# PART 3 WEEK 14 IP

CHEROTICH FAITH

2022-06-11

**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.

**Association Rules**

In R Language, this is an Unsupervised Non-linear algorithm to uncover how the items are associated with each other. In it, frequent Mining shows which items appear together in a transaction or relation. It's majorly used by retailers, grocery stores, an online marketplace that has a large transactional database. The same way when any online social media, marketplace, and e-commerce websites know what you buy next using recommendations engines. The recommendations you get on item or variable, while you check out the order is because of Association rule mining boarded on past customer data. There are three common ways to measure association:

Support

Confidence

Lift

```
#let's first install packages we shall need
#install.packages("arules")

library(arules)
```

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

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

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

```
library(arulesViz)
```

```
dir <- "C:/Users/user/Downloads/"
df <- file.path(dir,"Supermarket_Sales_Dataset II.csv")
data <- read.csv(df)
head(data)
```

```
##              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
```

```
#using the dim()function, we check the number of #rows and columns in our dataset

dim(data)
```

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

our dataset has 7500 entries with 20 columns

```
#let's preview the data types structure of our dataset using the str()
str(data)
```

```
## 'data.frame':    7500 obs. of  20 variables:
##  $ shrimp           : chr  "burgers" "chutney" "turkey" "mineral water" ...
##  $ almonds          : chr  "meatballs" "" "avocado" "milk" ...
##  $ avocado          : chr  "eggs" "" "" "energy bar" ...
##  $ vegetables.mix   : chr  "" "" "" "whole wheat rice" ...
##  $ green.grapes     : chr  "" "" "" "green tea" ...
##  $ whole.weat.flour : chr  "" "" "" "" ...
##  $ yams             : chr  "" "" "" "" ...
##  $ cottage.cheese   : chr  "" "" "" "" ...
##  $ energy.drink     : chr  "" "" "" "" ...
##  $ tomato.juice     : chr  "" "" "" "" ...
##  $ low.fat.yogurt   : chr  "" "" "" "" ...
##  $ green.tea        : chr  "" "" "" "" ...
```

```
##  $ honey            : chr  "" "" "" "" ...
##  $ salad            : chr  "" "" "" "" ...
##  $ mineral.water    : chr  "" "" "" "" ...
##  $ salmon           : chr  "" "" "" "" ...
##  $ antioxydant.juice: chr  "" "" "" "" ...
##  $ frozen.smoothie  : chr  "" "" "" "" ...
##  $ spinach          : chr  "" "" "" "" ...
##  $ olive.oil        : logi  NA NA NA NA NA NA ...
```

All the variables are factors except the olive.oil variable in logical form

**Data Cleaning**

```
#checking for missing values per column in the dataset
colSums(is.na(data))
```

```
##            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
```

```
#hence compute the sum of missing values
sum(is.na(data))
```

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

There are 7500 missing data as a result of the olive oil column with no entries at all. Therefore, we will have to drop this column.

```
#drop the olive oil column
data$olive.oil <- NULL
colnames(data)
```

```
##  [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"
```

Having dropped the olive oil column, we now have 19 columns for our analysis.

```
#let's preview a summary of our dataset
summary(data)
```

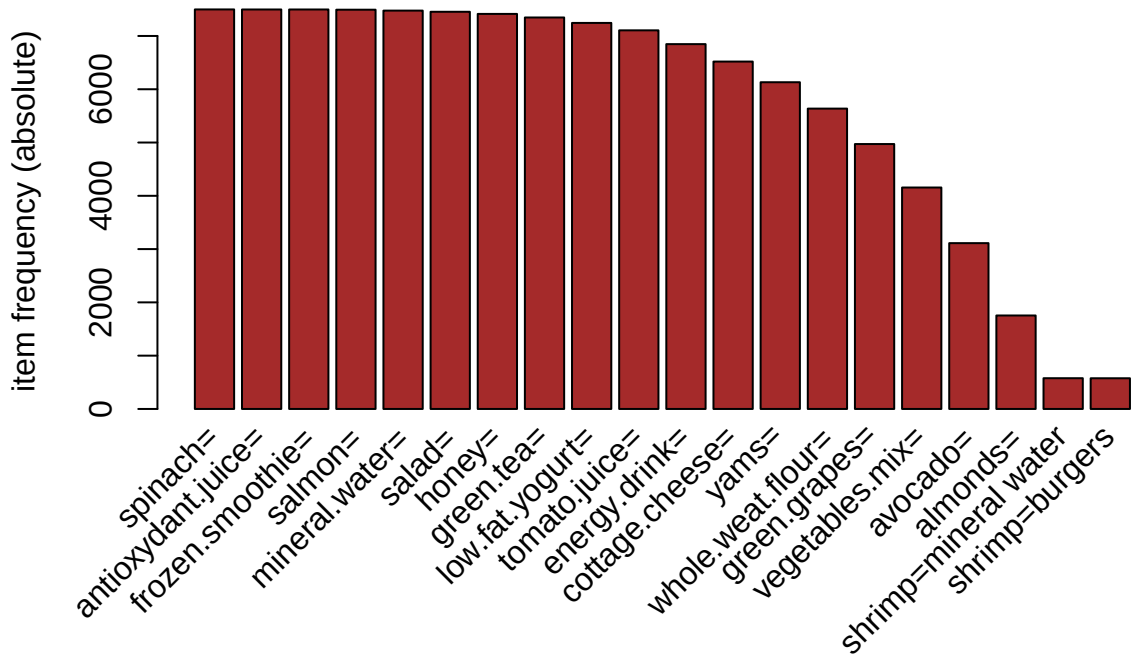
```
##     shrimp            almonds           avocado          vegetables.mix
##  Length:7500        Length:7500        Length:7500        Length:7500
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  green.grapes       whole.weat.flour       yams           cottage.cheese
##  Length:7500        Length:7500        Length:7500        Length:7500
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  energy.drink       tomato.juice       low.fat.yogurt      green.tea
##  Length:7500        Length:7500        Length:7500        Length:7500
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##     honey              salad           mineral.water         salmon
##  Length:7500        Length:7500        Length:7500        Length:7500
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  antioxydant.juice  frozen.smoothie      spinach
##  Length:7500        Length:7500        Length:7500
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
# we plot a frequency plot to visualize the items that were mostly purchased

plot_data <- as(data, "transactions")
```

```
## Warning: Column(s) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
## 17, 18, 19 not logical or factor. Applying default discretization (see '?
## discretizeDF').
```

```
itemFrequencyPlot(plot_data, topN=20, type="absolute", col="brown")
```




There are over 6,000 sales for Spinach, antioxydant.juice, frozen.smoothie, etc. However, shrimp.mineral water and burgers had lower sales.

```
#We now build a model using the apriori()function, where we use Min support as 0.5 and confidence as 0.
rules_df <- apriori(data, parameter = list(supp=0.5,conf=0.8, target="rules",minlen=2))
```

```
## Warning: Column(s) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
## 17, 18, 19 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     0.5      2
##  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: 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(data, 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 [0.12s].
## creating S4 object  ... done [0.28s].
```

```
#preview the summary of new dataset
summary(rules_df)
```

```
## 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        coverage           lift
##  Min.   :0.5541   Min.   :0.8108   Min.   :0.5541   Min.   :1.000
##  1st Qu.:0.5541   1st Qu.:1.0000   1st Qu.:0.5541   1st Qu.:1.001
##  Median :0.6629   Median :1.0000   Median :0.6629   Median :1.021
##  Mean   :0.6455   Mean   :0.9882   Mean   :0.6545   Mean   :1.095
##  3rd Qu.:0.7516   3rd Qu.:1.0000   3rd Qu.:0.7516   3rd Qu.:1.150
##  Max.   :0.9996   Max.   :1.0000   Max.   :0.9997   Max.   :1.508
##      count
##  Min.   :4156
##  1st Qu.:4156
##  Median :4972
##  Mean   :4841
##  3rd Qu.:5637
##  Max.   :7497
##
## mining info:
##  data ntransactions support confidence
##  data          7500     0.5        0.8
##                                                                                call
##  apriori(data = data, parameter = list(supp = 0.5, conf = 0.8, target = "rules", minlen = 2))
```

```
#lets use the inspect()function to preview our model
inspect(rules_df[1:5])
```

```
##     lhs                   rhs                 support   confidence coverage
## [1] {vegetables.mix=} => {green.grapes=}     0.5541333 1.0000000  0.5541333
```

```
## [2] {green.grapes=}   => {vegetables.mix=}   0.5541333 0.8358809  0.6629333
## [3] {vegetables.mix=} => {whole.weat.flour=} 0.5541333 1.0000000  0.5541333
## [4] {vegetables.mix=} => {yams=}             0.5541333 1.0000000  0.5541333
## [5] {vegetables.mix=} => {cottage.cheese=}   0.5541333 1.0000000  0.5541333
##     lift     count
## [1] 1.508447 4156
## [2] 1.508447 4156
## [3] 1.330495 4156
## [4] 1.223092 4156
## [5] 1.150307 4156
```

From the above results, for instance if a person picks vegetable mix, the possibility of picking green grapes is 100% which is contrary to if the person first picks green grapes, the possibility of picking vegetable mix decreases to 84%.

```
#let's use the sort() to order the dataset rules by confidence or support then inspect the 5 rules agai

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

```
##     lhs                   rhs                  support   confidence coverage
## [1] {vegetables.mix=} => {green.grapes=}     0.5541333 1          0.5541333
## [2] {vegetables.mix=} => {whole.weat.flour=} 0.5541333 1          0.5541333
## [3] {vegetables.mix=} => {yams=}             0.5541333 1          0.5541333
## [4] {vegetables.mix=} => {cottage.cheese=}   0.5541333 1          0.5541333
## [5] {vegetables.mix=} => {energy.drink=}     0.5541333 1          0.5541333
##     lift     count
## [1] 1.508447 4156
## [2] 1.330495 4156
## [3] 1.223092 4156
## [4] 1.150307 4156
## [5] 1.095370 4156
```

If we order the rules confidence by the decreasing order, the possibility of buying vegetable mix then green.grapes is 100%.