Association Analysis

Brian Michira

9/10/2021

Research Question

To create association rules that will allow you to identify relationships between variables in the dataset.Being provided with a separate dataset that comprises groups of items that will be associated with others.

```
# Loading the arules library
suppressWarnings(
        suppressMessages(if
                         (!require(arules, quietly=TRUE))
                 install.packages("arules")))
library(arules)
```

```
#Load the data and preview the head
path="http://bit.ly/SupermarketDatasetII"
supermarket <-read.transactions(path, sep = ",")
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

```
# Verifying the object's class
class(supermarket)
```

```
## [1] "transactions"
## attr(,"package")
## [1] "arules"
```

```
# Previewing our first 10 transactions
inspect(supermarket[1:10])
```

```
##      items
## [1]  {almonds,
##       antioxydant juice,
##       avocado,
##       cottage cheese,
##       energy drink,
##       frozen smoothie,
##       green grapes,
##       green tea,
##       honey,
```

```
##       low fat yogurt,
##       mineral water,
##       olive oil,
##       salad,
##       salmon,
##       shrimp,
##       spinach,
##       tomato juice,
##       vegetables mix,
##       whole weat flour,
##       yams}
## [2]  {burgers,
##       eggs,
##       meatballs}
## [3]  {chutney}
## [4]  {avocado,
##       turkey}
## [5]  {energy bar,
##       green tea,
##       milk,
##       mineral water,
##       whole wheat rice}
## [6]  {low fat yogurt}
## [7]  {french fries,
##       whole wheat pasta}
## [8]  {light cream,
##       shallot,
##       soup}
## [9]  {frozen vegetables,
##       green tea,
##       spaghetti}
## [10] {french fries}
```

```
# Creating a dataframe for the items
items<-as.data.frame(itemLabels(supermarket))
colnames(items) <- "Item"
head(items, 10)
```

```
##                 Item
## 1            almonds
## 2  antioxydant juice
## 3          asparagus
## 4            avocado
## 5        babies food
## 6              bacon
## 7     barbecue sauce
## 8          black tea
## 9        blueberries
## 10        body spray
```

```
# Generating a summary of the transaction dataset
summary(supermarket)
```

```
## transactions as itemMatrix in sparse format with
```

```
##  7501 rows (elements/itemsets/transactions) and
##  119 columns (items) and a density of 0.03288973
##
## most frequent items:
## mineral water          eggs     spaghetti  french fries     chocolate
##          1788          1348          1306          1282          1229
##       (Other)
##         22405
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1754 1358 1044  816  667  493  391  324  259  139  102   67   40   22   17    4
##   18   19   20
##    1    2    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.914   5.000  20.000
##
## includes extended item information - examples:
##              labels
## 1           almonds
## 2 antioxydant juice
## 3         asparagus
```

There are 7501 transactions in our dataset.The most purchased items were mineral water, eggs, spaghetti, french fries and chocolate.

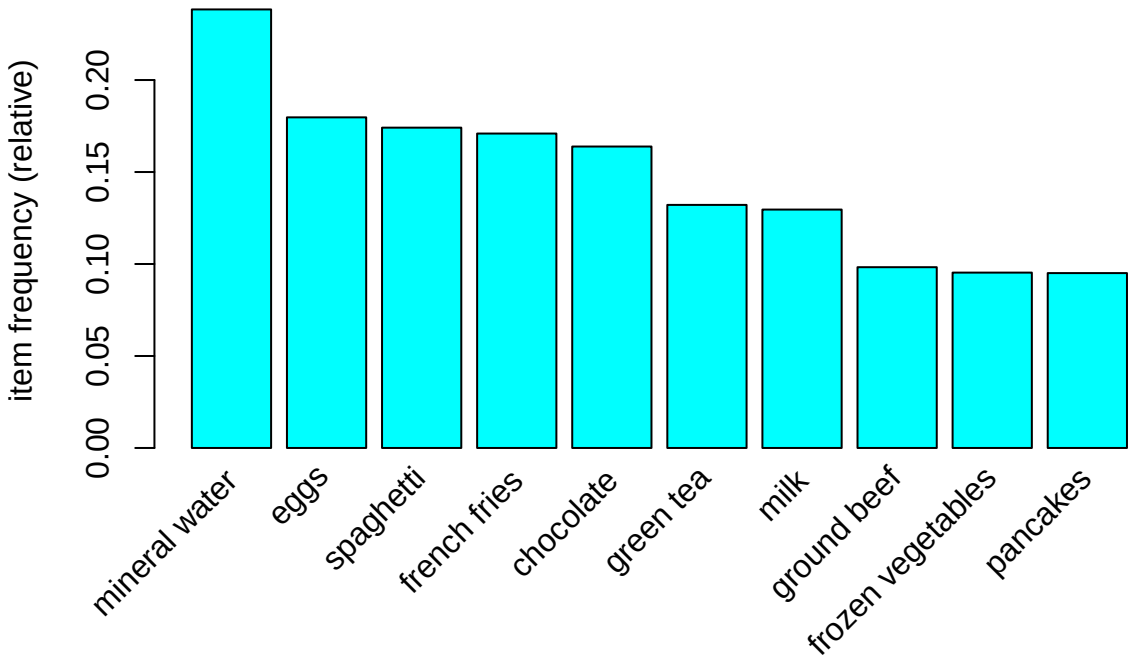
```
# Exploring the frequency of some articles and checking the transaction percentages of the first 10 ite
itemFrequency(supermarket[, 1:10],type = "absolute")
```

```
##           almonds antioxydant juice         asparagus           avocado
##               153                67                36               250
##       babies food             bacon    barbecue sauce         black tea
##                34                65                81               107
##       blueberries        body spray
##                69                86
```

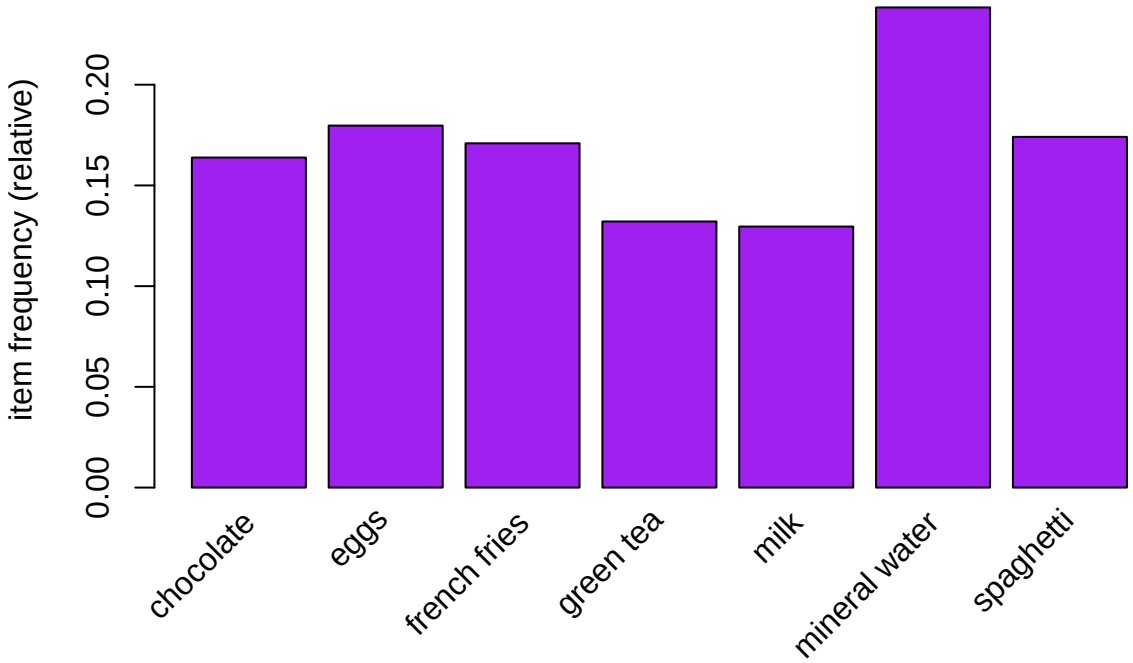
```
round(itemFrequency(supermarket[, 1:10],type = "relative")*100,2)
```

```
##           almonds antioxydant juice         asparagus           avocado
##              2.04              0.89              0.48              3.33
##       babies food             bacon    barbecue sauce         black tea
##              0.45              0.87              1.08              1.43
##       blueberries        body spray
##              0.92              1.15
```

```
# plot the frequency of items
# Displaying top 10 most common items in the transactions dataset
itemFrequencyPlot(supermarket, topN = 10,col="cyan")
```




```
# Frequency items whose relative importance is at least 10%
itemFrequencyPlot(supermarket, support = 0.1,col="purple")
```




```
# Building a model based on association rules using the apriori function
# We use Min Support as 0.001 and confidence as 0.8

rules <- apriori (supermarket, parameter = list(supp = 0.001, conf = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5   0.001      1
##  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: 7
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [116 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [74 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules
```

```
## set of 74 rules
```

We found a set of 74 rules with a Min support of 0.001 and a onfidence of 0.8.

```
# However, in order to illustrate the sensitivity of the model to these two parameters, we will see wha

# Building a apriori model with Min Support as 0.002 and confidence as 0.8.
rules2 <- apriori (supermarket,parameter = list(supp = 0.002, conf = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5   0.002      1
##  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: 15
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [115 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [2 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules2
```

```
## set of 2 rules
```

Increasing the Min support we obtain a set of 2 rules.

```
# Building apriori model with Min Support as 0.002 and confidence as 0.6.
rules3 <- apriori (supermarket, parameter = list(supp = 0.001, conf = 0.6))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.6    0.1    1 none FALSE            TRUE       5   0.001      1
##  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: 7
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [116 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [545 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules3
```

```
## set of 545 rules
```

Using a lower confidence of 0.6 we obtained 545 rules.

From the analysis we can see that using a higher level of support can make a mode lose interesting rules and reducing the confidence level increases the number of rules to quite an extent and many will not be useful.

```
# Check the summaries of the rules
summary(rules)
```

```
## set of 74 rules
##
## rule length distribution (lhs + rhs):sizes
##  3  4  5  6
## 15 42 16  1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   4.000   4.000   4.041   4.000   6.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001067   Min.   :0.8000   Min.   :0.001067   Min.   : 3.356
##  1st Qu.:0.001067   1st Qu.:0.8000   1st Qu.:0.001333   1st Qu.: 3.432
##  Median :0.001133   Median :0.8333   Median :0.001333   Median : 3.795
##  Mean   :0.001256   Mean   :0.8504   Mean   :0.001479   Mean   : 4.823
##  3rd Qu.:0.001333   3rd Qu.:0.8889   3rd Qu.:0.001600   3rd Qu.: 4.877
##  Max.   :0.002533   Max.   :1.0000   Max.   :0.002666   Max.   :12.722
##      count
##  Min.   : 8.000
##  1st Qu.: 8.000
##  Median : 8.500
##  Mean   : 9.419
##  3rd Qu.:10.000
##  Max.   :19.000
##
## mining info:
##         data ntransactions support confidence
##  supermarket          7501   0.001        0.8
```

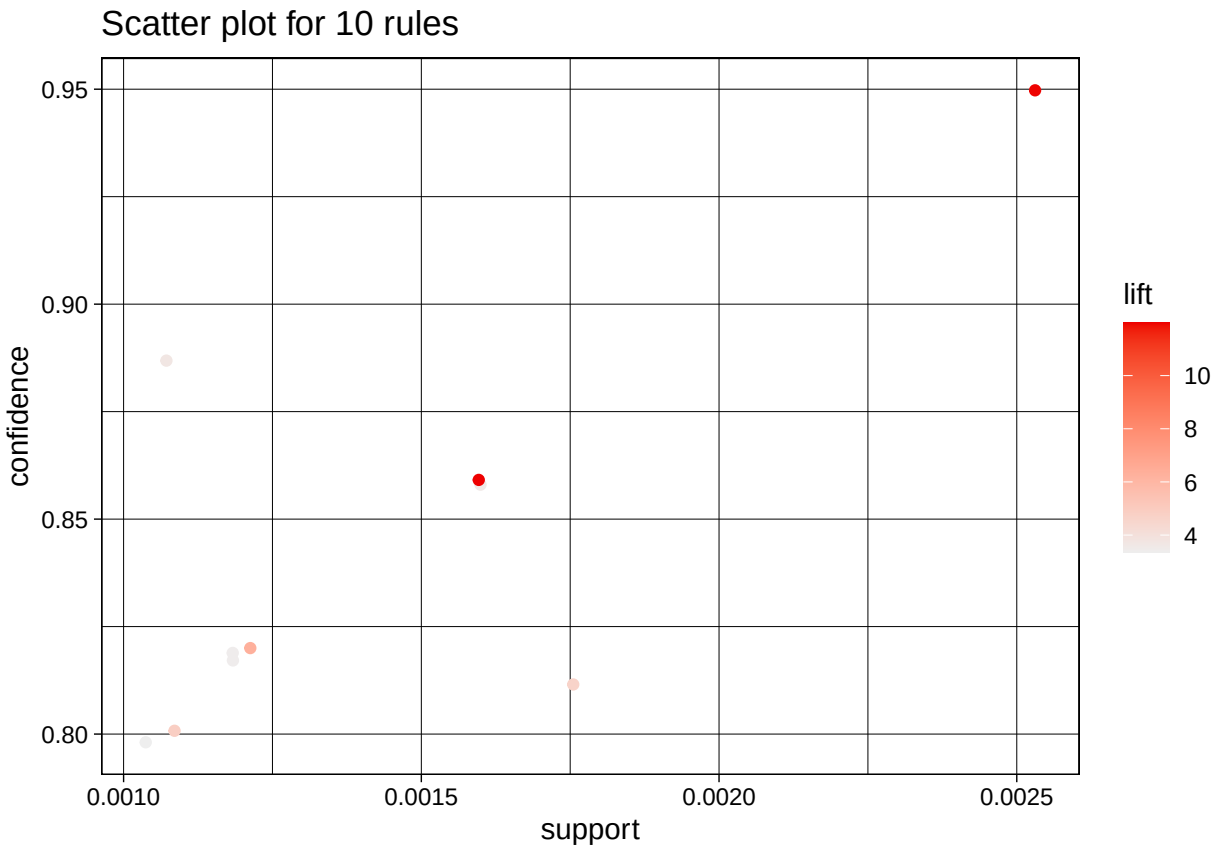
```
# Observing rules built in our model i.e. first 10 model rules
inspect(rules[1:10])
```

```
##      lhs                              rhs             support     confidence
## [1]  {frozen smoothie,spinach}     => {mineral water} 0.001066524 0.8888889
## [2]  {bacon,pancakes}              => {spaghetti}     0.001733102 0.8125000
## [3]  {nonfat milk,turkey}          => {mineral water} 0.001199840 0.8181818
## [4]  {ground beef,nonfat milk}     => {mineral water} 0.001599787 0.8571429
## [5]  {mushroom cream sauce,pasta} => {escalope}      0.002532996 0.9500000
## [6]  {milk,pasta}                  => {shrimp}        0.001599787 0.8571429
## [7]  {cooking oil,fromage blanc}   => {mineral water} 0.001199840 0.8181818
## [8]  {black tea,salmon}            => {mineral water} 0.001066524 0.8000000
## [9]  {black tea,frozen smoothie}   => {milk}          0.001199840 0.8181818
## [10] {red wine,tomato sauce}       => {chocolate}     0.001066524 0.8000000
##      coverage    lift      count
## [1]  0.001199840  3.729058  8
## [2]  0.002133049  4.666587 13
## [3]  0.001466471  3.432428  9
## [4]  0.001866418  3.595877 12
## [5]  0.002666311 11.976387 19
## [6]  0.001866418 11.995203 12
## [7]  0.001466471  3.432428  9
## [8]  0.001333156  3.356152  8
## [9]  0.001466471  6.313973  9
## [10] 0.001333156  4.882669  8
```

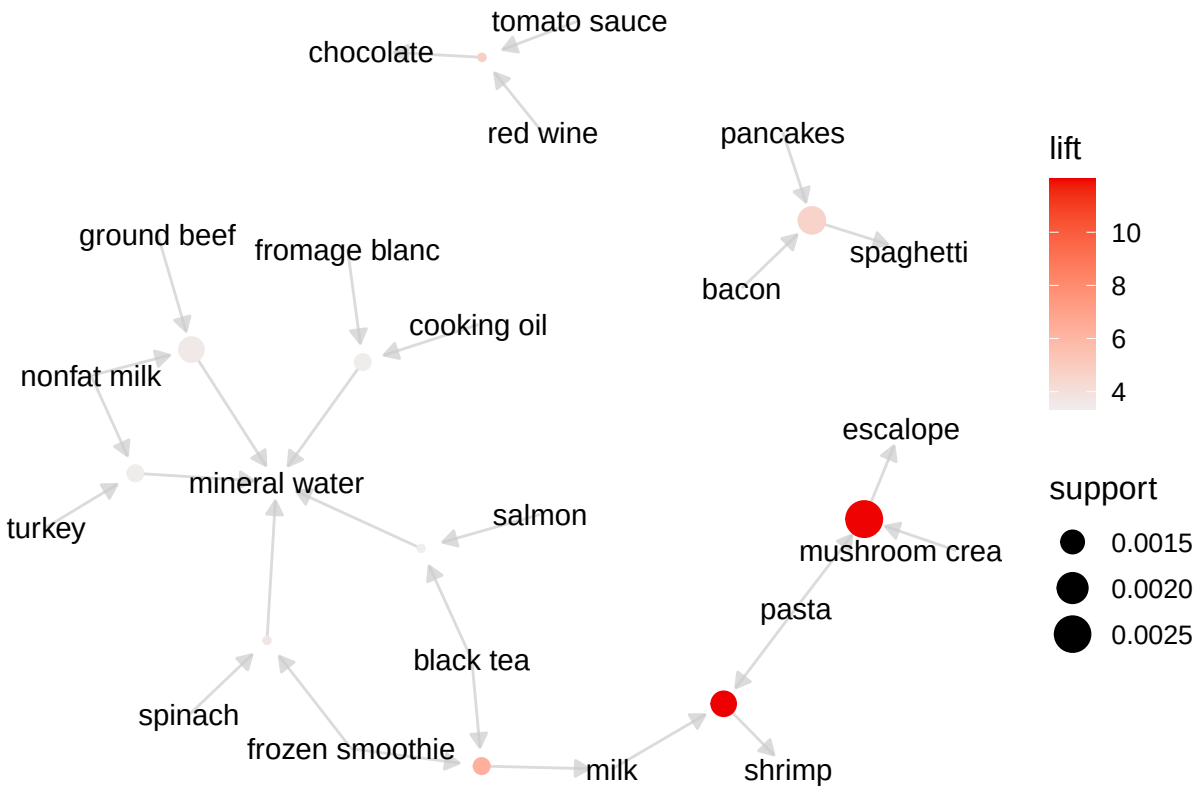
If a customer buys frozen smoothie and spinach there is 88% confidence of him or her purchasing mineral water or if one milk and pasta there is 85% confidence of him or her purchasing shrimp.

```
#plotting the top ten rules
library(arulesViz)
topRules <- rules[1:10]
plot(topRules)
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```




```
plot(topRules, method="graph")
```



The most porpular transaction was mushroom cream sauce and pasta.

```
# Ordering these rules by a criteria such as the level of confidence then looking at the first five rul

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

```
##     lhs                                                rhs             support
## [1] {french fries,mushroom cream sauce,pasta} => {escalope}      0.001066524
## [2] {ground beef,light cream,olive oil}       => {mineral water} 0.001199840
## [3] {cake,meatballs,mineral water}            => {milk}          0.001066524
## [4] {cake,olive oil,shrimp}                   => {mineral water} 0.001199840
## [5] {mushroom cream sauce,pasta}              => {escalope}      0.002532996
##     confidence coverage    lift      count
## [1] 1.00       0.001066524 12.606723  8
## [2] 1.00       0.001199840  4.195190  9
## [3] 1.00       0.001066524  7.717078  8
## [4] 1.00       0.001199840  4.195190  9
## [5] 0.95       0.002666311 11.976387 19
```

```
#ordering by lift
rules<-sort(rules, by="lift", decreasing=TRUE)
inspect(rules[1:5])
```

```
##     lhs                       rhs                    support confidence    coverage      lift count
## [1] {eggs,
```

```
##      mineral water,
##      pasta}                => {shrimp}            0.001333156  0.9090909 0.001466471 12.722185    10
## [2] {french fries,
##      mushroom cream sauce,
##      pasta}                => {escalope}          0.001066524  1.0000000 0.001066524 12.606723     8
## [3] {milk,
##      pasta}                => {shrimp}            0.001599787  0.8571429 0.001866418 11.995203    12
## [4] {mushroom cream sauce,
##      pasta}                => {escalope}          0.002532996  0.9500000 0.002666311 11.976387    19
## [5] {chocolate,
##      ground beef,
##      milk,
##      mineral water,
##      spaghetti}            => {frozen vegetables} 0.001066524  0.8888889 0.001199840  9.325253     8
```

```
#ordering by support
rules<-sort(rules, by="support", decreasing=TRUE)
inspect(rules[1:5])
```

```
##     lhs                       rhs                support confidence    coverage      lift count
## [1] {mushroom cream sauce,
##      pasta}                => {escalope}      0.002532996  0.9500000 0.002666311 11.976387    19
## [2] {frozen vegetables,
##      olive oil,
##      tomatoes}             => {spaghetti}     0.002133049  0.8421053 0.002532996  4.836624    16
## [3] {red wine,
##      soup}                 => {mineral water} 0.001866418  0.9333333 0.001999733  3.915511    14
## [4] {frozen vegetables,
##      olive oil,
##      shrimp}               => {mineral water} 0.001866418  0.8235294 0.002266364  3.454862    14
## [5] {frozen vegetables,
##      ground beef,
##      mineral water,
##      shrimp}               => {spaghetti}     0.001733102  0.8666667 0.001999733  4.977693    13
```

#Promotion

```
# If we're interested in making a promotion relating to the sale of chocolate,
# we could create a subset of rules concerning these products

spaghetti <- subset(rules, subset = rhs %pin% "spaghetti")

# Then order by confidence
spaghetti<-sort(spaghetti, by="confidence", decreasing=TRUE)
inspect(spaghetti[1:5])
```

```
##     lhs                    rhs             support confidence    coverage     lift count
## [1] {light cream,
##      mineral water,
##      shrimp}            => {spaghetti} 0.001066524  0.8888889 0.001199840 5.105326     8
## [2] {ground beef,
##      salmon,
##      shrimp}            => {spaghetti} 0.001066524  0.8888889 0.001199840 5.105326     8
```

```
## [3] {burgers,
##      milk,
##      salmon}            => {spaghetti} 0.001066524  0.8888889 0.001199840 5.105326     8
## [4] {frozen vegetables,
##      ground beef,
##      mineral water,
##      shrimp}            => {spaghetti} 0.001733102  0.8666667 0.001999733 4.977693    13
## [5] {burgers,
##      frozen vegetables,
##      pancakes}          => {spaghetti} 0.001466471  0.8461538 0.001733102 4.859877    11
```

```
# What if we wanted to determine items that customers might buy who have previously bought spaghetti?
# Subset the rules
spaghetti <- subset(rules, subset = lhs %pin% "spaghetti")

# Order by confidence
spaghetti<-sort(spaghetti, by="confidence", decreasing=TRUE)

# inspect top 5
inspect(spaghetti[1:5])
```

```
##     lhs                    rhs                     support confidence    coverage     lift count
## [1] {frozen vegetables,
##      milk,
##      spaghetti,
##      turkey}            => {mineral water}     0.001199840  0.9000000 0.001333156 3.775671     9
## [2] {chocolate,
##      ground beef,
##      milk,
##      mineral water,
##      spaghetti}         => {frozen vegetables} 0.001066524  0.8888889 0.001199840 9.325253     8
## [3] {black tea,
##      spaghetti,
##      turkey}            => {eggs}              0.001066524  0.8888889 0.001199840 4.946258     8
## [4] {chocolate,
##      frozen vegetables,
##      shrimp,
##      spaghetti}         => {mineral water}     0.001733102  0.8666667 0.001999733 3.635831    13
## [5] {frozen vegetables,
##      milk,
##      shrimp,
##      spaghetti}         => {mineral water}     0.001466471  0.8461538 0.001733102 3.549776    11
```