**Hannah Roach**
2/22/2019

# Lab Exercise 5: Association Rules

| | |
|---|---|
| **Purpose:** | This lab is designed to investigate and practice Association Rules. After completing the tasks in this lab you should able to:<br>• Use R functions for Association Rule based models |

| | |
|---|---|
| **Tasks:** | Tasks you will complete in this lab include:<br>• Use the R –Studio environment to code Association Rule models<br>• Apply constraints in the Market Basket Analysis methods such as minimum thresholds on support and confidence measures that can be used to select interesting rules from the set of all possible rules<br>• Use R graphics "arules" to execute and inspect the models and the effect of the various thresholds |

| | |
|---|---|
| **References:** | • The groceries data set - provided for arules by Michael Hahsler, Kurt Hornik and Thomas Reutterer. http://rss.acs.unt.edu/Rdoc/library/arules/html/Groceries.html<br>  ○ Michael Hahsler, Kurt Hornik, and Thomas Reutterer (2006) Implications of probabilistic data modeling for mining association rules. In M. Spiliopoulou, R. Kruse, C. Borgelt, A. Nuernberger, and W. Gaul, editors, *From Data and Information Analysis to Knowledge Engineering, Studies in Classification, Data Analysis, and Knowledge Organization*, pages 598–605. Springer-Verlag. |

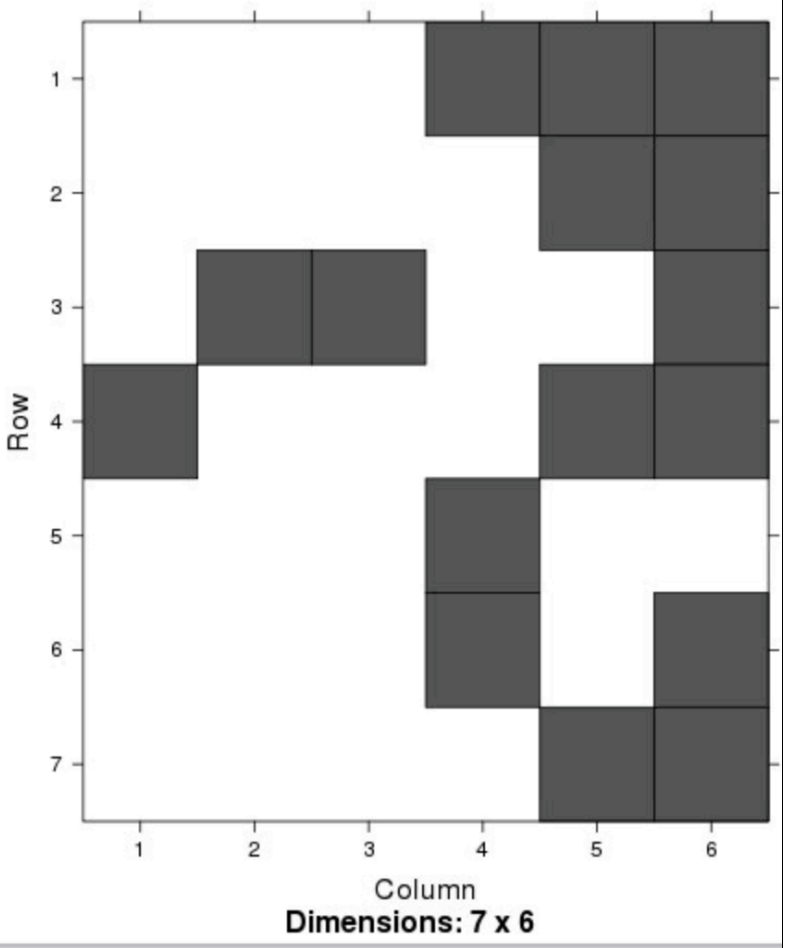
## Workflow Overview

1. Set the Working Directory and install the “arules” and "arulesViz" package
2. Read in the Data for Modeling
3. Review Transaction data
4. Plot Transactions
5. Mine the Association Rules
6. Read in Groceries dataset
7. Mine the Rules for the Groceries Data and Visualize results
8. Extract the Rules in which the Confidence Value is >0.8 and high lift and visualize resuts

# LAB Instructions

| Step | Action |
|---|---|
| 1 | Log in with GPADMIN credentials on to R-Studio. |
| 2 | **Set the Working Directory and install the "arules" package:**<br>To understand Market Basket Analysis and the R package "arules," use a simple set of transaction lists of "book-purchases".<br><br>1. Set the working directory to ~/LAB05/ by executing the command:<br><br>`setwd("~/LAB05")`<br><br>• (Or using the "Tools" option in the tool bar in the RStudio environment.)<br><br>2. Load the package (select the mirror if prompted) and the required libraries:<br><br>`#Install the packages and load libraries`<br>`>install.packages('arules')`<br>`>install.packages('arulesViz')`<br>`>library('arules')`<br>`>library ('arulesViz')` |

| Step | Action |
|---|---|
| 3 | **Read in the Data for Modeling:**<br><br>• **Transaction List** is a special data type function in the "arules" package.<br><br>1. Read the data in as a Transaction List using the following statement for the states data, "MBAdata.csv".<br><br>`> #read in the csv file as a transaction data`<br>`> txn <- read.transactions ("MBAdata.csv",rm.duplicates = FALSE,format="single",sep=",",cols=c(1,2))`<br><br>The arguments for the **read.transaction functions** are detailed below:<br><br>• **file** the file name.<br>• **format** a character string indicating the format of the data set. One of "basket" or "single", can be abbreviated.<br>• **Sep** a character string specifying how fields are separated in the data file, or NULL (default). For basket format, this can be a regular expression; otherwise, a single character must be given. The default corresponds to white space separators.<br>• **Cols** For the 'single' format, cols is a numeric vector of length two giving the numbers of the columns (fields) with the transaction and item ids, respectively. For the 'basket' format, cols can be a numeric scalar giving the number of the column (field) with the transaction ids. If cols = NULL<br>• **rm.duplicates** a logical value specifying if duplicate items should be removed from the transactions. |
| 4 | **Review Transaction data:**<br>1. First inspect the transaction data<br>`>txn@transactionInfo`<br>`>txn@itemInfo`<br>2. Review the results on the console<br>`> txn@transactionInfo`<br>`transactionID`<br>`1 101`<br>`2 102`<br>`3 103`<br>`4 104`<br>`5 105`<br>`6 106`<br>`7 107`<br>`>`<br>`> txn@itemInfo`<br>`labels`<br>`1 Harry-Potter-DVD`<br>`2 Jane-Austen`<br>`3 Learn-Spanish`<br>`4 PSQL-basics`<br>`5 R-basics`<br>`6 Stat-intro`<br>`>` |

| Step | Action |
|---|---|
| 5 | **Plot Transactions:**<br>1. Use the "image" function that shows a visual representation of the transaction set in which the rows are individual transactions (identified by transaction ids) and the dark squares are items contained in each transaction.<br>`> image(txn)`<br>  <br>2. Review the output in the graphics window |

6

**Mine the Association Rules:**

The **"apriori" function**, provided by the *arulesr* package, is used as follows:

```
rules <- apriori(File,
                 parameter = list(supp = 0.5, conf = 0.9,
                                  target = "rules"))
```

where the arguments are:

- **data** object of class transactions or any data structure which can be coerced into transactions (for example, a binary matrix or data.frame).

- **parameter** named list. The default behavior is to mine rules with support 0.1, confidence 0.8, and maxlen 5.

1. Read in the statement for the transaction data:

```
> #mine association rules
> basket_rules <-
apriori(txn,parameter=list(sup=0.5,conf=0.9,target="rules"
))
```

```
> basket_rules <- apriori(txn,parameter=list(sup=0.5,conf=0.9,target="rules"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
        0.9    0.1    1 none FALSE            TRUE     0.5      1     10  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[6 item(s), 7 transaction(s)] done [0.00s].
sorting and recoding items ... [2 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [1 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>
```

2. Review the output on the console. The number of rules generated can be seen in the output and is represented as follows:

```
writing ... [1 rule(s)] done [0.00s]
```

3. Inspect the rule using the following statement:

```
> inspect(basket_rules)
```

| Step | Action |
|---|---|
| | > inspect(basket_rules)<br>lhs rhs support confidence lift<br>1 {R-basics} => {Stat-intro} 0.571 1 1.17<br>><br><br>4. Review the output.<br><br>5. State the generated rule and the support, confidence and the lift thresholds for the rule<br><br>0.571, 1, 1.17 |

| Step | Action |
|---|---|
| 7 | **Read in Groceries dataset**<br><br>Use the standard data set, “Groceries” available with the “arules” package.<br><br>• The Groceries data set contains 1 month (30 days) of real-world point-of-sale transaction data from a typical local grocery outlet. The data set contains 9835 transactions and the items are aggregated to 169 categories.<br><br>1. Read in the data set and inspect the item information<br><br>`> #Read in Groceries data`<br>`> data(Groceries)`<br>`> Groceries@itemInfo`<br><br>`> Groceries@itemInfo`<br>labels level2 level1<br>1 frankfurter sausage meet and sausage<br>2 sausage sausage meet and sausage<br>3 liver loaf sausage meet and sausage<br>4 ham sausage meet and sausage<br>5 meat sausage meet and sausage<br>6 finished products sausage meet and sausage<br>7 organic sausage sausage meet and sausage<br>8 chicken poultry meet and sausage<br>9 turkey poultry meet and sausage<br>10 pork pork meet and sausage<br>11 beef beef meet and sausage<br>12 hamburger meat beef meet and sausage<br>13 fish fish meet and sausage<br>14 citrus fruit fruit fruit and vegetables |

| Step | Action |
|---|---|
| 8 | **Mine the Rules for the Groceries Data:**<br><br>`> #mine rules`<br>`> rules <- apriori(Groceries, parameter=list(support=0.001, confidence=0.5))`<br><br>• Note the values used for the parameter list.<br><br>`> rules <- apriori(Groceries, parameter=list(support=0.001, confidence=0.5))`<br><br>`parameter specification:`<br>`confidence minval smax arem aval originalSupport support minlen maxlen target ext`<br>`0.5 0.1 1 none FALSE TRUE 0.001 1 10 rules FALSE`<br><br>`algorithmic control:`<br>`filter tree heap memopt load sort verbose`<br>`0.1 TRUE TRUE FALSE TRUE 2 TRUE`<br><br>`apriori - find association rules with the apriori algorithm`<br>`version 4.21 (2004.05.09) (c) 1996-2004 Christian Borgelt`<br>`set item appearances ...[0 item(s)] done [0.00s].`<br>`set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].`<br>`sorting and recoding items ... [157 item(s)] done [0.00s].`<br>`creating transaction tree ... done [0.00s].`<br>`checking subsets of size 1 2 3 4 5 6 done [0.02s].`<br>`writing ... [5668 rule(s)] done [0.00s].`<br>`creating S4 object ... done [0.00s].`<br>`>`<br><br>1. How many rules are generated? 5668 rules |

| Step | Action |
|---|---|
| 9 | **Extract the Rules in which the Confidence Value is >0.8 and high lift:**<br><br>1. Execute the following commands:<br>`> subrules <- rules[quality(rules)$confidence > 0.8]`<br>`> plot(subrules, control = list(jitter=2))`<br>`> inspect(subrules)`<br>**Screenshot of scatterplot**<br>`> plot(subrules, control=list(jitter=2))`<br>`Error in as.double(y) :`<br>`cannot coerce type 'S4' to vector of type 'double'`<br>`>`<br><br>Note: This is the datatype error that we discussed.<br><br>2. Review the results.<br>3. How many sub-rules did you extract? 371<br>• These rules are more valuable for the business.<br><br>4. Extract the top three rules with high threshold for the parameter “lift” and plot.<br><br>`> #Extract the top three rules with high lift`<br>`> rules_high_lift <- head(sort(rules, by="lift"), 3)`<br>`> inspect(rules_high_lift)`<br>`> rules_high_lift <- head(sort(rules, by="lift"), 3)`<br>`> inspect(rules_high_lift)`<br>`lhs rhs support confidence lift`<br>`1 {Instant food products, soda} => {hamburger meat} 0.00122 0.632 19.0`<br>`2 {soda, popcorn} => {salty snack} 0.00122 0.632 16.7`<br>`3 {flour, baking powder} => {sugar} 0.00102 0.556 16.4`<br>`> plot(rules_high_lift, method="graph", control=list(type="items"))`<br>`Error in as.double(y) :`<br>`cannot coerce type 'S4' to vector of type 'double'`<br><br>Note: This is the datatype error that we discussed. |

*End of Lab Exercise*