Part 3

Wilkister Mbaka

2022-07-30

Part 3: Association Rules

Specifying the Question

- Create association rules that will allow you to identify relationships between variables in the dataset.
- Provide insights for your analysis.

```r
# Load Package
library(arules)
```

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

```
##
## Attaching package: 'arules'
```

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

Reading the Data

```r
# Load Dataset
path <- "http://bit.ly/SupermarketDatasetII"
trans<-read.transactions(path, sep = ",")
```

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

```r
trans
```

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

Checking the Data

```r
# check info on the data
trans
```

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

```r
# verifying the object class
class(trans)
```

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

```r
# Previewing our first 5 transactions
inspect(trans[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}
```

```r
# preview the items that make up our dataset,
# alternatively we can do the following
# ---
#
items<-as.data.frame(itemLabels(trans))
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
```

```r
# 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(trans)
```

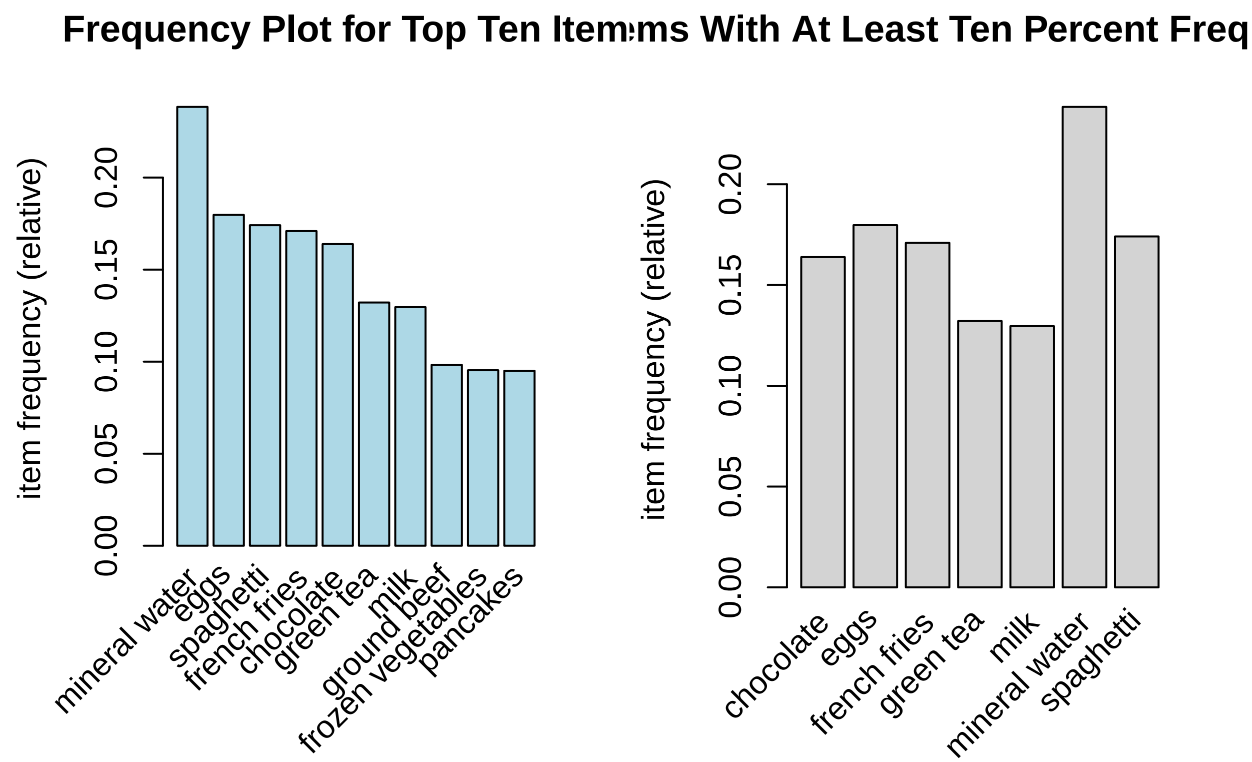
```
## transactions as itemMatrix in sparse format with
##  7501 rows (elements/itemsets/transactions) and
##  119 columns (items) and a density of 0.03288973
##
## most frequent items:
## mineral water          eggs     spaghetti  french fries     chocolate
##          1788          1348          1306          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
```

The top 5 most frequently bought items are mineral water, eggs, spaghetti, french fries and chocolate

```r
# Plot bar charts to visualize the frequencies of the most frequent items
# options(repr.plot.width = 15, repr.plot.height = 8)

par(mfrow = c(1, 2))

# plot the frequency of items
itemFrequencyPlot(trans, topN = 10,col="lightblue", main = "Frequency Plot for Top Ten Items")
itemFrequencyPlot(trans, support = 0.1,col="lightgray", main = "Items With At Least Ten Percent Frequen
```




```r
# find the 10 least popular items
least_items = itemFrequency(trans, type = "relative")
head(sort(least_items), 10)
```

```
##     water spray        napkins           cream         bramble             tea
##    0.0003999467   0.0006665778    0.0009332089    0.0018664178    0.0038661512
##         chutney mashed potato chocolate bread    dessert wine         ketchup
##    0.0041327823   0.0041327823    0.0042660979    0.0043994134    0.0043994134
```

The top 5 least frequently bought items are water spray, napkins, cream, bramble and tea

Building a Model

```r
# Building a model based on association rules
# We use Min Support as 0.001 and confidence as 0.8
rules <- apriori (trans, 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 [0.01s].
## 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
##  trans          7501   0.001        0.8
##                                                                       call
##  apriori(data = trans, parameter = list(supp = 0.001, conf = 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[1:10])
```

```
##      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
## [6]  {milk, pasta}                 => {shrimp}        0.001599787 0.8571429
## [7]  {cooking oil, fromage blanc}  => {mineral water} 0.001199840 0.8181818
## [8]  {black tea, salmon}           => {mineral water} 0.001066524 0.8000000
## [9]  {black tea, frozen smoothie}  => {milk}          0.001199840 0.8181818
## [10] {red wine, tomato sauce}      => {chocolate}     0.001066524 0.8000000
##      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
## [6]  0.001866418 11.995203 12
## [7]  0.001466471  3.432428  9
## [8]  0.001333156  3.356152  8
## [9]  0.001466471  6.313973  9
## [10] 0.001333156  4.882669  8
```

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.

The results reveal that the model is 95% confident that aperson buying mushroom cream sauce and pasta will buy escalope, 75% confident that a person buying milk and pasta will buy shrimp, etc,.

```r
# 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[1:10])
```

```
##      lhs                           rhs                 support confidence    coverage      lift count
## [1]  {french fries,
##       mushroom cream sauce,
##       pasta}                    => {escalope}      0.001066524  1.0000000 0.001066524 12.606723     8
## [2]  {ground beef,
##       light cream,
##       olive oil}                => {mineral water} 0.001199840  1.0000000 0.001199840  4.195190     9
## [3]  {cake,
##       meatballs,
##       mineral water}            => {milk}          0.001066524  1.0000000 0.001066524  7.717078     8
## [4]  {cake,
##       olive oil,
##       shrimp}                   => {mineral water} 0.001199840  1.0000000 0.001199840  4.195190     9
## [5]  {mushroom cream sauce,
##       pasta}                    => {escalope}      0.002532996  0.9500000 0.002666311 11.976387    19
## [6]  {red wine,
##       soup}                     => {mineral water} 0.001866418  0.9333333 0.001999733  3.915511    14
## [7]  {eggs,
##       mineral water,
##       pasta}                    => {shrimp}        0.001333156  0.9090909 0.001466471 12.722185    10
## [8]  {herb & pepper,
##       mineral water,
##       rice}                     => {ground beef}   0.001333156  0.9090909 0.001466471  9.252498    10
## [9]  {ground beef,
##       pancakes,
##       whole wheat rice}         => {mineral water} 0.001333156  0.9090909 0.001466471  3.813809    10
## [10] {frozen vegetables,
##       milk,
##       spaghetti,
##       turkey}                   => {mineral water} 0.001199840  0.9000000 0.001333156  3.775671     9
```

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.

There are 4 rules with 100% confidence

For instance, a promotion campaign would be like buy french fries and get 50 percent off on Mushroom cream sauce.

```r
# If we're interested in making a promotion relating to the sale of shrimp,
# we could create a subset of rules concerning these products
# This would tell us the items that the customers bought before purchasing shrimp

# If we wanted to determine the items that customers buying shrimps might buy

# Subset the rules
shrimp <- subset(rules, subset = lhs %pin% "shrimp")

# Order by confidence
```

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

# inspect top 5
inspect(shrimp[1:5])
```

```
##     lhs                        rhs                 support confidence    coverage     lift count
## [1] {cake,
##      olive oil,
##      shrimp}                => {mineral water} 0.001199840  1.0000000 0.001199840 4.195190     9
## [2] {chocolate,
##      frozen vegetables,
##      olive oil,
##      shrimp}                => {mineral water} 0.001199840  0.9000000 0.001333156 3.775671     9
## [3] {light cream,
##      mineral water,
##      shrimp}                => {spaghetti}     0.001066524  0.8888889 0.001199840 5.105326     8
## [4] {ground beef,
##      salmon,
##      shrimp}                => {spaghetti}     0.001066524  0.8888889 0.001199840 5.105326     8
## [5] {escalope,
##      french fries,
##      shrimp}                => {chocolate}     0.001066524  0.8888889 0.001199840 5.425188     8
```

Recommendations Shrimps could be bundled up with cake, olive oil, or with light cream, mineral water, etc, during the promotion season

Part 4: Anomaly Detection

Specifying the Question

- Check whether there are any anomalies in the given sales dataset. The objective of this task being fraud detection.

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

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x dplyr::recode() masks arules::recode()
## x tidyr::unpack() masks Matrix::unpack()
```

```r
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 </>
```

```r
library(tibbletime)
```

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

```r
library(timetk)
```

```r
# 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.

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

```
##         Date    Sales
## 1 2020-01-05 548.9715
## 2 2020-03-08  80.2200
## 3 2020-03-03 340.5255
## 4 2020-01-27 489.0480
## 5 2020-02-08 634.3785
## 6 2020-03-25 627.6165
```

```r
# Check dimensionality of the data
dim(anom)
```

```
## [1] 1000    2
```

There are 1000 rows and 2 columns in the dataset

```r
# Check for duplicates in the dataset
anyDuplicated(anom)
```

```
## [1] 0
```

There are no duplicates in the dataset

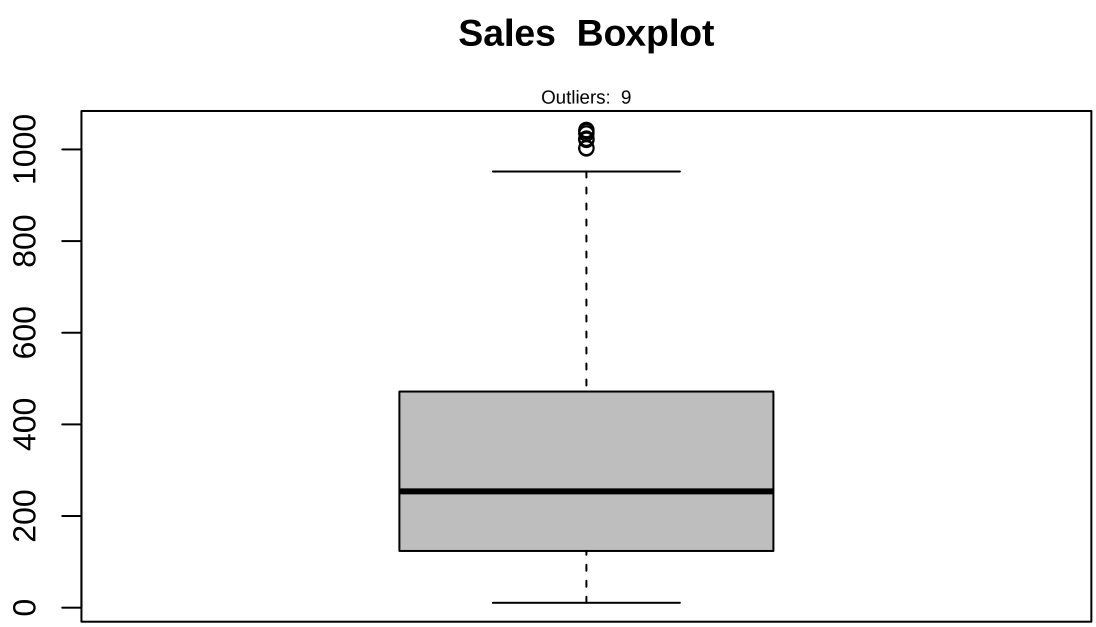
```r
# Check for missing values
colSums(is.na(anom))
```

```
##  Date Sales
##     0     0
```

There are no missing values in the dataset

```r
# Plotting boxplots to check for outliers
boxplot(anom$Sales,col='grey', main = 'Sales  Boxplot')

# display the number of outlier values in the column
outlier_sales <- boxplot.stats(anom$Sales)$out
mtext(paste("Outliers: ", paste(length(outlier_sales), collapse=", ")), cex=0.6)
```



There are 9 outliers. We will not be dropping the outliers because they represent actual goods sold

```r
# check for anomalies in the 'branch' column by scrutinizing its unique values
print(unique(anom$Date))
```

```
##  [1] "2020-01-05" "2020-03-08" "2020-03-03" "2020-01-27" "2020-02-08"
##  [6] "2020-03-25" "2020-02-25" "2020-02-24" "2020-01-10" "2020-02-20"
## [11] "2020-02-06" "2020-03-09" "2020-02-12" "2020-02-07" "2020-03-29"
## [16] "2020-01-15" "2020-03-11" "2020-01-01" "2020-01-21" "2020-03-05"
## [21] "2020-03-15" "2020-02-17" "2020-03-02" "2020-03-22" "2020-03-10"
## [26] "2020-01-25" "2020-01-28" "2020-01-07" "2020-03-23" "2020-01-17"
## [31] "2020-02-02" "2020-03-04" "2020-03-16" "2020-02-27" "2020-02-10"
## [36] "2020-03-19" "2020-02-03" "2020-03-07" "2020-02-28" "2020-03-27"
## [41] "2020-01-20" "2020-03-12" "2020-02-15" "2020-03-06" "2020-02-14"
## [46] "2020-03-13" "2020-01-24" "2020-01-06" "2020-02-11" "2020-01-22"
## [51] "2020-01-13" "2020-01-09" "2020-01-12" "2020-01-26" "2020-01-23"
## [56] "2020-02-23" "2020-01-02" "2020-02-09" "2020-03-26" "2020-03-01"
## [61] "2020-02-01" "2020-03-28" "2020-03-24" "2020-02-05" "2020-01-19"
## [66] "2020-01-16" "2020-01-08" "2020-02-18" "2020-01-18" "2020-02-16"
## [71] "2020-02-22" "2020-01-29" "2020-01-04" "2020-03-30" "2020-01-30"
## [76] "2020-01-03" "2020-03-21" "2020-02-13" "2020-01-14" "2020-03-18"
## [81] "2020-03-20" "2020-02-21" "2020-01-31" "2020-01-11" "2020-02-26"
## [86] "2020-03-17" "2020-03-14" "2020-02-04" "2020-02-19"
```

There are no anomalies in the date column

Univariate Analysis

```r
# identify numerical variables in the data1frame
nums <- unlist(lapply(anom, is.numeric))

# create a subset that contains the numerical variables
numerics <- subset(anom, select=nums)
```

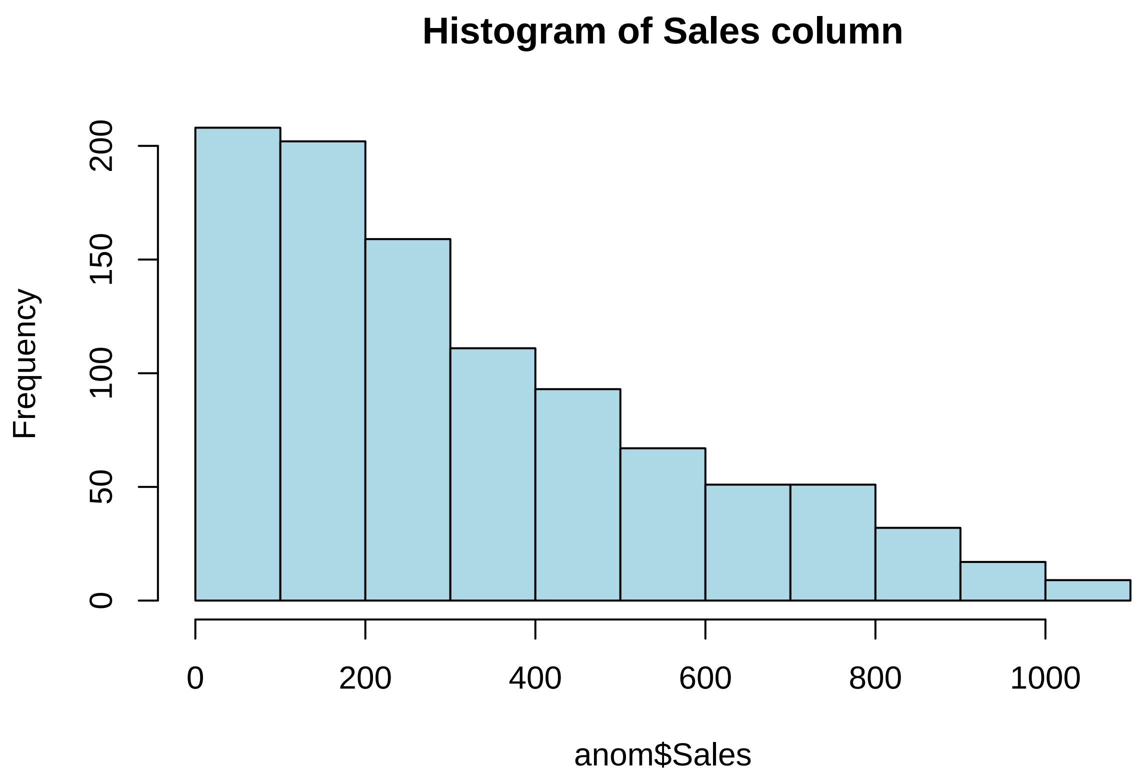
```r
# compute the measures of cenral tendancy and the measures of dispersion of the numerical variables and
library(moments)

statistics <- data.frame(
  Mean = apply(numerics, 2, mean),
  Median = apply(numerics, 2, median),
  Min = apply(numerics, 2, min),
  Max = apply(numerics, 2, max),
  Variance= apply(numerics, 2, var),
  Std = apply(numerics, 2, sd),
  Skewness = apply(numerics, 2, skewness),
  Kurtosis = apply(numerics, 2, kurtosis))

# round off the values to 2 decimal places and display the data1frame
statistics <- round(statistics, 2)
statistics
```

```
##         Mean Median   Min     Max Variance    Std Skewness Kurtosis
## Sales 322.97 253.85 10.68 1042.65  60459.6 245.89     0.89     2.91
```

```r
hist(anom$Sales, main = 'Histogram of Sales column',col="lightblue")
```



The data is left skewed and as the amount of sales increases the amount of goods bought reduces

```r
# Check the range of dates of our dataset
paste(c('Earliest:'), min(anom$Date))
```

```
## [1] "Earliest: 2020-01-01"
```

```r
paste(c('Latest:'), max(anom$Date))
```

```
## [1] "Latest: 2020-03-30"
```

The dataset has data from January 1st 2020 to March 30th 2020. So three months of data.

```r
# Check the range of Sales of our dataset
paste(c('Earliest:'), min(anom$Sales))
```

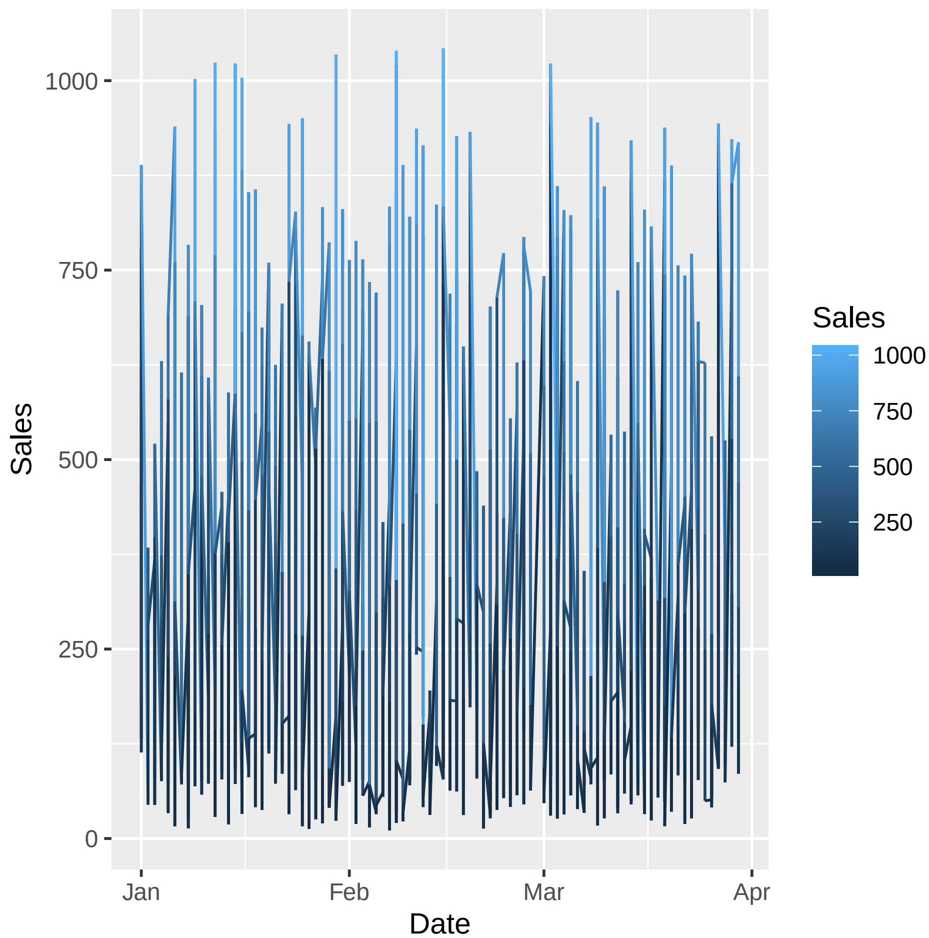
```
## [1] "Earliest: 10.6785"
```

```r
paste(c('Latest:'), max(anom$Sales))
```

```
## [1] "Latest: 1042.65"
```

The minimum sales is 10.67 and the maximum sale is 1042.65

```
#Plotting the data
library(ggplot2)
ggplot(anom, aes(x=Date, y=Sales, color=Sales)) + geom_line()
```



Anomaly Detection

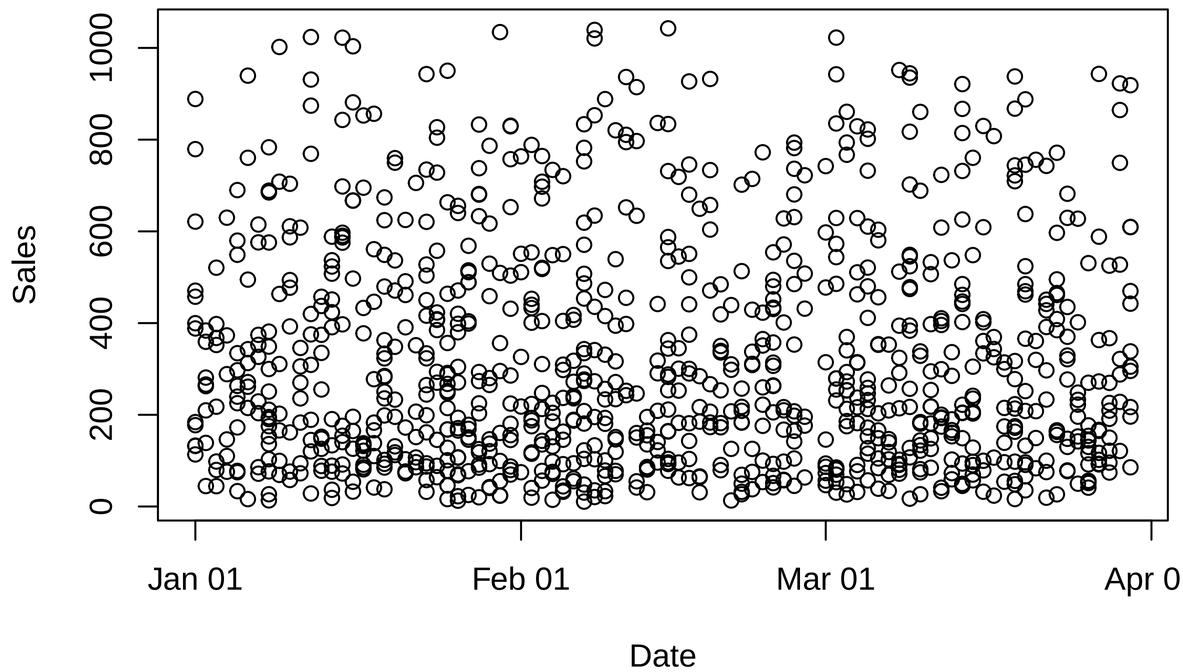
First lets convert the df to a different format.

```
# sort the table in ascending order of 'date'
anom  = anom[order(anom$Date),]

# convert dataset to tibble
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)
```

```r
sales_an
```

```
## $anoms
## data frame with 0 columns and 0 rows
##
## $plot
## NULL
```

```r
# 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