CS 176 Programming Assignment 1

Isabelle Tingzon
201146060
ibtingzon@upd.edu.ph

March 23, 2015

Contents

1 Introduction

Mining association rules and frequent item sets allows for the discovery of interesting and useful connections or relationships between items. Mining of association rules has numerous real-world applications in various areas including retail basket analysis, finding patterns in biological databases, analysis of protein combinations,to study population and economic census, etc. [11]

In this programming assignment, I use the Apriori and FP-growth algorithms to mine association rules from a sample retail market basket data set. I then present evaluate the results based on several interest measures (lift, IR, Kulc).

2 Objectives

In this programming assignment, I aim to demonstrate my knowledge of association rules by using Apriori and FP-growth algorithms to mine association rules from a sample retail market basket data set. The objectives of the study are the following:

- to obtain association rules and analyze them for better decision support, better understanding of data, or increasing company profit using the Apriori Algorithm and FP-Growth Algorithm
- to analyze association rules based on relevance, interestingness, and correlation, and use lift, Imbalance Ratio (IR), and Kulczynski (Kulc) measure as correlation measures.

3 Methodology

3.1 Preliminaries

In this section, I discuss the preliminaries and mathematical background used in mining association rules.

3.1.1 Association Rules

An association rule is an implication of the form $(A \Rightarrow B)$ where $A, B \neq \emptyset$, $A \subset I$, $B \subset I$, and $A \cap B \neq \emptyset$. Association rules are used to identify and uncover relationships among data. In this programming assignment, we use association rules to identify items that are frequently purchased together.

3.1.2 Support and Confidence

The support $supp(X)$ is the proportion of transactions in the data set that contains the itemset. That is, for an association rule $(X \Rightarrow Y)$ where X and Y are item sets, the support can be calculated using the following formula:

$$supp(X \Rightarrow Y) = P(X \cup Y) \tag{1}$$

Meanwhile, the confidence of an association rule $A \Rightarrow B$, where A and B are item sets, is simply the probability of B given A. The confidence is given by the formula in equation (2),

$$confidence(A \Rightarrow B) = P(B|A) = \frac{supp(A \cup B)}{supp(A)} \tag{2}$$

3.1.3 Interest Measures

In this programming assignment, we are interested in certain correlation measures, particularly the lift, Imbalance Ratio, and Kulczynski Measure. Later, we will see how these interest measures will be helpful in pattern evaluation. The lift is a measure of correlation given by equation (3).

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)} \tag{3}$$

If $lift < 1$, then the occurrence of A is negatively correlated with the occurrence of B, or the occurrence of one leads to the absence of the other. If $lift > 1$, then the occurrence of A is positively correlated with the occurrence of B, or the occurrence of one implies the occurrence of the other. If $lift = 1$, then there is no correlation between A and B, or A and B are independent. Obviously, we would want to remove rules with lift values less than or equal to 1.

For the programming assignment, we are also interested in the values of the Imbalance Ratio and Kulczynski Measure for each association rule. The Imbalance Ratio (IR) assesses the imbalance of two itemsets A and B and is given by equation (4).

$$IR(A,B) = \frac{|supp(A) - supp(B)|}{supp(A) + supp(B) - supp(A \cup B)} \tag{4}$$

An interesting property of IR is that if $confidence(A \Rightarrow B) = confidence(B \Rightarrow A)$, then $IR(A,B) = 0$. An IR value of 0 implies balance. [5]

Meanwhile the Kulczynski (Kulc) Measure is given by,

$$Kulc(A,B) = \frac{1}{2}(P(A|B) + P(B|A)) = \frac{1}{2}(Conf(A \Rightarrow B))(Conf(B \Rightarrow A)) \tag{5}$$

which is the average confidence of the two association rules $(A \Rightarrow B)$ and $(B \Rightarrow A)$ or the *arithmetic mean.* [5]

3.2 Apriori Algorithm

The Apriori algorithm uses prior knowledge of frequent itemset properties to mine frequent itemsets for Boolean association rules [3]. The algorithm for Apriori is given in Figure 1.

```
1)  L_1 = {large 1-itemsets};
2)  for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
3)      C_k = apriori-gen(L_{k-1});  // New candidates – see Section 2.1.1
4)      forall transactions t ∈ D do begin
5)          C_t = subset(C_k, t);  // Candidates contained in t – see Section 2.1.2
6)          forall candidates c ∈ C_t do
7)              c.count++;
8)      end
9)      L_k = {c ∈ C_k | c.count ≥ minsup}
10) end
11) Answer = ⋃_k L_k;
```

Figure 1: Apriori Algorithm by Agrawal, Rakesh and Srikant [3]

Recently, Hahsler et. al developed the *arules R-Package Ecosystem* which can be used to implement the basic infrastructure for basic algorithms to efficiently find and analyze association rules [6, 7]. In this programming assignment, we use the arules package to mine the dataset using Apriori.

3.2.1 Pre-processing Techniques

The data mining process may involve pre-processing steps in order to assure that the data set have the quality and the format required by the algorithms [10]. To process the retail market basket data set (given as *basket.dat*), I

read the data as a transaction, specified the delimiter as a white space, and removed duplicate transactions. The following code can be seen in Listing 1.

Listing 1: Formatting data for Apriori Algorithm

```
Basket <- read.transactions("basket.dat",
        sep = " ", rm.duplicates = TRUE)
```

3.2.2 System Parameters

To run the Apriori algorithm on the market basket data set *basket.dat*, I invoke the Apriori method, as seen in Listing 2, from the arules package to generate the association rules. In the arules package, *apriori* takes two parameters: minimal support and minimal confidence.

The default values for support and confidence are 0.1 and 0.8 respectively.

Listing 2: Invoking the Apriori Method

```
rules <- apriori(Basket, parameter =
        list(supp = 0.1, conf = 0.8))
```

The *apriori* method returns a set of rules. The number of rules produced depends on the the minimal support count and minimal confidence specified. For example, a minimal support count of 0.1 and a minimal confidence of 0.8 produces a set of 172 rules.

To inspect the rules we invoke *inspect(rules)* which gives a detailed description of the rules including the left-hand side (lhs), the right-hand side (rhs), and the lift. To calculate the IR and Kulc, I implemented additional R code. First, we need the support values for each of the lhs and rhs itemset (*rules.lhs.support* and *rules.rhs.support* respectively), as well as the the support count of the union of the lhs and rhs (*rules.union.support*). R readily supplies the support of an itemset through the *support* method as seen in Listing 3.

Listing 3: Getting Support Count

```
rules.lhs.supp <- support(rules@lhs, Basket,
        type= c("relative", "absolute"),
        control = NULL)
rules.rhs.supp <- support(rules@rhs, Basket,
        type= c("relative", "absolute"),
        control = NULL)
rules.union.supp <- support(rules, Basket,
```

```
        type= c("relative", "absolute"),
        control = NULL)
```

We can then calculate IR and Kulc as seen in Listings 4 and 5. We use *cbind* to bind the resulting vectors to *quality(rules)*, so that when we invoke *quality(rules)* or *inspect(rules)*, we find additional columns for IR and Kulc.

Listing 4: Imbalance Ratio

```
imbalance_ratio <- abs(rules.lhs.supp -
        rules.rhs.supp)/ (rules.lhs.supp +
        rules.rhs.supp - rules.union.supp)

quality(rules) <- cbind(quality(rules),
        imbalance_ratio)
```

Listing 5: Kulczynski Measure

```
kulc <- (1/2)*((rules.union.supp/rules.lhs.supp)
         + (rules.union.supp/rules.rhs.supp))

quality(rules) <- cbind(quality(rules), kulc)
```

3.2.3 Post-processing

To improve the quality of the results, I implement some post-processing techniques on the produced association rules. Since lift is a good interest measure for correlation, I sort the rules according to lift. Then I prune the redundant rules that provide no additional knowledge by following the process in [2]. That is, according to [2], if a rule A is a super rule of another rule B and rule A has a lower lift, then the former rule, rule A, is considered to be redundant. The complete R code for pruning redundant rules can be seen in Listing 9.

Listing 6: Sorting and Pruning Redundant Rules

```
rules.sorted.lift <- sort(rules, by="lift")
subset.matrix <- is.subset(rules.sorted.lift,
        rules.sorted.lift)
subset.matrix[lower.tri(subset.matrix, diag=T)]
         <- NA
redundant <- colSums(subset.matrix,
         na.rm=T) >= 1
rules.pruned <- rules.sorted.lift[!redundant]
```

Mining association rules may produce very large numbers and large sets of rules which may be time consuming to sift through. Therefore, it is helpful to use data visualization tools to have a better understanding of the data. Hashler et. al developed an R-extension package called arulesViz to aid in data visualization. [9] In this paper, I used graph-based techniques to visualize association rules. In graphs, vertices represent items or itemsets and edges indicate relationships in rules [9].

3.3 FP-Growth Algorithm

Being one of the fastest approaches to frequent item set mining, the Frequent Pattern-Growth or FP-Growth Algorithm uses a divide-and-conquer strategy to compress databases into a frequent pattern tree or FP-tree while retaining itemset association information and divides the database into a set of conditional databases (with each database associated with one frequent item) [4, 8]. Figure 2 illustrates the FP-growth algorithm as described in [8].

Method: *call FP-growth*(FP-tree, *null*).

Procedure *FP-growth*(*Tree*, α)
{
(1) *if Tree* contains a single prefix path // Mining single prefix-path FP-tree
(2) *then* {
(3) *let P* be the single prefix-path part of *Tree*;
(4) *let Q* be the multipath part with the top branching node replaced by a *null* root;
(5) *for each* combination (denoted as β) of the nodes in the path *P do*
(6) *generate* pattern $\beta \cup \alpha$ with *support* = *minimum support of nodes in* β;
(7) *let freq_pattern_set*(*P*) be the set of patterns so generated; }
(8) *else let Q* be *Tree*;
(9) *for each* item a_i in *Q do* { // Mining multipath FP-tree
(10) *generate* pattern $\beta = a_i \cup \alpha$ with *support* = a_i.*support*;
(11) *construct* β's conditional pattern-base and then β's conditional FP-tree $Tree_\beta$;
(12) *if* $Tree_\beta \neq \emptyset$
(13) *then call FP-growth*($Tree_\beta$, β);
(14) *let freq_pattern_set*(*Q*) be the set of patterns so generated; }
(15) *return*(*freq_pattern_set*(*P*) $\cup$ *freq_pattern_set*(*Q*) $\cup$ (*freq_pattern_set*(*P*) $\times$ *freq_pattern_set*(*Q*)))
}

Figure 2: FP-growth Algorithm by Han, Pei, Yin [8]

Borgelt [4] described a C implementation of the FP-growth algorithm which I use in this programming assignment. In particular, I used the ex-

ecutable file available at [1] to mine association rules and produce each of their corresponding support, confidence, and lift.

3.3.1 Pre-processing Techniques

For FP-growth, no pre-processing steps were necessary.

3.3.2 System Parameters

I execute the FP-growth C Implementation developed by Borgelt in [1] using the command line. The format of the command is given as *fpgrowth [options] infile [outfile].* The *-tr* option produces the association rules. *-s* and *-c* can be used to specify the minimal support count and minimal confidence, respectively. The default values for support is 10 and confidence is 80. The complete Windows command for producing association rules using FP-Growth by Borgelt is given in Listing 7.

Listing 7: Windows Command for Executing FP-Growth by Borgelt

```
cmd /k fpgrowth.exe -tr -s10 -c80
        basket.dat fpg.dat
```

To get the lift, I specify additional options *-el* and *-v,%el* for generating the additional interest measure, lift. The full command can be seen in Listing 8.

Listing 8: Windows Command for FP-Growth with Additional Options

```
cmd /k fpgrowth.exe -tr -el -v,%el
        -s10 -c80 basket.dat fpg.dat
```

The resulting association rules can be found in the output file (*fpg.dat*).

3.3.3 Post-processing

To have a better understanding of the data, it may be necessary to sort the rules according to the correlation measure of interest. I read the output file (*fpg.dat*) in R and converted it into a dataframe to enable sorting. The R code can be found in Listing 9.

Listing 9: Sorting Rules By Interest Measures

```
#Support and Confidence
#cmd /k fpgrowth.exe -tr basket.dat fpg.dat
FPG1 <- read.table("fpg.dat", sep = ",",
```

```
        col.names = c("rhs", "lhs", "support",
        "confidence"), header = FALSE,
        fill = TRUE)

#Lift
#cmd /k fpgrowth.exe -tr -el -v,%el
       -s10 -c80 basket.dat fpg2.dat
FPG2 <- read.table("fpg2.dat", sep = ",",
         col.names = c("rhs", "lhs", "lift"),
         header = FALSE, fill = TRUE)

FPG <- cbind(FPG1, FPG2[3])
FPG <- as.data.frame.matrix(FPG)
index<- with(FPG, order(FPG$confidence,
        decreasing=TRUE))

FPG[index,]
```

3.4 Summary

In this section, I present the complete R codes implemented for mining data using Apriori algorithm and FP-Growth Algorithm.

Listing 10: R Code for Data Mining using Apriori Algorithm

```
#Load arules package
library("arules")
library("arulesViz")

Basket <- read.transactions("basket.dat",
        sep=" ", rm.duplicates=TRUE)

#Generate Association Rules using Apriori Algo
rules <- apriori(Basket, parameter =
list(supp = 0.1, conf = 0.8))

rules.lhs.supp <- support(rules@lhs, Basket,
        type= c("relative", "absolute"),
        control = NULL)
rules.rhs.supp <- support(rules@rhs, Basket,
        type= c("relative", "absolute"),
```

```
        control = NULL)
rules.union.supp <- support(rules, Basket,
type= c("relative", "absolute"),
control = NULL)

#Caculate Imbalance Ratio
imbalance_ratio <- abs(rules.lhs.supp -
        rules.rhs.supp)/ (rules.lhs.supp +
        rules.rhs.supp - rules.union.supp)
quality(rules) <- cbind(quality(rules),
        imbalance_ratio)

#Calculate Kulc
kulc <- (1/2)*((rules.union.supp/rules.lhs.supp)
        + (rules.union.supp/rules.rhs.supp))
quality(rules) <- cbind(quality(rules), kulc)

#Pruning Redundant Rules
#Source: http://www.rdatamining.com/examples/
        association-rules
rules.sorted.lift <- sort(rules, by="lift")
subset.matrix <- is.subset(rules.sorted.lift,
        rules.sorted.lift)
subset.matrix[lower.tri(subset.matrix, diag=T)]
         <- NA
redundant <- colSums(subset.matrix, na.rm=T)
        >= 1
rules.pruned <- rules.sorted.lift[!redundant]

inspect(rules.pruned)
```

Listing 11: R Code for Data Mining using FP-Growth

```
#Execute the FP-growth C Implementation using the
#cmd /k fpgrowth.exe -tr basket.dat fpg.dat
FPG1 <- read.table("fpg.dat", sep = ",",
        col.names = c("rhs", "lhs", "support",
        "confidence"), header = FALSE,
        fill = TRUE)

#To obtain the lift, we run the FP-growth code
```

```
        again but with additional parameters
#cmd /k fpgrowth.exe -tr -el -v,%el -s10 -c80
        basket.dat fpg2.dat
FPG2 <- read.table("fpg2.dat", sep = ",",
        col.names = c("rhs", "lhs", "lift"),
        header = FALSE, fill = TRUE)

#Append lift
FPG <- cbind(FPG1, FPG2[3])
```

4 Experimental Results

Several sets of rules are produced through various experiments conducted on the market basket data set. To reiterate, the support count measures the percentage of transactions that contain all the itemset in a rule, and the confidence measures the certainty that purchasing one item may lead to purchasing another item. We can analyze the stength of a rule by the values of support and confidence. It is also important that the lift is greater than 1 to make sure that the occurrence of one event or *purchase* is positively correlated to another purchase.

4.1 Apriori Algorithm

The following trials or experiments conducted using the Apriori algorithm are as follows:

1. The support and confidence are set to 0.1 and 0.8, respectively. The rules are sorted by lift but without pruning the redundant rules. The Apriori algorithm produced a total of 172 association rules. The first five rules generated can be seen in Figure 3.

2. The support and confidence are set to 0.1 and 0.8 respectively. Resulting rules are sorted by *lift* and redundant rules are pruned. The result is a set of 81 rules. The first five rules with highest lifts can be seen in Figure 5 with the corresponding graph in Figure 6.

3. In this fourth experiment, I repeat the third experiment but this time, I sort the rules by *confidence*. See Figures 7 and 8.

4. Since we are interested in high-confidence, high-correlation association rules but not necessarily high-support rules, I set the support at 0.01

	lhs	rhs	support	confidence	lift	imbalance_ratio	kulc
1	{chicken, ice_crea, sardines}	=> {coke}	0.1158841	1.0000000	3.381757	0.6081081	0.6959459
2	{chicken, heineken, ice_crea, sardines}	=> {coke}	0.1158841	1.0000000	3.381757	0.6081081	0.6959459
3	{chicken, coke, heineken, ice_crea}	=> {sardines}	0.1158841	1.0000000	3.381757	0.6081081	0.6959459
4	{chicken, coke, heineken}	=> {sardines}	0.1158841	0.9830508	3.324439	0.5973154	0.6874714
5	{chicken, heineken, sardines}	=> {coke}	0.1158841	0.9747899	3.296502	0.5919732	0.6833409

Figure 3: Apriori Trial 1: minimal support = 0.1, minimal confidence = 0.8 (default settings) sorted by *lift* without pruning

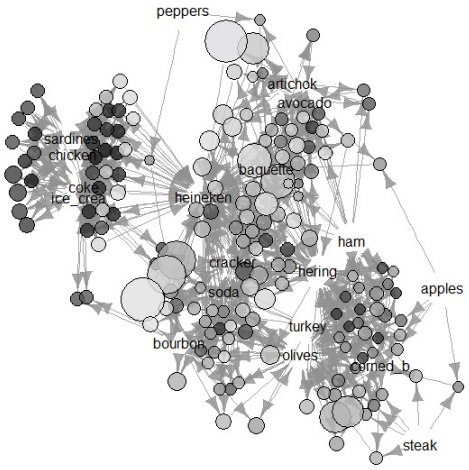

Figure 4: Apriori Trial 1 Graph

and confidence at 0.8 and sort the rules by lift. The resulting association rules can be seen in Figure 9. Because the Apriori algorithm produces a total of 879 association rules, I have omitted the graph as it is very cluttered.

5. The support is set at 0.1 and confidence is set at 0.8 and is sorted by increasing Imbalance Ratio. The resulting association rules can be seen in Figure 10.

6. The support is set at 0.1 and confidence is set at 0.8 and is sorted

	lhs	rhs	support	confidence	lift	imbalance_ratio	kulc
1	{chicken, ice_crea, sardines}	=> {coke}	0.1158841	1.0000000	3.381757	0.6081081	0.6959459
2	{chicken, coke, heineken}	=> {sardines}	0.1158841	0.9830508	3.324439	0.5973154	0.6874714
3	{chicken, heineken, ice_crea}	=> {sardines}	0.1158841	0.9747899	3.296502	0.5919732	0.6833409
4	{chicken, heineken, ice_crea}	=> {coke}	0.1158841	0.9747899	3.296502	0.5919732	0.6833409
5	{heineken, ice_crea, sardines}	=> {coke}	0.1168831	0.9512195	3.216793	0.5728477	0.6732449

Figure 5: Apriori Trial 2: minimal support = 0.1, minimal confidence = 0.8 (default settings) sorted by *lift* with pruning

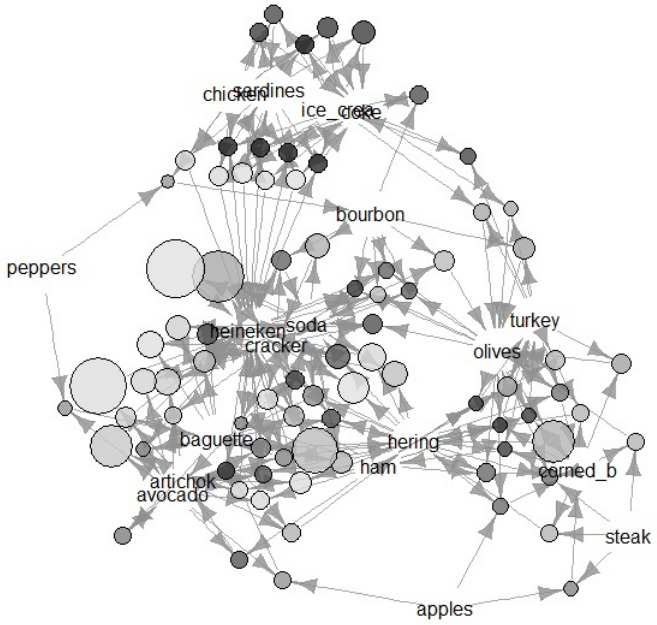

Figure 6: Apriori Trial 2 Graph

by decreasing support. The resulting association rules can be seen in Figure 11.

```
  lhs                                rhs          support confidence     lift imbalance_ratio      kulc
1 {chicken,
   ice_crea,
   sardines}                      => {coke}     0.1158841  1.0000000 3.381757       0.6081081 0.6959459
2 {artichok,
   avocado,
   cracker}                       => {heineken} 0.1118881  0.9911504 1.653569       0.8103161 0.5889086
3 {chicken,
   coke,
   heineken}                      => {sardines} 0.1158841  0.9830508 3.324439       0.5973154 0.6874714
4 {ham,
   hering,
   olives}                        => {corned_b} 0.1178821  0.9752066 2.496629       0.6852792 0.6384984
5 {chicken,
   heineken,
   ice_crea}                      => {sardines} 0.1158841  0.9747899 3.296502       0.5919732 0.6833409
```

Figure 7: Apriori Trial 3: minimal support = 0.1, minimal confidence = 0.8 (default settings) sorted by confidence with pruning

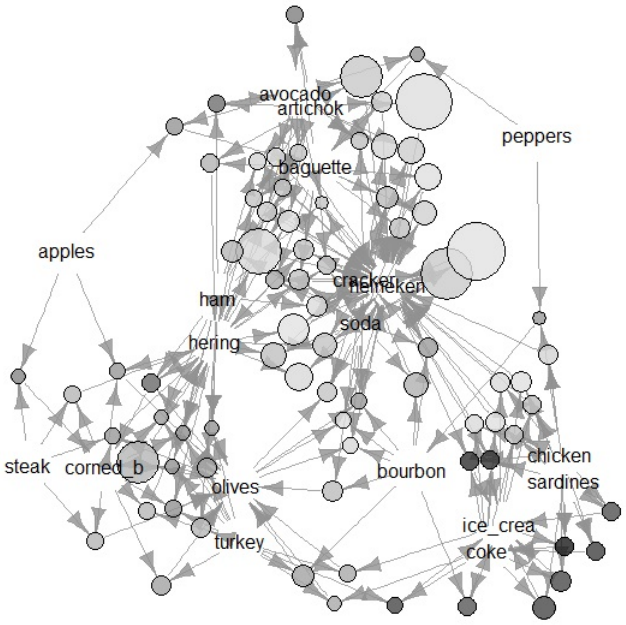

Figure 8: Apriori Trial 3 Graph

4.2 FP-Growth Algorithm

The following trials or experiments conducted using the FP-Growth algorithm are as follows:

1. Min support is set to 0.1, and min confidence is set to 0.8. Resulting rules are sorted by decreasing lift. The association rules can be found in Figure 12.

	lhs		rhs	support	confidence	lift	imbalance_ratio	kulc
1	{apples, corned_b, hering, sardines}	=>	{steak}	0.01198801	1.0000000	4.409692	0.9471366	0.5264317
2	{apples, hering, olives, sardines}	=>	{steak}	0.01198801	1.0000000	4.409692	0.9471366	0.5264317
3	{apples, artichok, corned_b, olives}	=>	{steak}	0.01398601	1.0000000	4.409692	0.9383260	0.5308370
4	{apples, artichok, hering, olives}	=>	{steak}	0.01398601	0.9333333	4.115712	0.9298246	0.4975037
5	{apples, corned_b, hering, ice_crea}	=>	{steak}	0.01398601	0.9333333	4.115712	0.9298246	0.4975037

Figure 9: Apriori Trial 4: minimal support = 0.01, minimal confidence = 0.8 sorted by *lift* with pruning

	lhs		rhs	support	confidence	lift	imbalance_ratio	kulc
1	{baguette, heineken}	=>	{hering}	0.2137862	0.8199234	1.688772	0.4221388	0.6301263
2	{soda}	=>	{heineken}	0.2567433	0.8081761	1.348307	0.4266263	0.6182547
3	{cracker, heineken, hering}	=>	{soda}	0.1368631	0.8404908	2.645696	0.4505814	0.6356542
4	{artichok}	=>	{heineken}	0.2517483	0.8262295	1.378426	0.4517611	0.6231148
5	{heineken, soda}	=>	{cracker}	0.2337662	0.9105058	1.867656	0.4520548	0.6950070
6	{ice_crea, sardines}	=>	{coke}	0.1318681	0.8741722	2.956238	0.4603175	0.6600591
7	{bourbon, ice_crea}	=>	{coke}	0.1168831	0.8068966	2.728728	0.4660494	0.6010834
8	{corned_b, olives}	=>	{hering}	0.2017982	0.8523207	1.755500	0.4779271	0.6339793
9	{bourbon, cracker, heineken}	=>	{soda}	0.1208791	0.8013245	2.522408	0.4798851	0.5909138
10	{chicken, ice_crea}	=>	{sardines}	0.1158841	0.8285714	2.802027	0.4875000	0.6102317

Figure 10: Apriori Trial 5: minimal support = 0.1, minimal confidence = 0.8 sorted by *increasing Imbalance Ratio* with pruning

2. Min support is set to 0.1, and min confidence is set to 0.8. Resulting rules are sorted by decreasing confidence. The association rules can be found in Figure 12.

	lhs		rhs	support	confidence	lift	imbalance_ratio	kulc
1	{soda}	=>	{heineken}	0.2567433	0.8081761	1.348307	0.4266263	0.6182547
2	{artichok}	=>	{heineken}	0.2517483	0.8262295	1.378426	0.4517611	0.6231148
3	{heineken, soda}	=>	{cracker}	0.2337662	0.9105058	1.867656	0.4520548	0.6950070
4	{baguette, heineken}	=>	{hering}	0.2137862	0.8199234	1.688772	0.4221388	0.6301263
5	{corned_b, olives}	=>	{hering}	0.2017982	0.8523207	1.755500	0.4779271	0.6339793
6	{artichok, avocado}	=>	{heineken}	0.1988012	0.9431280	1.573452	0.6356209	0.6373973
7	{cracker, hering}	=>	{heineken}	0.1628372	0.8109453	1.352927	0.6253918	0.5413060
8	{hering, soda}	=>	{heineken}	0.1488511	0.8563218	1.428630	0.6816000	0.5523276
9	{baguette, cracker}	=>	{heineken}	0.1448551	0.8192090	1.366714	0.6693038	0.5304379
10	{artichok, cracker}	=>	{heineken}	0.1438561	0.9171975	1.530191	0.7226754	0.5785987

Figure 11: Apriori Trial 6: minimal support = 0.1, minimal confidence = 0.8 sorted by *support* with pruning

rhs	lhs	support	confidence	lift
steak	apples corned_b olives hering	10.8891	88.9908	3.924221
steak	apples corned_b hering	11.2887	85.8407	3.785311
steak	apples olives hering	11.3886	85.0877	3.752111
steak	apples corned_b olives	11.5884	83.6207	3.687411
sardines	coke ice_crea chicken heineken	11.5884	100.0000	3.381761
coke	sardines ice_crea chicken heineken	11.5884	100.0000	3.381761
coke	sardines ice_crea chicken	11.5884	100.0000	3.381761
sardines	coke chicken heineken	11.7882	98.3051	3.324441
sardines	ice_crea chicken heineken	11.8881	97.4790	3.29651
coke	ice_crea chicken heineken	11.8881	97.4790	3.29651
coke	sardines chicken heineken	11.8881	97.4790	3.29651
artichok	ham avocado cracker heineken	10.0899	98.0198	3.216981
coke	sardines ice_crea heineken	12.2877	95.1220	3.216791
ice_crea	coke sardines chicken heineken	11.5884	100.0000	3.198081
sardines	coke ice_crea chicken	12.2877	94.3089	3.18931
ice_crea	coke sardines chicken	11.6883	99.1453	3.170751
chicken	coke sardines ice_crea heineken	11.6883	99.1453	3.150621
turkey	ice_crea bourbon olives	10.7892	88.8889	3.144091
ice_crea	coke chicken heineken	11.7882	98.3051	3.143881
ice_crea	coke sardines heineken	11.9880	97.5000	3.118131

Figure 12: FP-Growth Trial 1: minimal support = 0.1, minimal confidence = 0.8 sorted by *lift*

5 Analysis and Discussion of Results

As a data miner, my goal is to obtain rules for better decision support that may aid to increase company profit. In this section, I present the analysis of the experimental results produced by the Apriori algorithm and FP-Growth algorithm as seen in the previous section.

I analyze the rules depending on the level of support, confidence, lift, IR, and Kulc. Generally, we want high-confidence, high-correlation rules. For

rhs	lhs	support	confidence	lift
heineken	soda baguette hering cracker	11.4885	100.0000	1.668331
heineken	sardines ice_crea chicken	11.5884	100.0000	1.668331
heineken	coke sardines ice_crea chicken	11.5884	100.0000	1.668331
ice_crea	coke sardines chicken heineken	11.5884	100.0000	3.198081
sardines	coke ice_crea chicken heineken	11.5884	100.0000	3.381761
coke	sardines ice_crea chicken heineken	11.5884	100.0000	3.381761
coke	sardines ice_crea chicken	11.5884	100.0000	3.381761
olives	turkey ham corned_b hering	10.1898	100.0000	2.116281
heineken	soda baguette cracker	12.6873	99.2126	1.65521
heineken	coke sardines chicken	11.6883	99.1453	1.654071
chicken	coke sardines ice_crea heineken	11.6883	99.1453	3.150621
ice_crea	coke sardines chicken	11.6883	99.1453	3.170751
heineken	artichok avocado cracker	11.2887	99.1150	1.653571
olives	turkey corned_b hering	11.2887	99.1150	2.097551
cracker	soda bourbon olives heineken	10.6893	99.0654	2.032061
olives	turkey ham corned_b	10.4895	99.0476	2.096121
corned_b	turkey ham olives hering	10.2897	99.0291	2.535251
heineken	artichok ham cracker	10.0899	99.0099	1.651821
ice_crea	coke chicken heineken	11.7882	98.3051	3.143881
sardines	coke chicken heineken	11.7882	98.3051	3.324441

Figure 13: FP-Growth Trial 1: minimal support = 0.1, minimal confidence = 0.8 sorted by *confidence*

IR, the closer to zero, the more balanced the rules is. That is, if $IR = 0$, then $Conf(A \Rightarrow B) = Conf(B \Rightarrow A)$ (i.e. the implication goes both ways). On the other hand, Kulc, being the average of $Conf(A \Rightarrow B)$ and $Conf(B \Rightarrow A)$ is said to be neutral if equal to 50%.

5.1 Apriori Algorithm

- **Pruned Rules vs Unpruned Rules**
  The first experiment produced a set of 172 association rules sorted according to lift values. Observe that the second rule,

  $$\{chicken, heineken, ice\_cream, sardines\} \Rightarrow \{coke\}$$

  provides no addition knowledge because we know from the first rule that $\{chicken, ice\_cream, sardines\} \Rightarrow \{coke\}$.

  In other words, rule 2 is basically a redundant restatement of rule 1. Pruning redundant rules is done in the second trial which reduces the number of rules from 172 to 81. Pruning provides better quality of results which can be easily seen by comparing the graphs in Figure 4 and Figure 6. It is obvious that the graph of trial 2 (Figure 6) is less cluttered and provides a better visual representation of the rules.

- **High-confidence, High Correlation Rules**
  Sometimes, correct rules may not always have the highest lifts. For

this reason, I repeated the second experiment but this time, sorted the rules by *confidence* rather than lift. We find out that when the minimal support count is 0.1 and minimal confidence is 0.8, the rule with the highest confidence and highest correlation (in terms of lift) is still

$$\{chicken, ice\_cream, sardines\} \Rightarrow \{coke\}$$

. Based on the IR and Kulc values, this rule is imbalanced and somewhat neutral. This means that someone who purchases chicken, ice cream, and sardines is likely to purchase coke, but someone who purchases coke would not necessarily purchase chicken, ice cream, and sardines.

In the fourth trial, we uncover some high-confidence, high-correlation (albeit low-support) rules which include:

$$\{apples, corned\_beef, hering, sardines\} \Rightarrow \{steak\}$$
$$\{apples, hering, olives, sardines\} \Rightarrow \{steak\}$$
$$\{apples, artichoke, corned\_beef, olives\} \Rightarrow \{steak\}$$

Note that these rules are highly imbalanced, but neutral. For these high-confidence, high-correlation association rules, it may be a good idea for the retailer to send *coupons* of right-hand side items to customers who have purchased left-hand side itemsets. For example, a customer who has purchased apples, corned beef, hering, and sardines should be sent a coupon for steak.

- **Low IR, High Support Count**
  Since I was curious about the the rules with the least imbalance ratio (and therefore, are most balanced), I sorted the rules (min support = 0.1, min confidence = 0.8) according to increasing IR. The rules produced in Figure 10 are rules wherein, the purchase of the either itemset (regardless of whether lhs or rhs) may lead to the purchase of the other itemset. Notice that the rules with the lowest IR (Figure 10) are also amongst the rules with the highest support (Figure 11). These rules include:

$$\{soda\} \Rightarrow \{heineken\}$$
$$\{artichoke\} \Rightarrow \{heineken\}$$
$$\{baguette, heineken\} \Rightarrow \{hering\}$$
$$\{corned\_beef, olives\} \Rightarrow \{hering\}$$

$$\{artichoke, avocado\} \Rightarrow \{heineken\}$$

In a supermarket setting, it would be a good idea to place these items close together since they are very likely to be purchased together.

5.2 FP-Growth Algorithm

- **High-Confidence, High-Correlation Rules**
  Based on the first two experiments conducted using the FP-Growth Algorithm, there exist several high-confidence, high-correlation rules such as

$$\{sardines, icecream, chicken\} \Rightarrow \{coke\}$$
$$\{turkey, hame, cornedbeef, hering\} \Rightarrow \{olives\}$$
$$\{cokesardineschicken\} \Rightarrow \{icecream\}$$

  . Again, after discovering these rules, it may be advantageous for the company to send coupons or discounts for items in the right-hand side to customers who purchase itemsets in the left-hand side.

6 Conclusion

Mining association rules can be used for discovering interesting and useful connections between items or itemsets. In this programming assignment, I was able to mine a market basket dataset using Apriori Algorithm and FP-Growth Algorithm and analyze the resulting association rules based on several interest measures.

Generally, we would want to obtain high-confidence, high-correlation rules. These kinds of rules can be seen in Figures 7,9, 12, and 13. Using information from these high-confidence, high-correlation rules, companies could benefit by sending coupons or discounts for a given item to customers who are likely to purchase that item. For rules with low imbalance ratio and high support, as seen in Figures 10 and 11, it may also be advantageous to place these items near each other in a supermarket setting since customers are likely to purchase them together.

Through this programming assignment, I have achieved my objectives (as stated in section 2) and have demonstrated that mining association rules can indeed be useful in real-world applications.

References

[1] Frequent item set mining. http://www.borgelt.net//fpgrowth.html. Last accessed 21 March 2015.

[2] R data mining association rules. http://www.rdatamining.com/examples/association-rules. Last accessed 21 March 2015.

[3] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[4] Christian Borgelt. An implementation of the fp-growth algorithm. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 1–5. ACM, 2005.

[5] Keith E. Emmert. Mining frequent patterns, associations, and correlations: Basic concepts and methods. http://faculty.tarleton.edu/emmert/Teaching/Fall2012/DataMiningI/PatternMiningAssocia Last accessed 22 March 2015.

[6] Michael Hahsler, Bettina Grün, and Kurt Hornik. A computational environment for mining association rules and frequent item sets. 2005.

[7] Michael Hahsler and Kurt Hornik. Building on the arules infrastructure for analyzing transaction data with r. In *Advances in Data Analysis*, pages 449–456. Springer, 2007.

[8] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1):53–87, 2004.

[9] Sudheer Chelluboina Michael Hahsler. Visualizing association rules: Introduction to the r-extension package arulesviz. http://cran.r-project.org/web/packages/arulesViz/vignettes/arulesViz.pdfPatternMiningAssociaitonsCor Last accessed 22 March 2015.

[10] María N Moreno, Saddys Segrera, Vivian F López, and M José Polo. Improving the quality of association rules by preprocessing numerical data.

[11] Akash Rajak and Mahendra Kumar Gupta. Association rule mining-applications in various areas. In *Proceedings of International Conference on Data Management, Ghaziabad, India*, pages 3–7, 2008.