :1.089   Mean   :3185
##  3rd Qu.:0.7357   3rd Qu.:1.0000   3rd Qu.:1.144   3rd Qu.:3807
##  Max.   :0.9994   Max.   :1.0000   Max.   :1.562   Max.   :5172
##
## mining info:
##   data ntransactions support confidence
##  sales          5175     0.5        0.8
```

### Obtaining the rules

```
inspect(a_rules[1:20])
```

```
##      lhs                    rhs                   support   confidence lift
## [1]  {green.grapes=}     => {whole.weat.flour=}  0.5114976 1.0000000  1.562500
## [2]  {green.grapes=}     => {yams=}              0.5114976 1.0000000  1.359338
## [3]  {green.grapes=}     => {cottage.cheese=}    0.5114976 1.0000000  1.233611
## [4]  {green.grapes=}     => {energy.drink=}      0.5114976 1.0000000  1.144405
## [5]  {green.grapes=}     => {tomato.juice=}      0.5114976 1.0000000  1.082410
## [6]  {green.grapes=}     => {low.fat.yogurt=}    0.5114976 1.0000000  1.051829
```

```
## [7]  {green.grapes=}     => {green.tea=}         0.5114976 1.0000000  1.030466
## [8]  {green.grapes=}     => {honey=}             0.5114976 1.0000000  1.016899
## [9]  {green.grapes=}     => {salad=}             0.5114976 1.0000000  1.008969
## [10] {green.grapes=}     => {mineral.water=}     0.5114976 1.0000000  1.004659
## [11] {green.grapes=}     => {salmon=}            0.5114976 1.0000000  1.001354
## [12] {green.grapes=}     => {frozen.smoothie=}   0.5114976 1.0000000  1.000580
## [13] {green.grapes=}     => {antioxydant.juice=} 0.5114976 1.0000000  1.000580
## [14] {green.grapes=}     => {spinach=}           0.5114976 1.0000000  1.000387
## [15] {whole.weat.flour=} => {yams=}              0.6400000 1.0000000  1.359338
## [16] {yams=}             => {whole.weat.flour=}  0.6400000 0.8699764  1.359338
## [17] {whole.weat.flour=} => {cottage.cheese=}    0.6400000 1.0000000  1.233611
## [18] {whole.weat.flour=} => {energy.drink=}      0.6400000 1.0000000  1.144405
## [19] {whole.weat.flour=} => {tomato.juice=}      0.6400000 1.0000000  1.082410
## [20] {whole.weat.flour=} => {low.fat.yogurt=}    0.6400000 1.0000000  1.051829
##      count
## [1]  2647
## [2]  2647
## [3]  2647
## [4]  2647
## [5]  2647
## [6]  2647
## [7]  2647
## [8]  2647
## [9]  2647
## [10] 2647
## [11] 2647
## [12] 2647
## [13] 2647
## [14] 2647
## [15] 3312
## [16] 3312
## [17] 3312
## [18] 3312
## [19] 3312
## [20] 3312
```

**Conclusion**

- There is 100% confidence in someone purchases whole wheat flour with green grapes.
- There is an 86% chance that when a customer buys yam they but whole wheat flour as well.

# Part four: Anormally Detection

```
anom=read.csv("~/Downloads/Supermarket_Sales_Forecasting - Sales (1).csv")
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
```

### Importing libraries

```
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 </>
```

```
library(coindeskr)
```