Core III WK 3

Cindy Gachuhi

04/02/2022

R Week 2 IP

Defining 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). 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.

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.

Loading and cleaning the datasets

Dataset 1

```
# Let us import the library 'Data Table'
#
library("data.table")
cf_data <- fread('http://bit.ly/CarreFourDataset')
```

```r
# preview the first 6 values
head(cf_data)
```

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

```r
#Checking our dataset
rows <- nrow(cf_data)
cols <- ncol(cf_data)

rows
```

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

```r
cols
```

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

```r
#1000 rows and 16 columns. Let us clean it.
```

####Removing null values

```r
# using the function 'colSums', we can identify the total number of missing values in each column
#
colSums(is.na(cf_data))
```

```
##               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
```

```
# No null values!
```

```
# To get rid of duplicates, we will identify the unique values from our dataset and assign them to vari
unique_cf <- cf_data[!duplicated(cf_data), ]

# seeing what these unique items are...
#unique_cf

# let us confirm whether all the duplicates have been removed
rows <- nrow(unique_cf)
rows
```

Removing duplicates

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

```
# Since our records have remained the same, it means we didn't have any duplicates.
```

~ # Part 1: Dimensionality Reduction

```
# One hot encoding our branch column
unique(unique_cf$Branch)
```

```
## [1] "A" "C" "B"
```

```
unique_cf$Branch <- factor(unique_cf$Branch,
levels <- c("A", "B", "C"),
labels <- c(1,2,3))
unique(unique_cf$Branch)
```

```
## [1] 1 3 2
## Levels: 1 2 3
```

```
# One hot encoding our Payment column
unique(unique_cf$Payment)
```

```
## [1] "Ewallet"     "Cash"        "Credit card"
```

```
unique_cf$Payment <- factor(unique_cf$Payment,
levels <- c("Ewallet" , "Cash", "Credit card"),
labels <- c(1,2,3))
unique(unique_cf$Payment)
```

```
## [1] 1 2 3
## Levels: 1 2 3
```

```r
# One hot encoding our Customer.type column
unique(unique_cf$Customer.type)
```

```
## NULL
```

```r
unique_cf$Customer.type <- factor(unique_cf$Customer.type,
levels <- c("Member" , "Normal"),
labels <- c(1,0))
unique(unique_cf$Customer.type)
```

```
## [1] <NA>
## Levels: 1 0
```

```r
# One hot encoding our Product.line column
unique(unique_cf$Product.line)
```

```
## NULL
```

```r
unique_cf$Product.line <- factor(unique_cf$Product.line,
levels <- c("Health and beauty" , "Electronic accessories", "Home and lifestyle","Sports and travel", "
labels <- c(1,2,3,4,5,6))
unique(unique_cf$Product.line)
```

```
## [1] <NA>
## Levels: 1 2 3 4 5 6
```

```r
# One hot encoding our Gender column
unique(unique_cf$Gender)
```

```
## [1] "Female" "Male"
```

```r
unique_cf$Gender <- factor(unique_cf$Gender,
levels <- c("Female", "Male" ),
labels <- c(0,1))
unique(unique_cf$Gender)
```

```
## [1] 0 1
## Levels: 0 1
```

```r
# Confirming by using summary the outcomes of the one hot encoding
cf_new <- unique_cf[, c(2:8,11,12,13,14,15,16)]
str(cf_new)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  13 variables:
##  $ Branch                 : Factor w/ 3 levels "1","2","3": 1 3 1 1 1 3 1 3 1 2 ...
##  $ Customer type          : chr  "Member" "Normal" "Normal" "Member" ...
```

```
##  $ Gender                 : Factor w/ 2 levels "0","1": 1 1 2 2 2 2 1 1 1 1 ...
##  $ Product line           : chr  "Health and beauty" "Electronic accessories" "Home and lifestyle" "
##  $ 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                : Factor w/ 3 levels "1","2","3": 1 2 3 1 1 1 1 1 3 3 ...
##  $ 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>
```

```
library(Rtsne)
```

```
# Transforming the target feature to factors
carr <- cf_new$Rating
cf_new$Rating <- as.factor(cf_new$Rating)
colors = rainbow(length(unique(cf_new$Rating)))
names(colors) = unique(cf_new$Rating)
```

```
# Calling out the reduction technique t-SNE
cf.tsne <- Rtsne(cf_new[,1:11,13], dims = 2, perplexity = 30, verbose = TRUE, max_iter = 500)
```

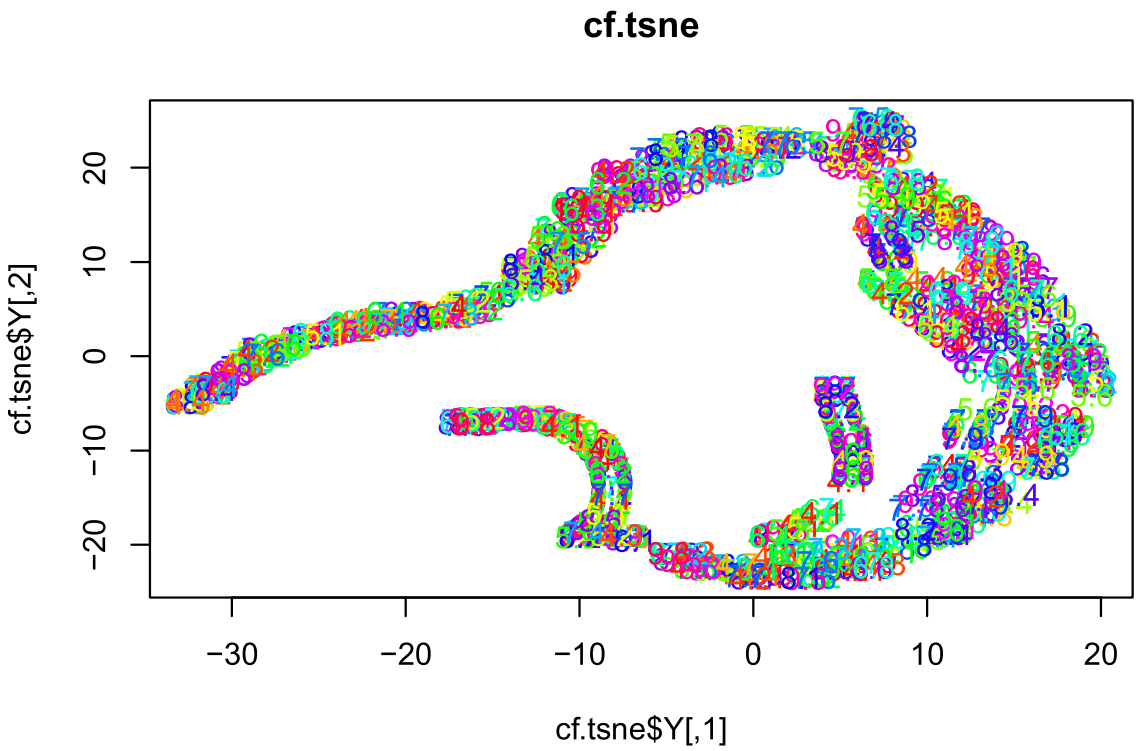
```
## Performing PCA
## Read the 1000 x 18 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.23 seconds (sparsity = 0.102668)!
## Learning embedding...
## Iteration 50: error is 59.746176 (50 iterations in 0.18 seconds)
## Iteration 100: error is 52.076119 (50 iterations in 0.16 seconds)
## Iteration 150: error is 51.045896 (50 iterations in 0.17 seconds)
## Iteration 200: error is 50.742097 (50 iterations in 0.19 seconds)
## Iteration 250: error is 50.578497 (50 iterations in 0.18 seconds)
## Iteration 300: error is 0.576962 (50 iterations in 0.18 seconds)
## Iteration 350: error is 0.402198 (50 iterations in 0.18 seconds)
## Iteration 400: error is 0.365203 (50 iterations in 0.19 seconds)
## Iteration 450: error is 0.350125 (50 iterations in 0.19 seconds)
## Iteration 500: error is 0.339069 (50 iterations in 0.19 seconds)
## Fitting performed in 1.80 seconds.
```

```
cf.time <- system.time(Rtsne(cf_new[,1:11,13], dims = 2, perplexity = 30, verbose = TRUE, max_iter = 50
```

```
## Performing PCA
## Read the 1000 x 18 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
```

```
## Done in 0.22 seconds (sparsity = 0.102668)!
## Learning embedding...
## Iteration 50: error is 60.239665 (50 iterations in 0.23 seconds)
## Iteration 100: error is 52.554588 (50 iterations in 0.16 seconds)
## Iteration 150: error is 51.496980 (50 iterations in 0.17 seconds)
## Iteration 200: error is 50.933478 (50 iterations in 0.17 seconds)
## Iteration 250: error is 50.484348 (50 iterations in 0.16 seconds)
## Iteration 300: error is 0.580900 (50 iterations in 0.17 seconds)
## Iteration 350: error is 0.402669 (50 iterations in 0.17 seconds)
## Iteration 400: error is 0.366858 (50 iterations in 0.17 seconds)
## Iteration 450: error is 0.349441 (50 iterations in 0.15 seconds)
## Iteration 500: error is 0.342239 (50 iterations in 0.16 seconds)
## Fitting performed in 1.70 seconds.
```

```r
# pLOTTING THE OUTCOME
plot(cf.tsne$Y, t = 'n', main = "cf.tsne")
text(cf.tsne$Y,labels = cf_new$Rating,col = colors[cf_new$Rating])
```



Part 2: Feature Selection

```r
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
#Convert the data to factors and integers
cf_new2 <- unique_cf[, c(2:8,11,12,13,14,15,16)]

cf_new2$Branch <- as.numeric(cf_new2$Branch)
cf_new2$Customer.type <- as.numeric(cf_new2$Customer.type)
cf_new2$Gender <- as.numeric(cf_new2$Gender)
cf_new2$Product.line <- as.numeric(cf_new2$Product.line)
cf_new2$Payment <- as.numeric(cf_new2$Payment)

str(cf_new2)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  15 variables:
##  $ Branch                 : num  1 3 1 1 1 3 1 3 1 2 ...
##  $ Customer type          : chr  "Member" "Normal" "Normal" "Member" ...
##  $ Gender                 : num  1 1 2 2 2 2 1 1 1 1 ...
##  $ Product line           : chr  "Health and beauty" "Electronic accessories" "Home and lifestyle" "
##  $ 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                : num  1 2 3 1 1 1 1 1 3 3 ...
##  $ 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 ...
##  $ Customer.type          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Product.line           : num  NA NA NA NA NA NA NA NA NA NA ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
# selecting the numerical and integer columns
CF = (cf_new2[1,3,5:13] )
str(CF)
```

```
## Classes 'data.table' and 'data.frame':   1 obs. of  10 variables:
##  $ Unit price             : num 74.7
##  $ Quantity               : int 7
##  $ Tax                    : num 26.1
##  $ Payment                : num 1
##  $ cogs                   : num 523
##  $ gross margin percentage: num 4.76
##  $ gross income           : num 26.1
##  $ Rating                 : num 9.1
##  $ Total                  : num 549
##  $ V1                     : num 3
##  - attr(*, ".internal.selfref")=<externalptr>
```

Part 3: Association Rules

Loading and cleaning the dataset

```
cf2 <- fread('http://bit.ly/SupermarketDatasetII')
```

```
## Warning in fread("http://bit.ly/SupermarketDatasetII"): Detected 2 column
## names but the data has 3 columns (i.e. invalid file). Added 1 extra default
## column name for the first column which is guessed to be row names or an index.
## Use setnames() afterwards if this guess is not correct, or fix the file write
## command that created the file to create a valid file.

## Warning in fread("http://bit.ly/SupermarketDatasetII"): Stopped early on line
## 10. Expected 3 fields but found 1. Consider fill=TRUE and comment.char=. First
## discarded non-empty line: <<french fries>>
```

```
# preview the first 6 values
head(cf2)
```

```
##                   V1 whole wheat pasta french fries
## 1:              soup       light cream      shallot
## 2: frozen vegetables         spaghetti    green tea
```

```
#Checking our dataset
rows <- nrow(cf2)
cols <- ncol(cf2)

rows
```

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

```
cols
```

```
## [1] 3
```

```
# Loading the arules library

library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

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

```r
#Checking the dimensions of the table
dim(cf2)
```

```
## [1] 2 3
```

```r
# Verifying the object's class
class(cf2)
```

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

```r
## Inspecting rules
rules <- apriori(cf2)
```

```
## Warning: Column(s) 1, 2, 3 not logical or factor. Applying default
## discretization (see '? discretizeDF').
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      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: 0
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[6 item(s), 2 transaction(s)] done [0.00s].
## sorting and recoding items ... [6 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [18 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
rules
```

```
## set of 18 rules
```

```r
summary(rules)
```

```
## set of 18 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
## 12  6
##
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   2.000   2.333   3.000   3.000
##
## summary of quality measures:
##     support      confidence    coverage        lift       count
##  Min.   :0.5   Min.   :1    Min.   :0.5   Min.   :2   Min.   :1
##  1st Qu.:0.5   1st Qu.:1    1st Qu.:0.5   1st Qu.:2   1st Qu.:1
##  Median :0.5   Median :1    Median :0.5   Median :2   Median :1
##  Mean   :0.5   Mean   :1    Mean   :0.5   Mean   :2   Mean   :1
##  3rd Qu.:0.5   3rd Qu.:1    3rd Qu.:0.5   3rd Qu.:2   3rd Qu.:1
##  Max.   :0.5   Max.   :1    Max.   :0.5   Max.   :2   Max.   :1
##
## mining info:
##  data ntransactions support confidence                call
##   cf2             2     0.1        0.8 apriori(data = cf2)
```

```
rules <- apriori(cf2,parameter = list(supp =0.001, conf = 0.8))
```

```
## Warning: Column(s) 1, 2, 3 not logical or factor. Applying default
## discretization (see '? discretizeDF').
```

```
## 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: 0
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[6 item(s), 2 transaction(s)] done [0.00s].
## sorting and recoding items ... [6 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [18 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(rules)
```

```
##      lhs                                rhs                              support confidence coverage
## [1]  {V1=soup}                       => {whole wheat pasta=light cream}     0.5          1      0.5
## [2]  {whole wheat pasta=light cream} => {V1=soup}                           0.5          1      0.5
## [3]  {V1=soup}                       => {french fries=shallot}              0.5          1      0.5
## [4]  {french fries=shallot}          => {V1=soup}                           0.5          1      0.5
## [5]  {whole wheat pasta=light cream} => {french fries=shallot}              0.5          1      0.5
## [6]  {french fries=shallot}          => {whole wheat pasta=light cream}     0.5          1      0.5
```

```
## [7]  {V1=frozen vegetables}           => {whole wheat pasta=spaghetti}       0.5          1      0.5
## [8]  {whole wheat pasta=spaghetti}    => {V1=frozen vegetables}              0.5          1      0.5
## [9]  {V1=frozen vegetables}           => {french fries=green tea}            0.5          1      0.5
## [10] {french fries=green tea}         => {V1=frozen vegetables}              0.5          1      0.5
## [11] {whole wheat pasta=spaghetti}    => {french fries=green tea}            0.5          1      0.5
## [12] {french fries=green tea}         => {whole wheat pasta=spaghetti}       0.5          1      0.5
## [13] {V1=soup,
##       whole wheat pasta=light cream}  => {french fries=shallot}              0.5          1      0.5
## [14] {V1=soup,
##       french fries=shallot}           => {whole wheat pasta=light cream}     0.5          1      0.5
## [15] {whole wheat pasta=light cream,
##       french fries=shallot}           => {V1=soup}                           0.5          1      0.5
## [16] {V1=frozen vegetables,
##       whole wheat pasta=spaghetti}    => {french fries=green tea}            0.5          1      0.5
## [17] {V1=frozen vegetables,
##       french fries=green tea}         => {whole wheat pasta=spaghetti}       0.5          1      0.5
## [18] {whole wheat pasta=spaghetti,
##       french fries=green tea}         => {V1=frozen vegetables}              0.5          1      0.5
```

```
summary(cf2)
```

```
##       V1             whole wheat pasta  french fries
##  Length:2           Length:2           Length:2
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
summary(rules)
```

```
## set of 18 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
## 12  6
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   2.000   2.333   3.000   3.000
##
## summary of quality measures:
##     support      confidence    coverage        lift       count
##  Min.   :0.5   Min.   :1    Min.   :0.5   Min.   :2   Min.   :1
##  1st Qu.:0.5   1st Qu.:1    1st Qu.:0.5   1st Qu.:2   1st Qu.:1
##  Median :0.5   Median :1    Median :0.5   Median :2   Median :1
##  Mean   :0.5   Mean   :1    Mean   :0.5   Mean   :2   Mean   :1
##  3rd Qu.:0.5   3rd Qu.:1    3rd Qu.:0.5   3rd Qu.:2   3rd Qu.:1
##  Max.   :0.5   Max.   :1    Max.   :0.5   Max.   :2   Max.   :1
##
## mining info:
##  data ntransactions support confidence
##   cf2             2   0.001        0.8
##                                                                  call
##  apriori(data = cf2, parameter = list(supp = 0.001, conf = 0.8))
```

```r
rules <- sort(rules, by = "confidence",decreasing = TRUE)
inspect(rules[1:5])
```

```
##     lhs                                rhs                              support confidence coverage li
## [1] {V1=soup}                       => {whole wheat pasta=light cream}     0.5          1      0.5
## [2] {whole wheat pasta=light cream} => {V1=soup}                           0.5          1      0.5
## [3] {V1=soup}                       => {french fries=shallot}              0.5          1      0.5
## [4] {french fries=shallot}          => {V1=soup}                           0.5          1      0.5
## [5] {whole wheat pasta=light cream} => {french fries=shallot}              0.5          1      0.5
```

```r
whole.wheat.pasta <- subset(rules, subset = rhs %pin% "whole.wheat.pasta")
whole.wheat.pasta <- sort(whole.wheat.pasta, by = "confidence", decreasing = TRUE)
inspect(whole.wheat.pasta[1:5])
```

```
##     lhs                         rhs                              support confidence coverage lift cou
## [1] {V1=soup}                => {whole wheat pasta=light cream}     0.5          1      0.5    2
## [2] {french fries=shallot}   => {whole wheat pasta=light cream}     0.5          1      0.5    2
## [3] {V1=frozen vegetables}   => {whole wheat pasta=spaghetti}       0.5          1      0.5    2
## [4] {french fries=green tea} => {whole wheat pasta=spaghetti}       0.5          1      0.5    2
## [5] {V1=soup,
##      french fries=shallot}   => {whole wheat pasta=light cream}     0.5          1      0.5    2
```

Part 4: Anomaly detection

```r
#loading the dataset
cf3 <- fread('http://bit.ly/CarreFourSalesDataset')

# preview the first 6 values
head(cf3)
```

```
##         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
```

```r
#Checking our dataset
rows <- nrow(cf3)
cols <- ncol(cf3)

rows
```

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

```r
cols
```

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

```r
# Loading the necessary libraries
library(tidyverse)
```

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

## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x tidyr::expand()    masks Matrix::expand()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::lift()      masks caret::lift()
## x tidyr::pack()      masks Matrix::pack()
## x dplyr::recode()    masks arules::recode()
## x purrr::transpose() masks data.table::transpose()
## 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(Matrix)
#library(tinytex)
tinytex::install_tinytex()
```

```r
pkg <- c('tidyverse','tibbletime','anomalize','timetk')
#library(pkg)
```

```r
#install.packages("pacman")
# Loading the necessary libraries and automatically installing them if not present
library(pacman)
pacman :: p_load(rio,tidyverse, Amelia ,anomalize)
```

```r
# Checking the size and shape of data
dim(cf3)
```

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

```r
# Viewing data types using str().
str(cf3)
```

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  2 variables:
##  $ Date : chr  "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ Sales: num  549 80.2 340.5 489 634.4 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# To ensure uniformity, I will lowercase all the columns
names(cf3)<- tolower(names(cf3))
head(cf3)
```

```
##         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
```

```
#changing date to date time.
cf3$date <- as.Date(cf3$date, "%m/%d/%y")
head(cf3)
```

```
##          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
```

```
#install.packages("magrittr")
#install.packages("dplyr")
library(magrittr)
```

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

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

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

```
library(dplyr)
```

```
# group and tally the number of transactions per day
cf3.new <- cf3 %>% group_by(date) %>% tally()
colnames(cf3.new) <- c('transactionDate', 'totalCount')
head(cf3.new)
```

```
## # A tibble: 6 x 2
##   transactionDate totalCount
##   <date>               <int>
## 1 2020-01-01              12
## 2 2020-01-02               8
## 3 2020-01-03               8
## 4 2020-01-04               6
## 5 2020-01-05              12
## 6 2020-01-06               9
```

```r
# we now plot using plot_anomaly_decomposition() to visualize out data.
cf3.new %>%
    time_decompose(totalCount) %>%
    anomalize(remainder) %>%
    plot_anomaly_decomposition(ncol = 1, alpha_dots = 0.5) +
    ggtitle("Anomaly Detection Plot")
```

```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = transactionDate

## frequency = 7 days

## trend = 30 days

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
# plotting the recomposition to try and see anomalies
#
cf3.new %>%
    time_decompose(totalCount) %>%
    anomalize(remainder) %>%
    time_recompose() %>%
    plot_anomalies(time_recomposed = TRUE, ncol = 1, alpha_dots = 0.25, fill="cyan") +
    ggtitle("Anomaly detection plots")
```

```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = transactionDate

## frequency = 7 days

## trend = 30 days
```

Anomaly detection plots
observed
20
10
0
Jan
Feb
Mar
Apr
transactionDate
anomaly
No
Yes