Practice 7

Smit Patil

7/13/2020

Problem 1.

Build an R Notebook of the concrete strength example in the textbook on pages 232 to 239. Show each step and add appropriate documentation.

```r
#Importing Concrete data
concrete <- read.csv("concrete.csv")

#Looking at the headand the structure of the data
head(concrete)
```

```
##   cement  slag ash water superplastic coarseagg fineagg age strength
## 1  540.0   0.0   0   162          2.5    1040.0   676.0  28    79.99
## 2  540.0   0.0   0   162          2.5    1055.0   676.0  28    61.89
## 3  332.5 142.5   0   228          0.0     932.0   594.0 270    40.27
## 4  332.5 142.5   0   228          0.0     932.0   594.0 365    41.05
## 5  198.6 132.4   0   192          0.0     978.4   825.5 360    44.30
## 6  266.0 114.0   0   228          0.0     932.0   670.0  90    47.03
```

```r
str(concrete)
```

```
## 'data.frame':    1030 obs. of  9 variables:
##  $ cement      : num  540 540 332 332 199 ...
##  $ slag        : num  0 0 142 142 132 ...
##  $ ash         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ water       : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ superplastic: num  2.5 2.5 0 0 0 0 0 0 0 0 ...
##  $ coarseagg   : num  1040 1055 932 932 978 ...
##  $ fineagg     : num  676 676 594 594 826 ...
##  $ age         : int  28 28 270 365 360 90 365 28 28 28 ...
##  $ strength    : num  80 61.9 40.3 41 44.3 ...
```

```r
#Creating a function for normalization
normalize <- function(x)
  {
    return((x - min(x)) / (max(x) - min(x)))
  }

#Normalizing the concrete data
concrete_norm <- as.data.frame(lapply(concrete, normalize))
```

```r
#Looking at the summary of the normalized concrete strength and the original concrete strength data
summary(concrete_norm$strength)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2664  0.4001  0.4172  0.5457  1.0000
```

```r
summary(concrete$strength)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.33   23.71   34.45   35.82   46.13   82.60
```

```r
#Importing libraries for training the model
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 4.0.2
```

```r
#Creating a training and testing data sets from the normalized concrete data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

#Training the model
concrete_model <- neuralnet(strength ~ cement + slag + ash + water + superplastic + coarseagg + fineagg

#Plotting the trained model
plot(concrete_model)

#Evaluating the model performance
model_results <- compute(concrete_model, concrete_test[1:8])

#Predicting the strength from the testing data
predicted_strength <- model_results$net.result

#Calculating the correlation
cor(predicted_strength, concrete_test$strength)
```

```
##           [,1]
## [1,] 0.7190348
```

```r
#Improving the model performance
concrete_model2 <- neuralnet(strength ~ cement + slag + ash + water + superplastic + coarseagg + fineag

#Plotting the graph for the second model
plot(concrete_model2)

#Evaluating the performance of the second model
model_results2 <- compute(concrete_model2, concrete_test[1:8])

#Predicting the strength for the second model
predicted_strength2 <- model_results2$net.result

#Calculating the correlation
cor(predicted_strength2, concrete_test$strength)
```

```
##             [,1]
## [1,] 0.7911617
```

Problem 2.

Build an R Notebook of the optical character recognition example in the textbook on pages 249 to 257. Show each step and add appropriate documentation.

```r
#Importing Libraries
library(kernlab)

#Importing letters data
letters <- read.csv("letterdata.csv")

#Viewing at the head and the structure of the letters data
head(letters)
```

```
##   letter xbox ybox width height onpix xbar ybar x2bar y2bar xybar x2ybar xy2bar
## 1      T    2    8     3      5     1    8   13     0     6     6     10      8
## 2      I    5   12     3      7     2   10    5     5     4    13      3      9
## 3      D    4   11     6      8     6   10    6     2     6    10      3      7
## 4      N    7   11     6      6     3    5    9     4     6     4      4     10
## 5      G    2    1     3      1     1    8    6     6     6     6      5      9
## 6      S    4   11     5      8     3    8    8     6     9     5      6      6
##   xedge xedgey yedge yedgex
## 1     0      8     0      8
## 2     2      8     4     10
## 3     3      7     3      9
## 4     6     10     2      8
## 5     1      7     5     10
## 6     0      8     9      7
```

```r
str(letters)
```

```
## 'data.frame':    20000 obs. of  17 variables:
##  $ letter: chr  "T" "I" "D" "N" ...
##  $ xbox  : int  2 5 4 7 2 4 4 1 2 11 ...
##  $ ybox  : int  8 12 11 11 1 11 2 1 2 15 ...
##  $ width : int  3 3 6 6 3 5 5 3 4 13 ...
##  $ height: int  5 7 8 6 1 8 4 2 4 9 ...
##  $ onpix : int  1 2 6 3 1 3 4 1 2 7 ...
##  $ xbar  : int  8 10 10 5 8 8 8 8 10 13 ...
##  $ ybar  : int  13 5 6 9 6 8 7 2 6 2 ...
##  $ x2bar : int  0 5 2 4 6 6 6 2 2 6 ...
##  $ y2bar : int  6 4 6 6 6 9 6 2 6 2 ...
##  $ xybar : int  6 13 10 4 6 5 7 8 12 12 ...
##  $ x2ybar: int  10 3 3 4 5 6 6 2 4 1 ...
##  $ xy2bar: int  8 9 7 10 9 6 6 8 8 9 ...
##  $ xedge : int  0 2 3 6 1 0 2 1 1 8 ...
##  $ xedgey: int  8 8 7 10 7 8 8 6 6 1 ...
##  $ yedge : int  0 4 3 2 5 9 7 2 1 1 ...
##  $ yedgex: int  8 10 9 8 10 7 10 7 7 8 ...
```

```r
#Creating factors for the letter column in the letters dataset
letters$letter <- as.factor(letters$letter)

#Spliting the data for training and testing
letters_train <- letters[1:16000, ]
letters_test <- letters[16001:20000, ]

#Building a classifier using the kernlab package
letter_classifier <- ksvm(letter ~ ., data = letters_train, kernel = "vanilladot")
```

```
##  Setting default kernel parameters
```

```r
#Printing the letters_classifier
letter_classifier
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 7037
##
## Objective Function Value : -14.1746 -20.0072 -23.5628 -6.2009 -7.5524 -32.7694 -49.9786 -18.1824 -62
## Training error : 0.130062
```

```r
#Evaluating the performance of the model
letter_predictions <- predict(letter_classifier, letters_test)

#Looking at the head of the letters prediction
head(letter_predictions)
```

```
## [1] U N V X N H
## Levels: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

```r
#Creating a table of predicted vs actual values
table(letter_predictions, letters_test$letter)
```

```
##
## letter_predictions   A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
##                  A 144   0   0   0   0   0   0   0   0   1   0   0   1   2   2
##                  B   0 121   0   5   2   0   1   2   0   0   1   0   1   0   0
##                  C   0   0 120   0   4   0  10   2   2   0   1   3   0   0   2
##                  D   2   2   0 156   0   1   3  10   4   3   4   3   0   5   5
##                  E   0   0   5   0 127   3   1   1   0   0   3   4   0   0   0
##                  F   0   0   0   0   0 138   2   2   6   0   0   0   0   0   0
##                  G   1   1   2   1   9   2 123   2   0   0   1   2   1   0   1
##                  H   0   0   0   1   0   1   0 102   0   2   3   2   3   4  20
##                  I   0   1   0   0   0   1   0   0 141   8   0   0   0   0   0
##                  J   0   1   0   0   0   1   0   2   5 128   0   0   0   0   1
```

```
##                   K   1   1   9   0   0   0   2   5   0   0 118   0   0   2   0
##                   L   0   0   0   0   2   0   1   1   0   0   0 133   0   0   0
##                   M   0   0   1   1   0   0   1   1   0   0   0   0 135   4   0
##                   N   0   0   0   0   0   1   0   1   0   0   0   0   0 145   0
##                   O   1   0   2   1   0   0   1   2   0   1   0   0   0   1  99
##                   P   0   0   0   1   0   2   1   0   0   0   0   0   0   0   2
##                   Q   0   0   0   0   0   0   8   2   0   0   0   3   0   0   3
##                   R   0   7   0   0   1   0   3   8   0   0  13   0   0   1   1
##                   S   1   1   0   0   1   0   3   0   1   1   0   1   0   0   0
##                   T   0   0   0   0   3   2   0   0   0   0   1   0   0   0   0
##                   U   1   0   3   1   0   0   0   2   0   0   0   0   0   0   1
##                   V   0   0   0   0   0   1   3   4   0   0   0   0   1   2   1
##                   W   0   0   0   0   0   0   1   0   0   0   0   0   2   0   0
##                   X   0   1   0   0   2   0   0   1   3   0   1   6   0   0   1
##                   Y   3   0   0   0   0   0   0   1   0   0   0   0   0   0   0
##                   Z   2   0   0   0   1   0   0   0   3   4   0   0   0   0   0
##
## letter_predictions   P   Q   R   S   T   U   V   W   X   Y   Z
##                   A   0   5   0   1   1   1   0   1   0   0   1
##                   B   2   2   3   5   0   0   2   0   1   0   0
##                   C   0   0   0   0   0   0   0   0   0   0   0
##                   D   3   1   4   0   0   0   0   0   3   3   1
##                   E   0   2   0  10   0   0   0   0   2   0   3
##                   F  16   0   0   3   0   0   1   0   1   2   0
##                   G   2   8   2   4   3   0   0   0   1   0   0
##                   H   0   2   3   0   3   0   2   0   0   1   0
##                   I   1   0   0   3   0   0   0   0   5   1   1
##                   J   1   3   0   2   0   0   0   0   1   0   6
##                   K   1   0   7   0   1   3   0   0   5   0   0
##                   L   0   1   0   5   0   0   0   0   0   0   1
##                   M   0   0   0   0   0   3   0   8   0   0   0
##                   N   0   0   3   0   0   1   0   2   0   0   0
##                   O   3   3   0   0   0   3   0   0   0   0   0
##                   P 130   0   0   0   0   0   0   0   0   1   0
##                   Q   1 124   0   5   0   0   0   0   0   2   0
##                   R   1   0 138   0   1   0   1   0   0   0   0
##                   S   0  14   0 101   3   0   0   0   2   0  10
##                   T   0   0   0   3 133   1   0   0   0   2   2
##                   U   0   0   0   0   0 152   0   0   1   1   0
##                   V   0   3   1   0   0   0 126   1   0   4   0
##                   W   0   0   0   0   0   4   4 127   0   0   0
##                   X   0   0   0   1   0   0   0   0 137   1   1
##                   Y   7   0   0   0   3   0   0   0   0 127   0
##                   Z   0   0   0  18   3   0   0   0   0   0 132
```

```r
#Creating a logical vector of the predicted value to the actual values
agreement <- letter_predictions == letters_test$letter

#Creating a table for the logiacl vector
table(agreement)
```

```
## agreement
## FALSE  TRUE
##   643  3357
```

```r
#Creating a table with their probability for the logical vector
prop.table(table(agreement))
```

```
## agreement
##   FALSE    TRUE
## 0.16075 0.83925
```

```r
#Improving the model performance
letter_classifier_rbf <- ksvm(letter ~ ., data = letters_train, kernel = "rbfdot")

#Evaluating the performance of the new model
letter_predictions_rbf <- predict(letter_classifier_rbf, letters_test)

#Creating a logical vector of the predicted value to the actual values for the new model
agreement_rbf <- letter_predictions_rbf == letters_test$letter

#Creating a table for the logiacl vector of the new model
table(agreement_rbf)
```

```
## agreement_rbf
## FALSE  TRUE
##   282  3718
```

```r
#Creating a table with their probability for the logical vector of the new model
prop.table(table(agreement_rbf))
```

```
## agreement_rbf
##  FALSE   TRUE
## 0.0705 0.9295
```

Problem 3.

Build an R Notebook of the grocery store transactions example in the textbook on pages 266 to 284. Show each step and add appropriate documentation.

```r
#Importing libraries
library(arules)
```

```
## Warning: package 'arules' was built under R version 4.0.2

## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following object is masked from 'package:kernlab':
##
##     size

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

```r
#Importing groceries data
groceries <- read.transactions("groceries.csv", sep = ",")

#Summarising the data
summary(groceries)
```

```
## 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
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

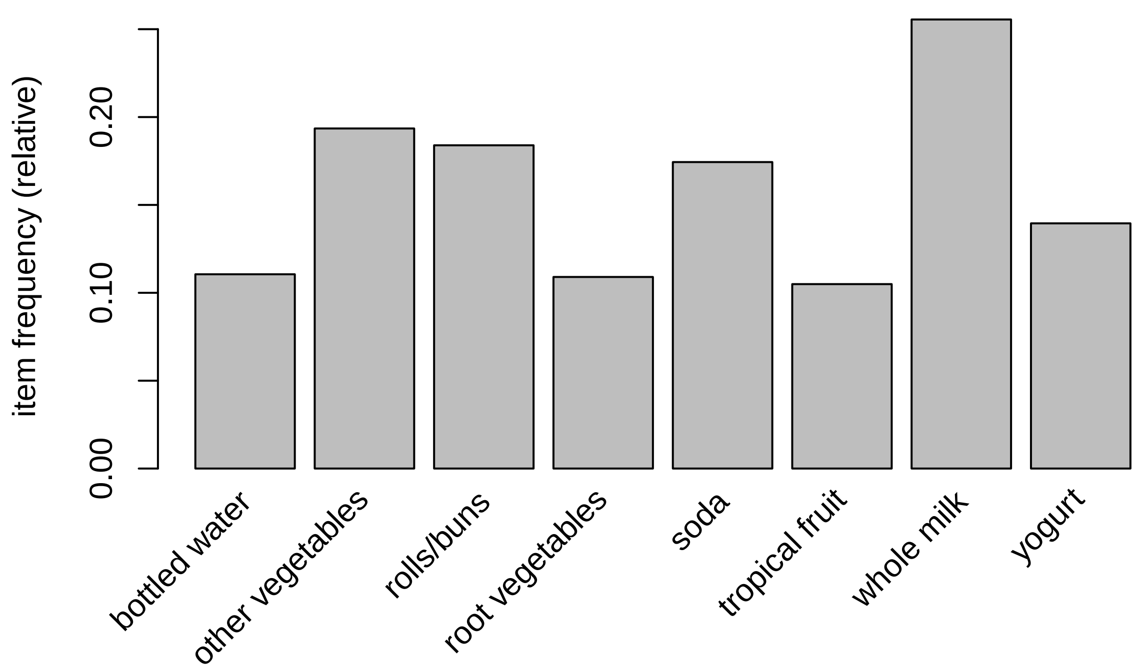
```r
#Inspecting the first 5 rows of the groceries data
inspect(groceries[1:5])
```

```
##     items
## [1] {citrus fruit,
##      margarine,
##      ready soups,
##      semi-finished bread}
## [2] {coffee,
##      tropical fruit,
##      yogurt}
## [3] {whole milk}
## [4] {cream cheese,
##      meat spreads,
##      pip fruit,
##      yogurt}
## [5] {condensed milk,
##      long life bakery product,
##      other vegetables,
##      whole milk}
```

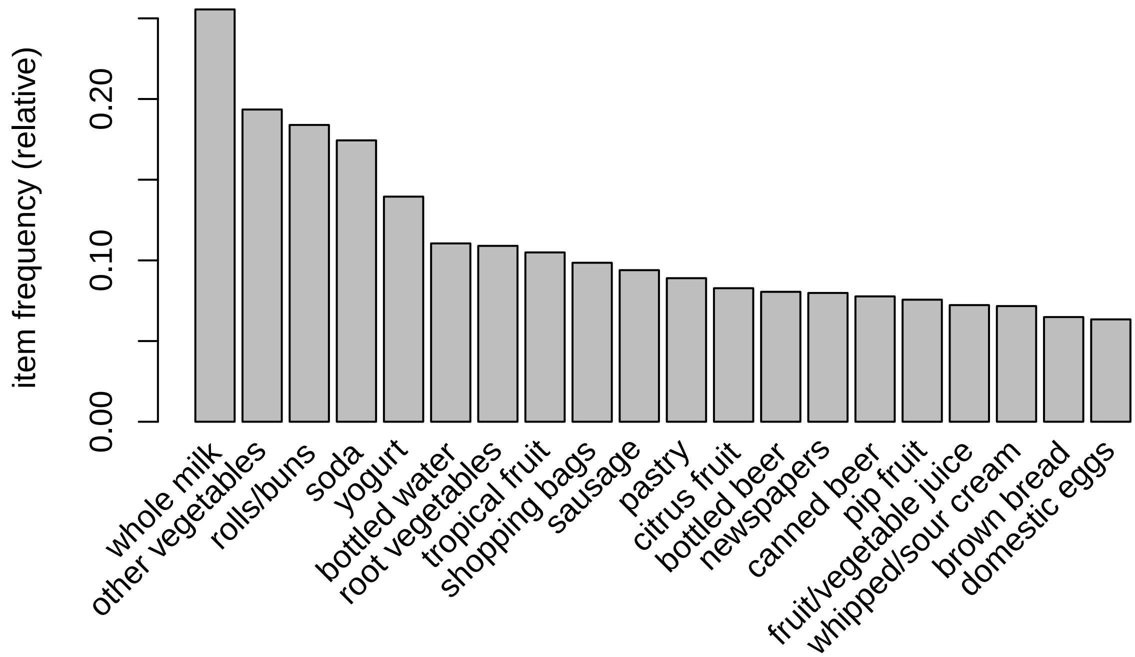
```r
#Calculating the frequency of the first 3 items
itemFrequency(groceries[, 1:3])
```

```
## abrasive cleaner artif. sweetener   baby cosmetics
##     0.0035587189     0.0032536858     0.0006100661
```

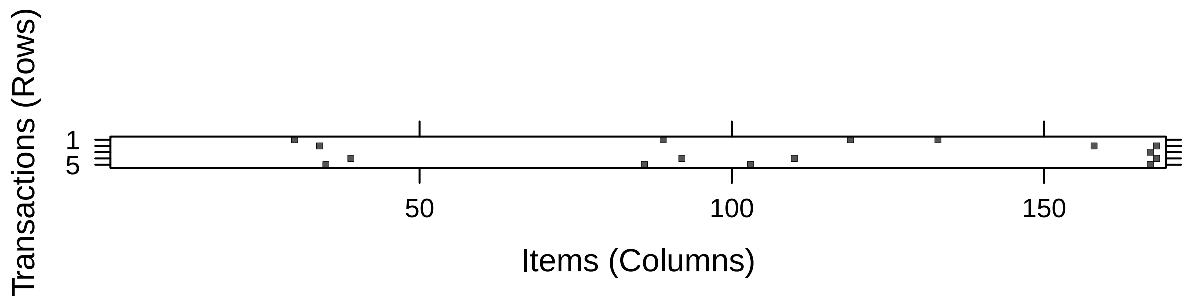
```r
#Plotting the ietms with atleast 10% support
itemFrequencyPlot(groceries, support = 0.1)
```



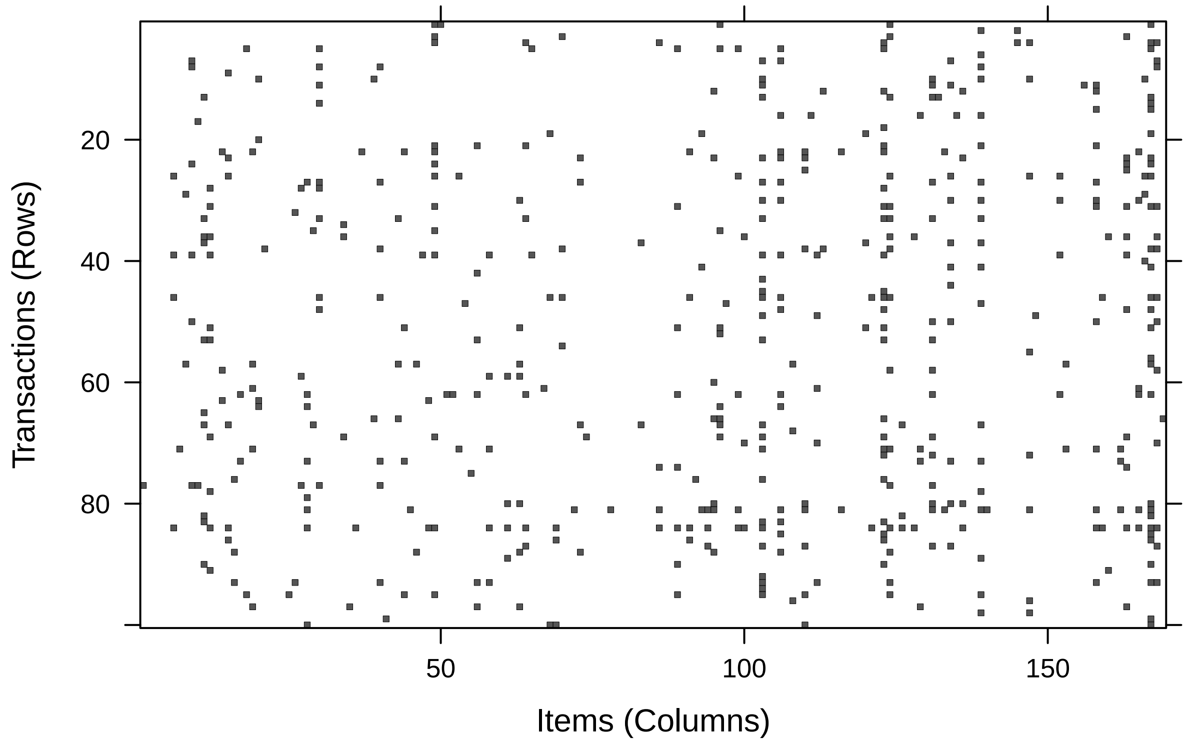

```r
#Plotting the top 20 ietms
itemFrequencyPlot(groceries, topN = 20)
```




```r
#Creating a sparse matrix for the first five transactions
image(groceries[1:5])
```




```r
#Creating a sparse matrix for a random sample of 100 transactions
image(sample(groceries, 100))
```




```r
#Calculating rules using the default values of the apriori function
apriori(groceries)
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      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: 983
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [8 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

## set of 0 rules
```

```r
#Calculating rules using custom values
groceryrules <- apriori(groceries, parameter = list(support = 0.006, confidence = 0.25, minlen = 2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5   0.006      2
##  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: 59
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [109 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [463 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
groceryrules
```

```
## set of 463 rules
```

```r
#Summarising the new rules
summary(groceryrules)
```

```
## set of 463 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4
## 150 297  16
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.711   3.000   4.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.006101   Min.   :0.2500   Min.   :0.009964   Min.   :0.9932
##  1st Qu.:0.007117   1st Qu.:0.2971   1st Qu.:0.018709   1st Qu.:1.6229
##  Median :0.008744   Median :0.3554   Median :0.024809   Median :1.9332
##  Mean   :0.011539   Mean   :0.3786   Mean   :0.032608   Mean   :2.0351
##  3rd Qu.:0.012303   3rd Qu.:0.4495   3rd Qu.:0.035892   3rd Qu.:2.3565
##  Max.   :0.074835   Max.   :0.6600   Max.   :0.255516   Max.   :3.9565
##      count
##  Min.   : 60.0
##  1st Qu.: 70.0
```

```
##   Median : 86.0
##   Mean   :113.5
##   3rd Qu.:121.0
##   Max.   :736.0
##
## mining info:
##       data ntransactions support confidence
##  groceries          9835   0.006       0.25
```

```r
#Inspecting the first 3 rules of the grocery rules
inspect(groceryrules[1:3])
```

```
##     lhs             rhs                support     confidence coverage
## [1] {pot plants} => {whole milk}       0.006914082 0.4000000  0.01728521
## [2] {pasta}      => {whole milk}       0.006100661 0.4054054  0.01504830
## [3] {herbs}      => {root vegetables} 0.007015760 0.4312500  0.01626843
##     lift     count
## [1] 1.565460 68
## [2] 1.586614 60
## [3] 3.956477 69
```

```r
#Combining the sort function with the inspect function to find the best five rules according to the lif
inspect(sort(groceryrules, by = "lift")[1:5])
```

```
##     lhs                   rhs                     support confidence   coverage     lift count
## [1] {herbs}            => {root vegetables}    0.007015760  0.4312500 0.01626843 3.956477    69
## [2] {berries}          => {whipped/sour cream} 0.009049314  0.2721713 0.03324860 3.796886    89
## [3] {other vegetables,
##      tropical fruit,
##      whole milk}       => {root vegetables}    0.007015760  0.4107143 0.01708185 3.768074    69
## [4] {beef,
##      other vegetables} => {root vegetables}    0.007930859  0.4020619 0.01972547 3.688692    78
## [5] {other vegetables,
##      tropical fruit}   => {pip fruit}          0.009456024  0.2634561 0.03589222 3.482649    93
```

```r
#Using subset function to find any rules with berries appearing in the rule
berryrules <- subset(groceryrules, items %in% "berries")
inspect(berryrules)
```

```
##     lhs          rhs                  support     confidence coverage  lift
## [1] {berries} => {whipped/sour cream} 0.009049314 0.2721713  0.0332486 3.796886
## [2] {berries} => {yogurt}             0.010574479 0.3180428  0.0332486 2.279848
## [3] {berries} => {other vegetables}   0.010269446 0.3088685  0.0332486 1.596280
## [4] {berries} => {whole milk}         0.011794611 0.3547401  0.0332486 1.388328
##     count
## [1]  89
## [2] 104
## [3] 101
## [4] 116
```

```r
#Saving association rules to a file or data frame
write(groceryrules, file = "groceryrules.csv", sep = ",", quote = TRUE, row.names = FALSE)
groceryrules_df <- as(groceryrules, "data.frame")

#Looking at the structure of the newly created data frame
str(groceryrules_df)
```

```
## 'data.frame':    463 obs. of  6 variables:
##  $ rules     : chr  "{pot plants} => {whole milk}" "{pasta} => {whole milk}" "{herbs} => {root veget
##  $ support   : num  0.00691 0.0061 0.00702 0.00773 0.00773 ...
##  $ confidence: num  0.4 0.405 0.431 0.475 0.475 ...
##  $ coverage  : num  0.0173 0.015 0.0163 0.0163 0.0163 ...
##  $ lift      : num  1.57 1.59 3.96 2.45 1.86 ...
##  $ count     : int  68 60 69 76 76 69 70 67 63 88 ...
```