Learning outcomes

After solving these exercises, you should be able to understand the following:

1. Reading transaction data and exploring the data and items
2. Implementing association rule mining in R
3. Understanding the computation of support, confidence and lift
4. Interpreting rules and results
5. Obtaining the patterns/rules from a supervised dataset and computing the metrics support, confidence, lift for the rules

1. Let us understand the apriori algorithm by using sample retail set. This is a data set of stationery store across three days. The data set name is "Transactions.csv".
   a. Arrange the above data in the transaction format
   b. Find the frequent items set using minimum support = 50%
   c. Form the rules taking the items from the Frequent item set and retail or drop the rules taking confidence as 50%
   d. Compute lift for the above rules

2. Association Rules for transaction data:
   Steps to follow:
   a. Install and load 'arules' package

      install.packages("arules")
   b. Read transaction 'Transactions.csv' data in the way arules package should treats the transaction data
      trans = read.transactions(file="Transactions.csv", rm.duplicates= FALSE, format="single",sep=",",cols =c(1,2))
   c. You will get an error in the above step. This is due to duplicates in the Data. Please change the parameter rm.dulicates=TRUE, to fix the error
   d. Check the data in transactions format
      inspect(trans)
   e. Explore and understand the data and items of transaction data
      trans
      itemFrequency(trans)
      itemFrequencyPlot(trans)
   f. Implementing association mining using 'Apriori' algorithm to extract rules
      rules <- apriori(trans,parameter = list(sup = 0.5, conf = 0.6,target="rules"))
   g. Understanding the rules
      summary(rules)
      inspect(rules)
      image(trans)

3. Practice above analysis on 'Groceries' data set (in-built data set in R) which has 9835 transactions and 169 items.
   #To load data

```
data("Groceries")
Groceries
```

4. Now let us come to the real world application of association rules. You are now going to apply arules on a gaming set that consists of 0.7 million records.

   a. Please take the data set “games.csv”.

   b. Since we are going to build the algorithm on the customerid (CONTACT_WID) and item(ITEM_NAME). You can subset this data and store in another data frame

   c. You can now give this data frame as input to the “read.transactions” function, however, as a best practice, you can save this data frame into a CSV (with row.names=FALSE) file first, and then read this and then give as input to “read.transactions”.

      Note: The above step results in 106625 transactions with 20 items

   d. Want to see a sample of say, 5 transactions? Use the code mentioned below

      inspect(head(trans.data,5))

   e. Apply Arules algorithm and generate the rules. You may start with support as 0.001 and confidence as 0.8, as we so many transactions

   f. Print the 5 rules sorted by confidence and then support as a data.frame.

      as(head(sort(rules, by = c("confidence","support")), n=5), "data.frame")

   g. Order of rules by decreasing support

   h. as(rules[sort(rules, by = "confidence", order = TRUE)],"data.frame")

   i. Storing the rules in a data frame. This will help us to apply filters and processing of the rules, such as sorting the rules based on the support, confidence or lift. Or Filtering the rules satisfying a particular condition either on LHS or RHS

      library(stringr)

      rules <- as(rules,"data.frame")

      write.csv(rules,"rules.csv")

   j. m=str_split(rules$rules,"=>")

   k. Class = data.frame(Class = unlist(lapply(m,function(x){str_trim(x[2])})))

   l. Rule = data.frame(Rule = unlist(lapply(m,function(x){str_trim(x[1])})))

   m. rules2= data.frame(Rule,Class,"support"=rules$support,

```
        "confidence"= rules$confidence,
        "lift"= rules$lift)
```

n. Rulesset = unique(rules2)

o. To filter the rules

```
subset(Rulesset,Rulesset$Class=="{Choclates}")
```

p. You can control the length of the rules by setting the parameters maxlen='n'.

```
rules <-  apriori(transData,parameter = list(maxlen=3,sup = 0.001,
                              conf = 0.80,target="rules"))
```

q. If you want to only most frequent item sets, "target" parameter to "frequent itemsets"