Part 3: Association Rules

```
# We first we install the required library
#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)

## Warning: package 'tidyverse' was built under R version 4.0.5

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

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v stringr 1.4.0
## v tidyr   1.1.3     v forcats 0.5.1
## v readr   1.4.0

## -- 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)

## Warning: package 'devtools' was built under R version 4.0.5

## Loading required package: usethis

#install_github("vqv/ggbiplot")
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

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

## Warning: package 'arules' was built under R version 4.0.5

## 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

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

## Warning: package 'arulesViz' was built under R version 4.0.5

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

## Warning: package 'Rtsne' was built under R version 4.0.5

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

## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: lattice

##
## Attaching package: 'caret'

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

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

## Warning: package 'corrplot' was built under R version 4.0.5

## corrplot 0.84 loaded

path <-"Supermarket_Sales_Dataset II.csv"

sales <- read.transactions(path)

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

sales

## transactions in sparse format with
##  7501 transactions (rows) and
##  5729 items (columns)

# Verifying the object's class
# ---
# This should show us transactions as the type of data that we will need
```

```r
# ---
#
class(sales)
```

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

```r
# Previewing our first 5 transactions
#
inspect(sales[1:5])
```

```
##     items
## [1] {cheese,energy,
##      drink,tomato,
##      fat,
##      flour,yams,cottage,
##      grapes,whole,
##      juice,frozen,
##      juice,low,
##      mix,green,
##      oil,
##      shrimp,almonds,avocado,vegetables,
##      smoothie,spinach,olive,
##      tea,honey,salad,mineral,
##      water,salmon,antioxydant,
##      weat,
##      yogurt,green}
## [2] {burgers,meatballs,eggs}
## [3] {chutney}
## [4] {turkey,avocado}
## [5] {bar,whole,
##      mineral,
##      rice,green,
##      tea,
##      water,milk,energy,
##      wheat}
```

```r
# Previewinf items that make up our dataset
#
items<-as.data.frame(itemLabels(sales))
colnames(items) <- "Item"
head(items, 10)
```

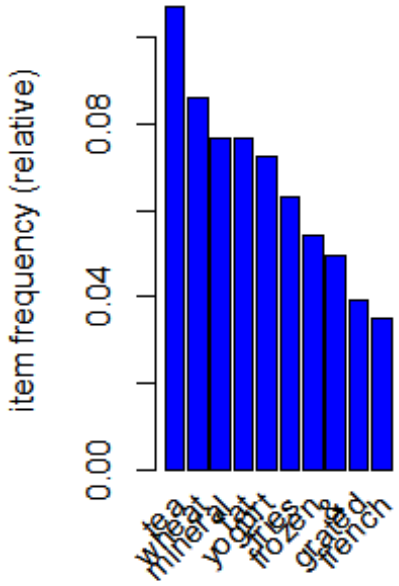
```
##                                       Item
## 1                                        &
## 2                               accessories
## 3                  accessories,antioxydant
## 4              accessories,champagne,fresh
## 5            accessories,champagne,protein
## 6                     accessories,chocolate
```

```
## 7  accessories,chocolate,champagne,frozen
## 8            accessories,chocolate,frozen
## 9               accessories,chocolate,low
## 10       accessories,chocolate,pasta,salt
```

```
# Generating a summary of the transaction dataset

# summary(Transactions)

par(mfrow = c(1, 2))
# plot the frequency of items
itemFrequencyPlot(sales, topN = 10,col="blue")
```




```
# Building a model based on association rules
# using the apriori function
# We use Min Support as 0.001 and confidence as 0.8
#
rules <- apriori (sales, 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 ...[5729 item(s), 7501 transaction(s)] done [0.02s].
## sorting and recoding items ... [354 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [271 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules
```

```
## set of 271 rules
```

```
# Building a apriori model with Min Support as 0.002 and confidence as 0.8.
#rules2 <- apriori (Transactions,parameter = list(supp = 0.002, conf = 0.8))

# Building apriori model with Min Support as 0.002 and confidence as 0.6.
#rules3 <- apriori (Transactions, parameter = list(supp = 0.001, conf = 0.6))

#rules2

#rules3

# We can perform an exploration of our model
# through the use of the summary function as shown
```

```
summary(rules)
```

```
## set of 271 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4
## 107 144  20
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.679   3.000   4.000
##
## summary of quality measures:
##     support           confidence       coverage             lift
##  Min.   :0.001067   Min.   :0.800   Min.   :0.001067   Min.   :  7.611
##  1st Qu.:0.001200   1st Qu.:0.931   1st Qu.:0.001200   1st Qu.: 11.630
##  Median :0.001600   Median :1.000   Median :0.001600   Median : 13.068
##  Mean   :0.002834   Mean   :0.963   Mean   :0.002973   Mean   : 22.372
##  3rd Qu.:0.002666   3rd Qu.:1.000   3rd Qu.:0.002800   3rd Qu.: 20.218
##  Max.   :0.068391   Max.   :1.000   Max.   :0.076523   Max.   :613.718
##      count
```

```
##  Min.   :  8.00
##  1st Qu.:  9.00
##  Median : 12.00
##  Mean   : 21.26
##  3rd Qu.: 20.00
##  Max.   :513.00
##
## mining info:
##   data ntransactions support confidence
##  sales          7501   0.001        0.8
```

```
# Observing rules built in our model
# ---
#
inspect(rules[1:15])
```

```
##      lhs                                  rhs      support     confidence
## [1]  {cookies,low}                     => {yogurt} 0.001066524 1.0000000
## [2]  {cookies,low}                     => {fat}    0.001066524 1.0000000
## [3]  {extra}                           => {dark}   0.001066524 1.0000000
## [4]  {burgers,whole}                   => {wheat}  0.001199840 1.0000000
## [5]  {fries,escalope,pasta,mushroom} => {cream}  0.001066524 1.0000000
## [6]  {fries,cookies,green}             => {tea}    0.001333156 1.0000000
## [7]  {shrimp,whole}                    => {wheat}  0.001066524 1.0000000
## [8]  {rice,cake}                       => {wheat}  0.001333156 1.0000000
## [9]  {tomatoes,whole}                  => {wheat}  0.001066524 0.8000000
## [10] {rice,chocolate}                  => {wheat}  0.001199840 0.9000000
## [11] {flour,green}                     => {weat}   0.001199840 1.0000000
## [12] {rice,chocolate,french}           => {wheat}  0.001066524 1.0000000
## [13] {cake,low}                        => {yogurt} 0.001066524 0.8888889
## [14] {cake,low}                        => {fat}    0.001199840 1.0000000
## [15] {water,low}                       => {yogurt} 0.001199840 0.9000000
##      coverage    lift       count
## [1]  0.001066524  13.813996  8
## [2]  0.001066524  13.067944  8
## [3]  0.001066524  83.344444  8
## [4]  0.001199840  11.629457  9
## [5]  0.001066524  47.777070  8
## [6]  0.001333156   9.341220 10
## [7]  0.001066524  11.629457  8
## [8]  0.001333156  11.629457 10
## [9]  0.001333156   9.303566  8
## [10] 0.001333156  10.466512  9
## [11] 0.001199840 107.157143  9
## [12] 0.001066524  11.629457  8
## [13] 0.001199840  12.279108  8
## [14] 0.001199840  13.067944  9
## [15] 0.001333156  12.432597  9
```

```
# Ordering these rules by a criteria such as the level of confidence
# then looking at the first five rules

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

```
##     lhs                                rhs      support     confidence
## [1] {cookies,low}                   => {yogurt} 0.001066524 1
## [2] {cookies,low}                   => {fat}    0.001066524 1
## [3] {extra}                         => {dark}   0.001066524 1
## [4] {burgers,whole}                 => {wheat}  0.001199840 1
## [5] {fries,escalope,pasta,mushroom} => {cream}  0.001066524 1
##     coverage    lift     count
## [1] 0.001066524 13.81400 8
## [2] 0.001066524 13.06794 8
## [3] 0.001066524 83.34444 8
## [4] 0.001199840 11.62946 9
## [5] 0.001066524 47.77707 8
```

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

yogurt <- subset(rules, subset = rhs %pin% "yogurt")

# Then order by confidence
yogurt<-sort(yogurt, by="confidence", decreasing=TRUE)
inspect(yogurt[1:5])
```

```
##     lhs                   rhs      support     confidence coverage    lift
## [1] {cookies,low}     => {yogurt} 0.001066524 1          0.001066524
13.814
## [2] {wine,low}        => {yogurt} 0.001333156 1          0.001333156
13.814
## [3] {cheese,low}      => {yogurt} 0.001733102 1          0.001733102
13.814
## [4] {mayo,low}        => {yogurt} 0.001733102 1          0.001733102
13.814
## [5] {cookies,low,fat} => {yogurt} 0.001066524 1          0.001066524
13.814
##     count
## [1]  8
## [2] 10
## [3] 13
## [4] 13
## [5]  8
```

If someone buys cookies and low, they are 100% likely to buy yogurt too

Part 4: Anomalies Detection

```
# Installing anomalize package
#
#install.packages("anomalize")
library(tidyverse)
library(anomalize)
```

```
## Warning: package 'anomalize' was built under R version 4.0.5
```

```
## == 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(dplyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:arules':
##
##     intersect, setdiff, union
```

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

```
library(tibbletime)
```

```
## Warning: package 'tibbletime' was built under R version 4.0.5
```

```
##
## Attaching package: 'tibbletime'
```

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

```
# Loading the dataset
forecasting<- read.csv('Supermarket_Sales_Forecasting - Sales.csv')
head(forecasting)
```

```
##        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
```

```
# Collect our time series data
tidyverse_cran_downloads
```

```
## # A time tibble: 6,375 x 3
## # Index:  date
## # Groups: package [15]
##    date       count package
##    <date>     <dbl> <chr>
##  1 2017-01-01   873 tidyr
##  2 2017-01-02  1840 tidyr
##  3 2017-01-03  2495 tidyr
##  4 2017-01-04  2906 tidyr
##  5 2017-01-05  2847 tidyr
##  6 2017-01-06  2756 tidyr
##  7 2017-01-07  1439 tidyr
##  8 2017-01-08  1556 tidyr
##  9 2017-01-09  3678 tidyr
## 10 2017-01-10  7086 tidyr
## # ... with 6,365 more rows
```

```
#converting the data frame to tibble

forecast_tb <- as_tibble(forecasting)
head(forecast_tb)
```

```
## # A tibble: 6 x 2
##   Date      Sales
##   <chr>     <dbl>
## 1 1/5/2019  549.
## 2 3/8/2019   80.2
## 3 3/3/2019  341.
## 4 1/27/2019 489.
## 5 2/8/2019  634.
## 6 3/25/2019 628.
```

```
# install.packages("tibbletime")
#install.packages("tsibble")
library(tibbletime)
library(tsibble)
```

```
## Warning: package 'tsibble' was built under R version 4.0.5
```

```
##
## Attaching package: 'tsibble'
```

```
## The following object is masked from 'package:lubridate':
##
##     interval
```

```
## The following objects are masked from 'package:arules':
##
##     intersect, setdiff, union
```

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

library(lubridate)

#forecast_tb <- forecast_tb %>%
#                    tibbletime::as_tbl_time(index = Date)

tidyverse_cran_downloads %>%
    time_decompose(count) %>%
    anomalize(remainder) %>%
    time_recompose() %>%
    plot_anomalies(time_recomposed = TRUE, ncol = 3, alpha_dots = 0.5)

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