Intro to Data Science - Lab 10

Week 10 - Association Rules Mining

```
# Enter your name here: Hendi Kushta
```

Please include nice comments.

Instructions:

Run the necessary code on your own instance of R-Studio.

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this lab assignment by myself, with help from the book and the professor.
```

Association rules mining, also known as **market basket analysis**, is an **unsupervised data mining technique** that discovers patterns in the form of if-then rules. The technique is ** unsupervised ** in the sense that there is no prediction or classification happening. We are simply trying to find interesting **patterns**. In addition to working with baskets of objects, association rules mining is good at working with any kind of data that can be expressed as **lists of attributes**. For example, a trip to Washington DC might consist of the following attributes: train, July, morning departure, afternoon arrival, Union Station, first class, express.

In these exercises we will work with a built in data set called **groceries**. Make sure to library the **arules** and **arulesViz** packages before running the following:

```
data (Groceries) # Load data into memory
myGroc <- Groceries # Make a copy for safety
```

```
# install.packages("arules")
# install.packages("arulesViz")

library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

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

```
library(arulesViz)
data (Groceries) # Load data into memory
myGroc <- Groceries # Make a copy for safety
```

1. Examine the data structure that **summary()** reveals. This is called a **sparse matrix** and it efficiently stores a set of market baskets along with meta-data. Report using R comments about some of the item labels.

```
summary(myGroc)
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
```

```
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
##   17   18   19   20   21   22   23   24   26   27   28   29   32
##   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##        labels  level2           level1
## 1 frankfurter sausage meat and sausage
## 2     sausage sausage meat and sausage
## 3  liver loaf sausage meat and sausage
```

```
# most frequent items are whole milk, vegetables, rolls/buns, sode,
# yogurt. There is no basket without any item. minimum 1 item in a
# basket and maximum 32. We can see that whole milk is bought more
# frequently.
```

2. Use the **itemFrequency(myGroc)** command to generate a list of item frequencies. Save that list in a new data object. Run **str( )** on the data object and write a comment describing what it is. Run **sort( )** on the data object and save the results. Run **head( )** and **tail( )** on the sorted object to show the most and least frequently occurring items. What s the most frequently purchased item?

```
newData <- itemFrequency(myGroc)
str(newData)
```

```
##  Named num [1:169] 0.05897 0.09395 0.00508 0.02603 0.02583 ...
##  - attr(*, "names")= chr [1:169] "frankfurter" "sausage" "liver loaf" "ham" ...
```

```
# it shows the frequency of an item being purchased.
# some of the items include "frankfurter" "sausage" "liver loaf" "ham" etc.
newData <- sort(newData)
head(newData)
```

```
##             baby food  sound storage medium preservation products
##          0.0001016777          0.0001016777          0.0002033554
##       kitchen utensil                  bags        frozen chicken
##          0.0004067107          0.0004067107          0.0006100661
```

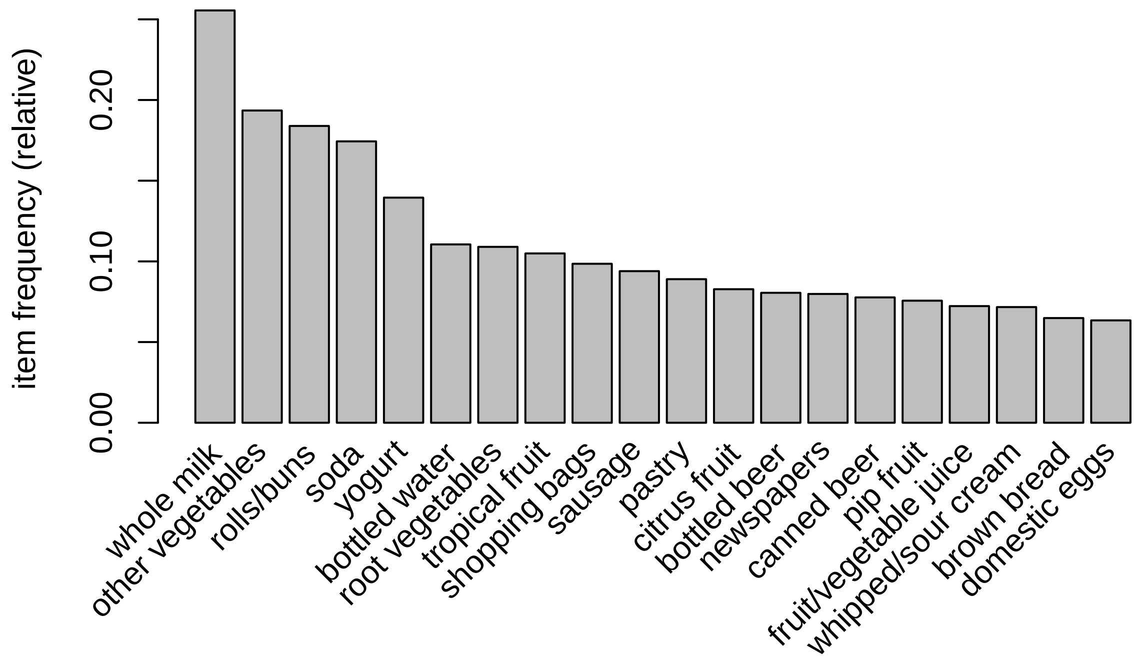
```
tail(newData)
```

```
##    bottled water           yogurt             soda       rolls/buns
##        0.1105236        0.1395018        0.1743772        0.1839349
## other vegetables       whole milk
##        0.1934926        0.2555160
```

```
# the most frequent purchased item is whole milk.
```

3. Create a frequency plot with `itemFrequencyPlot(myGroc, topN=20)` and confirm that the plot shows the most frequently purchased item with the left-most bar. Write a comment describing the meaning of the Y-axis.

```
itemFrequencyPlot(myGroc, topN=20)
```




```
# it shows the relative frequency of occurrence of different items
# in the matrix for example the whole milk appears 2513 out of 9835 rows
# which means, 2513 / 9835 = 0.2555... or approximately 25.5% of the rows.
```

4. Create a cross table with `ct <-  crossTable(myGroc,  sort=TRUE)`. Examine the first few rows and columns of **ct** by using the square brackets subsetting technique. For example, the first two rows and first three columns would be `ct[1:2,    1:3]`. Write a comment describing one of values. Write a comment describing what is on the diagonal of the matrix.

```
ct  <-  crossTable(myGroc,  sort=TRUE)
ct[1:2, 1:3]
```

```
##                    whole milk other vegetables rolls/buns
## whole milk               2513              736        557
## other vegetables          736             1903        419
```

```
# the intersection of the same items with one another create a diagonal,
# which shows the real number of how many times the item is shown in the basket.
# the intersection of items with one another for example other vegetables and whole
# milk 736 means that we have bought vegetables and whole milk together this many times.
```

5. Run the following analysis:

```
rules1 <- apriori(myGroc,
 parameter=list(supp=0.0008, conf=0.55),
 control=list(verbose=F),
 appearance=list(default="lhs",rhs=("bottled beer")))
```

```
rules1 <- apriori(myGroc,
 parameter=list(supp=0.0008, conf=0.55),
 control=list(verbose=F),
```

```
 appearance=list(default="lhs",rhs=("bottled beer")))
```

6. Examine the resulting rule set with **inspect( )** and make sense of the results. There should be four rules in total.

```
inspect(rules1)
```

```
##     lhs                                rhs            support      confidence
## [1] {liquor, red/blush wine}        => {bottled beer} 0.0019318760 0.9047619
## [2] {soda, liquor}                  => {bottled beer} 0.0012201322 0.5714286
## [3] {red/blush wine, napkins}       => {bottled beer} 0.0008134215 0.5714286
## [4] {soda, liquor, red/blush wine} => {bottled beer} 0.0008134215 1.0000000
##     coverage     lift     count
## [1] 0.0021352313 11.23527 19
## [2] 0.0021352313  7.09596 12
## [3] 0.0014234875  7.09596  8
## [4] 0.0008134215 12.41793  8
```

```
# There are huge chances that the items in the 4 lists, are bought together.
# confidence level, lift is very high in rule 1 and 4
# for rule 2 and 3 the confidence level is above 50%
```

7. Adjust the **support** parameter to a new value so that you get more rules. Anywhere between 10 and 30 rules would be fine. Examine the new rule set with **inspect( )**. Does your interpretation of the situation still make sense?

```
rules2 <- apriori(myGroc,
 parameter=list(supp=0.0005, conf=0.55),
 control=list(verbose=F),
 appearance=list(default="lhs",rhs=("bottled beer")))
inspect(rules2)
```

```
##      lhs                        rhs                   support confidence     coverage      lift  count
## [1]  {liquor (appetizer),
##       dishes}                => {bottled beer} 0.0006100661   0.8571429 0.0007117438 10.643939      6
## [2]  {liquor,
##       red/blush wine}        => {bottled beer} 0.0019318760   0.9047619 0.0021352313 11.235269     19
## [3]  {soda,
##       liquor}                => {bottled beer} 0.0012201322   0.5714286 0.0021352313  7.095960     12
## [4]  {red/blush wine,
##       napkins}               => {bottled beer} 0.0008134215   0.5714286 0.0014234875  7.095960      8
## [5]  {soda,
##       liquor,
##       red/blush wine}        => {bottled beer} 0.0008134215   1.0000000 0.0008134215 12.417929      8
## [6]  {whole milk,
##       soups,
##       bottled water}         => {bottled beer} 0.0005083884   0.8333333 0.0006100661 10.348274      5
## [7]  {yogurt,
##       pastry,
##       flower (seeds)}        => {bottled beer} 0.0005083884   0.8333333 0.0006100661 10.348274      5
## [8]  {whole milk,
##       yogurt,
##       flower (seeds)}        => {bottled beer} 0.0005083884   0.7142857 0.0007117438  8.869949      5
## [9]  {other vegetables,
##       salt,
##       margarine}             => {bottled beer} 0.0005083884   0.7142857 0.0007117438  8.869949      5
```

```
## [10] {soda,
##       red/blush wine,
##       napkins}             => {bottled beer} 0.0005083884  0.8333333 0.0006100661 10.348274     5
## [11] {citrus fruit,
##       oil,
##       bottled water}       => {bottled beer} 0.0005083884  0.5555556 0.0009150991  6.898850     5
## [12] {root vegetables,
##       herbs,
##       other vegetables,
##       bottled water}       => {bottled beer} 0.0006100661  0.6000000 0.0010167768  7.450758     6
## [13] {whole milk,
##       butter,
##       rolls/buns,
##       napkins}             => {bottled beer} 0.0005083884  0.5555556 0.0009150991  6.898850     5
## [14] {pork,
##       whole milk,
##       domestic eggs,
##       rolls/buns}          => {bottled beer} 0.0005083884  0.5555556 0.0009150991  6.898850     5
```

```r
# now there are more rules, but still, the confidence level and lift are high.
```

8. Power User (not required): use **mtcars** to create a new data frame with **factors** (e.g., cyl attribute). Then create an mpg column with good or bad (good MPG is above 25). Convert the data frame to a transactions dataset and then predict rules for having bad MPG.

```r
# data("mtcars")
# mtcars.t <- data.frame(mtcars)
# mtcars.t$mpg <- as.factor(mtcars.t$mpg)
# mtcars.t$cyl <- as.factor(mtcars.t$cyl)
# mtcars.t$disp <- as.factor(mtcars.t$disp)
# mtcars.t$hp <- as.factor(mtcars.t$hp)
# mtcars.t$drat <- as.factor(mtcars.t$drat)
# mtcars.t$wt <- as.factor(mtcars.t$wt)
# mtcars.t$qsec <- as.factor(mtcars.t$qsec)
# mtcars.t$vs <- as.factor(mtcars.t$vs)
# mtcars.t$am <- as.factor(mtcars.t$am)
# mtcars.t$gear <- as.factor(mtcars.t$gear)
# mtcars.t$carb <- as.factor(mtcars.t$carb)

# mtcars.t$goodOrBadMpg <- with(mtcars.t, ifelse(mpg > 25, 'Good', 'Bad'))
# str(mtcars.t)

# mtcars.t$mpg <- as.factor(mtcars.t$mpg)
# mtcars.t$goodOrBadMpg <- as.factor(mtcars.t$goodOrBadMpg)

# df_test <- mtcars.t[, c("mpg", "cyl", "goodOrBadMpg")]
# trans_df_test = as(df_test, "transactions")
# apriori(trans_df_test, parameter=list(supp=0.0004, conf=0.3))

# trans_mtcars.t = as(mtcars.t, "transactions")

# rulesMtgBadOrGood <- apriori(trans_mtcars.t, parameter=list(supp=0.0004, conf=0.3))

myCars <- mtcars
myCars <- data.frame(cyl = as.factor(mtcars$cyl),
```

```r
                     gear = as.factor(mtcars$gear),
                     goodMpg = as.factor(mtcars$mpg > 25))

myCars$goodMpg
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE TRUE  TRUE  TRUE  FALSE FALSE FALSE FALSE
## [25] FALSE TRUE  TRUE  TRUE  FALSE FALSE FALSE FALSE
## Levels: FALSE TRUE
```

```r
rules1 <- apriori(myCars, control = list(verbose=F),
                  parameter = list(supp=0.0001, conf=0.05),
                  appearance = list(default="lhs", rhs=("goodMpg=FALSE")))

inspect(rules1)
```

```
##      lhs                 rhs             support confidence coverage lift
## [1]  {}               => {goodMpg=FALSE} 0.81250 0.8125000  1.00000  1.0000000
## [2]  {gear=5}         => {goodMpg=FALSE} 0.09375 0.6000000  0.15625  0.7384615
## [3]  {cyl=6}          => {goodMpg=FALSE} 0.21875 1.0000000  0.21875  1.2307692
## [4]  {cyl=4}          => {goodMpg=FALSE} 0.15625 0.4545455  0.34375  0.5594406
## [5]  {gear=4}         => {goodMpg=FALSE} 0.25000 0.6666667  0.37500  0.8205128
## [6]  {cyl=8}          => {goodMpg=FALSE} 0.43750 1.0000000  0.43750  1.2307692
## [7]  {gear=3}         => {goodMpg=FALSE} 0.46875 1.0000000  0.46875  1.2307692
## [8]  {cyl=6, gear=5} => {goodMpg=FALSE} 0.03125 1.0000000  0.03125  1.2307692
## [9]  {cyl=8, gear=5} => {goodMpg=FALSE} 0.06250 1.0000000  0.06250  1.2307692
## [10] {cyl=6, gear=4} => {goodMpg=FALSE} 0.12500 1.0000000  0.12500  1.2307692
## [11] {cyl=6, gear=3} => {goodMpg=FALSE} 0.06250 1.0000000  0.06250  1.2307692
## [12] {cyl=4, gear=4} => {goodMpg=FALSE} 0.12500 0.5000000  0.25000  0.6153846
## [13] {cyl=4, gear=3} => {goodMpg=FALSE} 0.03125 1.0000000  0.03125  1.2307692
## [14] {cyl=8, gear=3} => {goodMpg=FALSE} 0.37500 1.0000000  0.37500  1.2307692
##      count
## [1]  26
## [2]   3
## [3]   7
## [4]   5
## [5]   8
## [6]  14
## [7]  15
## [8]   1
## [9]   2
## [10]  4
## [11]  2
## [12]  4
## [13]  1
## [14] 12
```