M4_L4_RomilShah

Romil Shah

June 19, 2016

Read Data and additional packages

```
require(arules)
```

```
## Loading required package: arules
```

```
## Warning: package 'arules' was built under R version 3.2.5
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 3.2.5
```

```
##
## Attaching package: 'arules'
```

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

```
require(arulesViz)
```

```
## Loading required package: arulesViz
```

```
## Warning: package 'arulesViz' was built under R version 3.2.5
```

```
## Loading required package: grid
```

```
require(Matrix)

urlData <- 'http://fimi.ua.ac.be/data/retail.dat'
retail <- read.transactions(url(urlData))
summary(retail)
```

```
## transactions as itemMatrix in sparse format with
##  88162 rows (elements/itemsets/transactions) and
##  16470 columns (items) and a density of 0.0006257289
##
## most frequent items:
##      39      48      38      32      41 (Other)
##   50675   42135   15596   15167   14945  770058
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 3016 5516 6919 7210 6814 6163 5746 5143 4660 4086 3751 3285 2866 2620 2310
```

```
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
## 2115 1874 1645 1469 1290 1205  981  887  819  684  586  582  472  480  355
##   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45
##  310  303  272  234  194  136  153  123  115  112   76   66   71   60   50
##   46   47   48   49   50   51   52   53   54   55   56   57   58   59   60
##   44   37   37   33   22   24   21   21   10   11   10    9   11    4    9
##   61   62   63   64   65   66   67   68   71   73   74   76
##    7    4    5    2    2    5    3    3    1    1    1    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    4.00    8.00   10.31   14.00   76.00
##
## includes extended item information - examples:
##   labels
## 1      0
## 2      1
## 3     10
```

```
# First few transactions
inspect(retail[1:20])
```

```
##    items
## 1  {0,
##     1,
##     10,
##     11,
##     12,
##     13,
##     14,
##     15,
##     16,
##     17,
##     18,
##     19,
##     2,
##     20,
##     21,
##     22,
##     23,
##     24,
##     25,
##     26,
##     27,
##     28,
##     29,
##     3,
##     4,
##     5,
##     6,
##     7,
```

```
##     8,
##     9}
## 2  {30,
##     31,
##     32}
## 3  {33,
##     34,
##     35}
## 4  {36,
##     37,
##     38,
##     39,
##     40,
##     41,
##     42,
##     43,
##     44,
##     45,
##     46}
## 5  {38,
##     39,
##     47,
##     48}
## 6  {38,
##     39,
##     48,
##     49,
##     50,
##     51,
##     52,
##     53,
##     54,
##     55,
##     56,
##     57,
##     58}
## 7  {32,
##     41,
##     59,
##     60,
##     61,
##     62}
## 8  {3,
##     39,
##     48}
## 9  {63,
##     64,
##     65,
##     66,
##     67,
```

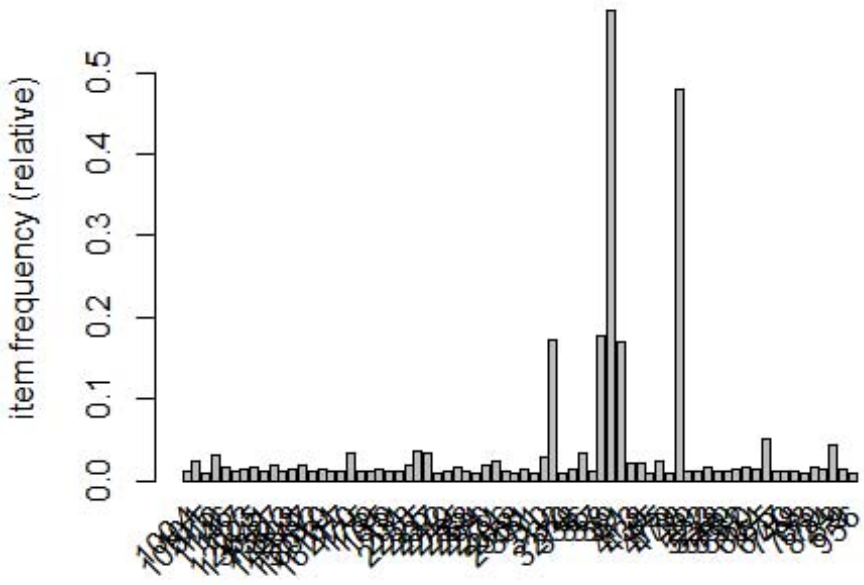
```
##     68}
## 10 {32,
##     69}
## 11 {48,
##     70,
##     71,
##     72}
## 12 {39,
##     73,
##     74,
##     75,
##     76,
##     77,
##     78,
##     79}
## 13 {36,
##     38,
##     39,
##     41,
##     48,
##     79,
##     80,
##     81}
## 14 {82,
##     83,
##     84}
## 15 {41,
##     85,
##     86,
##     87,
##     88}
## 16 {100,
##     101,
##     39,
##     48,
##     89,
##     90,
##     91,
##     92,
##     93,
##     94,
##     95,
##     96,
##     97,
##     98,
##     99}
## 17 {36,
##     38,
##     39,
##     48,
```

```
##     89}
## 18 {102,
##     103,
##     104,
##     105,
##     106,
##     107,
##     108,
##     39,
##     41}
## 19 {109,
##     110,
##     38,
##     39,
##     41}
## 20 {111,
##     112,
##     113,
##     114,
##     115,
##     116,
##     117,
##     118,
##     39}
```

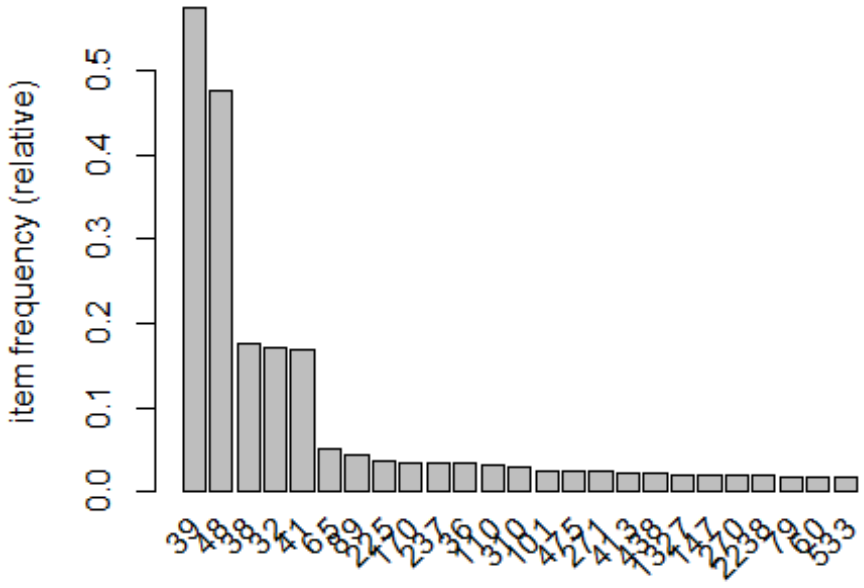
```
# Frequency of the data
itemFrequency(retail[1:100,1:20])
```

```
##     0     1    10   100  1000 10000 10001 10002 10003 10004 10005 10006
##  0.01  0.02  0.02  0.01  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
## 10007 10008 10009  1001 10010 10011 10012 10013
##  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
```

```
# Plotting the frequency
itemFrequencyPlot(retail,support=0.01)
```

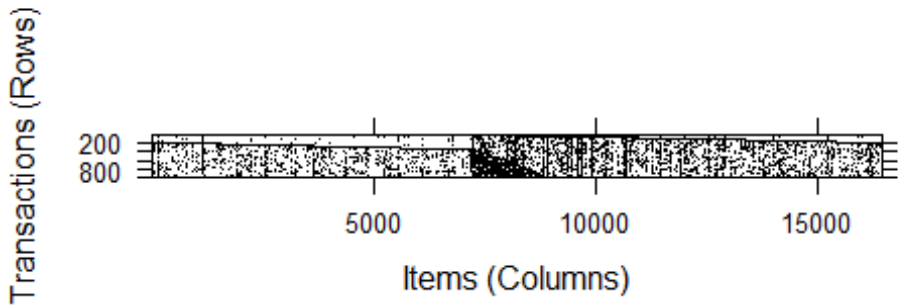



```
itemFrequencyPlot(retail,topN=25)
```

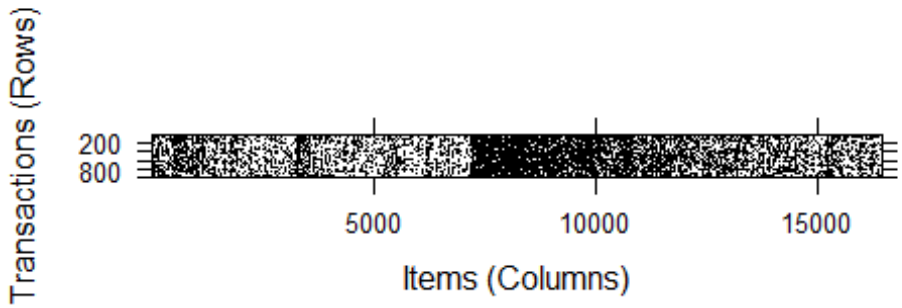



```
# First 1000 transaction visualization
image(retail[1:1000])
```

```
## Warning: closing unused connection 5
(http://fimi.ua.ac.be/data/retail.dat)
```




```
image(sample(retail,1000))
```



Generation of 50 or so non-redundant rules

```
rules1 <- apriori(retail)
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.8    0.1    1 none FALSE            TRUE     0.1      1     10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 8816
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[16470 item(s), 88162 transaction(s)] done [0.14s].
## sorting and recoding items ... [5 item(s)] done [0.01s].
## creating transaction tree ... done [0.03s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
rules1
```

```
## set of 0 rules
```

This gives set of 0 rules

```
rules2 <-
apriori(retail,parameter=list(support=0.01,confidence=0.52,minlen=2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##        0.52    0.1    1 none FALSE            TRUE    0.01      2     10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 881
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[16470 item(s), 88162 transaction(s)] done [0.17s].
## sorting and recoding items ... [70 item(s)] done [0.01s].
```

```
## creating transaction tree ... done [0.04s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [107 rule(s)] done [0.00s].
## creating S4 object  ... done [0.02s].
```

```
rules2
```

```
## set of 107 rules
```

```
summary(rules2)
```

```
## set of 107 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3  4
## 51 41 15
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.664   3.000   4.000
##
## summary of quality measures:
##     support          confidence          lift
##  Min.   :0.01013   Min.   :0.5205   Min.   :0.9698
##  1st Qu.:0.01263   1st Qu.:0.6146   1st Qu.:1.1688
##  Median :0.01588   Median :0.6848   Median :1.2558
##  Mean   :0.03038   Mean   :0.7069   Mean   :1.8113
##  3rd Qu.:0.02338   3rd Qu.:0.7642   3rd Qu.:1.3833
##  Max.   :0.33055   Max.   :0.9942   Max.   :5.6202
##
## mining info:
##    data ntransactions support confidence
##  retail         88162    0.01       0.52
```

This gives 107 rules

```
# Rules to dataframe conversion
inspect(rules2)
```

```
##     lhs            rhs  support    confidence lift
## 1   {37}        => {38} 0.01186452 0.9739292  5.5054853
## 2   {286}       => {38} 0.01265852 0.9433643  5.3327062
## 3   {12925}     => {39} 0.01063950 0.6394001  1.1123985
## 4   {1146}      => {39} 0.01114993 0.6893408  1.1992830
## 5   {79}        => {48} 0.01012908 0.5581250  1.1678039
## 6   {79}        => {39} 0.01260180 0.6943750  1.2080412
## 7   {1327}      => {48} 0.01097979 0.5419933  1.1340504
## 8   {1327}      => {39} 0.01311223 0.6472564  1.1260665
## 9   {438}       => {48} 0.01162632 0.5501879  1.1511965
## 10  {438}       => {39} 0.01429187 0.6763285  1.1766448
## 11  {60}        => {39} 0.01114993 0.6601746  1.1485410
## 12  {255}       => {48} 0.01198929 0.7170963  1.5004307
## 13  {255}       => {39} 0.01198929 0.7170963  1.2475707
```

```
## 14  {533}       => {39} 0.01045802 0.6200403  1.0787173
## 15  {270}       => {48} 0.01085502 0.5519031  1.1547854
## 16  {270}       => {39} 0.01354325 0.6885813  1.1979616
## 17  {2238}      => {48} 0.01083233 0.5568513  1.1651388
## 18  {2238}      => {39} 0.01459813 0.7504373  1.3055758
## 19  {110}       => {38} 0.03090901 0.9753042  5.5132579
## 20  {110}       => {39} 0.01995191 0.6295634  1.0952849
## 21  {147}       => {48} 0.01175109 0.5823496  1.2184908
## 22  {147}       => {39} 0.01289671 0.6391231  1.1119165
## 23  {271}       => {48} 0.01236360 0.5205349  1.0891514
## 24  {271}       => {39} 0.01626551 0.6848138  1.1914070
## 25  {413}       => {48} 0.01287403 0.6037234  1.2632126
## 26  {413}       => {39} 0.01281731 0.6010638  1.0457028
## 27  {36}        => {38} 0.03164629 0.9502725  5.3717570
## 28  {36}        => {39} 0.02310519 0.6938011  1.2070428
## 29  {475}       => {48} 0.01619745 0.6589755  1.3788205
## 30  {475}       => {39} 0.01701413 0.6922012  1.2042593
## 31  {170}       => {38} 0.03437989 0.9780574  5.5288215
## 32  {170}       => {39} 0.02335473 0.6644079  1.1559058
## 33  {101}       => {48} 0.01487035 0.5860527  1.2262391
## 34  {101}       => {39} 0.01587986 0.6258382  1.0888041
## 35  {310}       => {48} 0.01919194 0.6522745  1.3647994
## 36  {310}       => {39} 0.02100678 0.7139553  1.2421061
## 37  {237}       => {48} 0.01907851 0.5547493  1.1607407
## 38  {237}       => {39} 0.02188018 0.6362137  1.1068549
## 39  {225}       => {48} 0.01969102 0.5330058  1.1152453
## 40  {225}       => {39} 0.02666682 0.7218299  1.2558060
## 41  {89}        => {48} 0.03173703 0.7292155  1.5257885
## 42  {89}        => {39} 0.03118123 0.7164451  1.2464378
## 43  {65}        => {48} 0.02868583 0.5655188  1.1832744
## 44  {65}        => {39} 0.03161226 0.6232111  1.0842336
## 45  {38}        => {39} 0.11734080 0.6633111  1.1539977
## 46  {32}        => {48} 0.09112770 0.5297026  1.1083338
## 47  {32}        => {39} 0.09590300 0.5574603  0.9698434
## 48  {41}        => {48} 0.10228897 0.6034125  1.2625621
## 49  {41}        => {39} 0.12946621 0.7637337  1.3287082
## 50  {48}        => {39} 0.33055058 0.6916340  1.2032726
## 51  {39}        => {48} 0.33055058 0.5750765  1.2032726
## 52  {110,48}    => {38} 0.01543749 0.9862319  5.5750305
## 53  {110,38}    => {39} 0.01973639 0.6385321  1.1108884
## 54  {110,39}    => {38} 0.01973639 0.9891984  5.5917998
## 55  {110,48}    => {39} 0.01176244 0.7514493  1.3073364
## 56  {110,39}    => {48} 0.01176244 0.5895395  1.2335346
## 57  {36,48}     => {38} 0.01542615 0.9604520  5.4293003
## 58  {36,38}     => {39} 0.02206166 0.6971326  1.2128388
## 59  {36,39}     => {38} 0.02206166 0.9548355  5.3975514
## 60  {36,48}     => {39} 0.01265852 0.7881356  1.3711615
## 61  {36,39}     => {48} 0.01265852 0.5478645  1.1463351
## 62  {475,48}    => {39} 0.01238629 0.7647059  1.3303996
## 63  {39,475}    => {48} 0.01238629 0.7280000  1.5232452
```

```
## 64  {170,48}    => {38} 0.01744516 0.9877970  5.5838781
## 65  {170,38}    => {39} 0.02290102 0.6661168  1.1588789
## 66  {170,39}    => {38} 0.02290102 0.9805731  5.5430421
## 67  {170,48}    => {39} 0.01367936 0.7745665  1.3475546
## 68  {170,39}    => {48} 0.01367936 0.5857212  1.2255454
## 69  {101,48}    => {39} 0.01073025 0.7215866  1.2553826
## 70  {101,39}    => {48} 0.01073025 0.6757143  1.4138441
## 71  {310,48}    => {39} 0.01527869 0.7960993  1.3850164
## 72  {310,39}    => {48} 0.01527869 0.7273218  1.5218262
## 73  {237,48}    => {39} 0.01411039 0.7395957  1.2867141
## 74  {237,39}    => {48} 0.01411039 0.6448937  1.3493561
## 75  {225,48}    => {39} 0.01587986 0.8064516  1.4030269
## 76  {225,39}    => {48} 0.01587986 0.5954913  1.2459879
## 77  {48,89}     => {39} 0.02410336 0.7594711  1.3212923
## 78  {39,89}     => {48} 0.02410336 0.7730084  1.6174193
## 79  {48,65}     => {39} 0.02038293 0.7105575  1.2361948
## 80  {39,65}     => {48} 0.02038293 0.6447793  1.3491168
## 81  {32,38}     => {48} 0.01867018 0.5810095  1.2156868
## 82  {32,38}     => {39} 0.02087067 0.6494882  1.1299492
## 83  {38,41}     => {48} 0.02692770 0.6091866  1.2746435
## 84  {38,41}     => {39} 0.03460675 0.7829099  1.3620702
## 85  {38,48}     => {39} 0.06921349 0.7681269  1.3363513
## 86  {38,39}     => {48} 0.06921349 0.5898502  1.2341847
## 87  {32,41}     => {48} 0.02340010 0.6454944  1.3506129
## 88  {32,41}     => {39} 0.02675756 0.7381101  1.2841296
## 89  {32,48}     => {39} 0.06127356 0.6723923  1.1697968
## 90  {32,39}     => {48} 0.06127356 0.6389119  1.3368399
## 91  {41,48}     => {39} 0.08355074 0.8168108  1.4210493
## 92  {39,41}     => {48} 0.08355074 0.6453478  1.3503063
## 93  {110,38,48} => {39} 0.01169438 0.7575312  1.3179175
## 94  {110,38,39} => {48} 0.01169438 0.5925287  1.2397892
## 95  {110,39,48} => {38} 0.01169438 0.9942141  5.6201527
## 96  {36,38,48}  => {39} 0.01225018 0.7941176  1.3815688
## 97  {36,38,39}  => {48} 0.01225018 0.5552699  1.1618300
## 98  {36,39,48}  => {38} 0.01225018 0.9677419  5.4705094
## 99  {170,38,48} => {39} 0.01353191 0.7756827  1.3494966
## 100 {170,38,39} => {48} 0.01353191 0.5908866  1.2363532
## 101 {170,39,48} => {38} 0.01353191 0.9892206  5.5919251
## 102 {32,38,48}  => {39} 0.01401965 0.7509113  1.3064004
## 103 {32,38,39}  => {48} 0.01401965 0.6717391  1.4055266
## 104 {38,41,48}  => {39} 0.02258343 0.8386689  1.4590770
## 105 {38,39,41}  => {48} 0.02258343 0.6525729  1.3654239
## 106 {32,41,48}  => {39} 0.01867018 0.7978672  1.3880921
## 107 {32,39,41}  => {48} 0.01867018 0.6977533  1.4599579
```

```
rulesDF <- as(rules2,"data.frame")
str(rulesDF)
```

```
## 'data.frame':    107 obs. of  4 variables:
##  $ rules     : Factor w/ 107 levels "{101,39} => {48}",..: 74 47 16 15 105
```

```
104 18 17 93 92 ...
##  $ support   : num  0.0119 0.0127 0.0106 0.0111 0.0101 ...
##  $ confidence: num  0.974 0.943 0.639 0.689 0.558 ...
##  $ lift      : num  5.51 5.33 1.11 1.2 1.17 ...
```

```
# Removing redundant rules
subset.matrix <- is.subset(rules2,rules2)
subset.matrix[lower.tri(subset.matrix,diag = T)] <- NA
redundant <- colSums(subset.matrix,na.rm = T) >= 1
which(redundant)
```

```
##        {39,48}    {110,38,48}    {110,38,39}    {110,38,39}    {110,39,48}
##             51             52             53             54             55
##    {110,39,48}     {36,38,48}     {36,38,39}     {36,38,39}     {36,39,48}
##             56             57             58             59             60
##     {36,39,48}    {39,475,48}    {39,475,48}    {170,38,48}    {170,38,39}
##             61             62             63             64             65
##    {170,38,39}    {170,39,48}    {170,39,48}    {101,39,48}    {101,39,48}
##             66             67             68             69             70
##    {310,39,48}    {310,39,48}    {237,39,48}    {237,39,48}    {225,39,48}
##             71             72             73             74             75
##    {225,39,48}     {39,48,89}     {39,48,89}     {39,48,65}     {39,48,65}
##             76             77             78             79             80
##     {32,38,48}     {32,38,39}     {38,41,48}     {38,39,41}     {38,39,48}
##             81             82             83             84             85
##     {38,39,48}     {32,41,48}     {32,39,41}     {32,39,48}     {32,39,48}
##             86             87             88             89             90
##     {39,41,48}     {39,41,48} {110,38,39,48} {110,38,39,48} {110,38,39,48}
##             91             92             93             94             95
##  {36,38,39,48}  {36,38,39,48}  {36,38,39,48} {170,38,39,48} {170,38,39,48}
##             96             97             98             99            100
## {170,38,39,48}  {32,38,39,48}  {32,38,39,48}  {38,39,41,48}  {38,39,41,48}
##            101            102            103            104            105
##  {32,39,41,48}  {32,39,41,48}
##            106            107
```

```
rulesPruned <- rules2[!redundant]
rulesPruned
```

```
## set of 50 rules
```

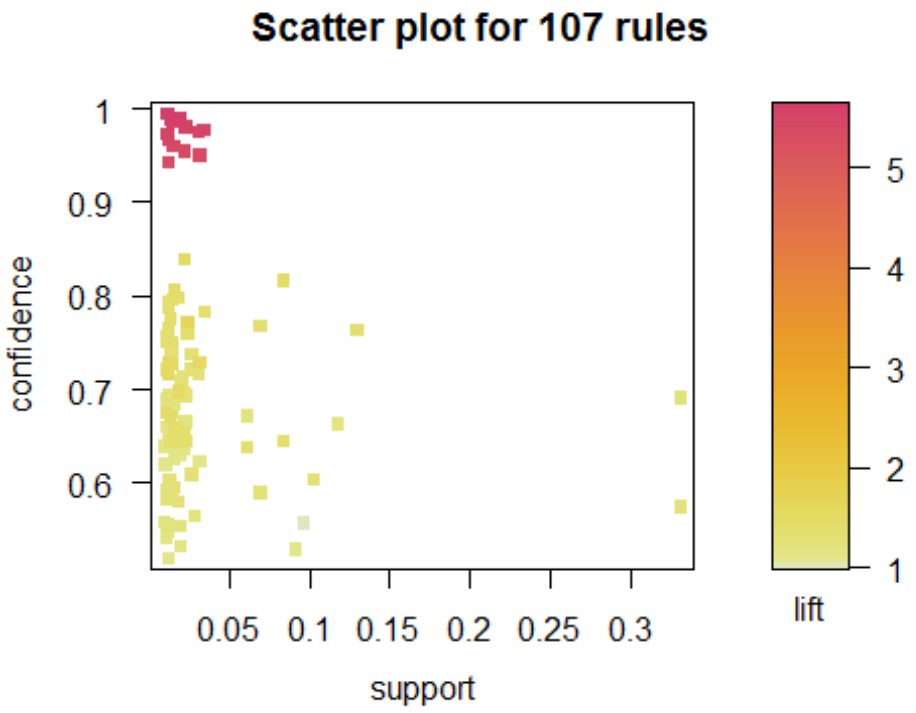
```
inspect(rulesPruned)
```

```
##    lhs         rhs  support    confidence lift
## 1  {37}    => {38} 0.01186452 0.9739292  5.5054853
## 2  {286}   => {38} 0.01265852 0.9433643  5.3327062
## 3  {12925} => {39} 0.01063950 0.6394001  1.1123985
## 4  {1146}  => {39} 0.01114993 0.6893408  1.1992830
## 5  {79}    => {48} 0.01012908 0.5581250  1.1678039
## 6  {79}    => {39} 0.01260180 0.6943750  1.2080412
## 7  {1327}  => {48} 0.01097979 0.5419933  1.1340504
## 8  {1327}  => {39} 0.01311223 0.6472564  1.1260665
```

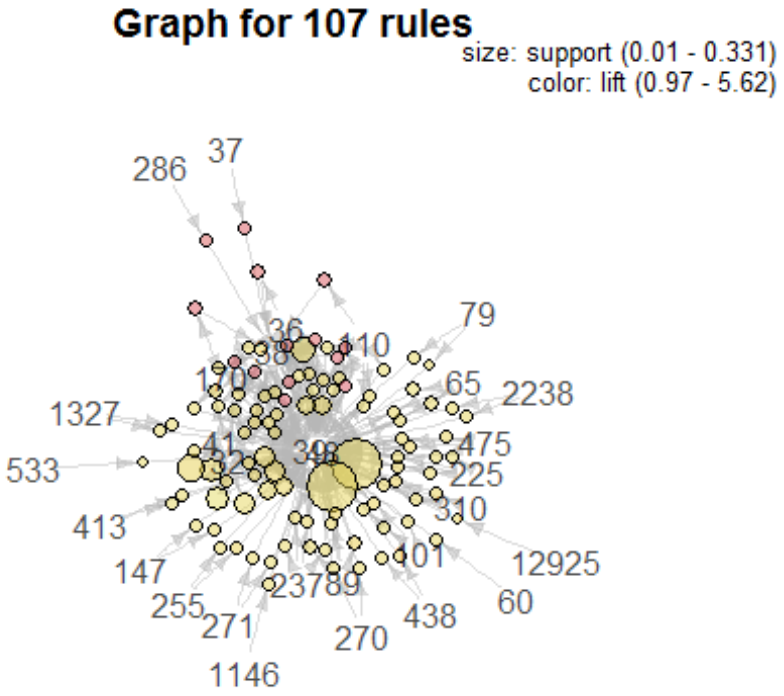
```
## 9  {438}   => {48} 0.01162632 0.5501879  1.1511965
## 10 {438}   => {39} 0.01429187 0.6763285  1.1766448
## 11 {60}    => {39} 0.01114993 0.6601746  1.1485410
## 12 {255}   => {48} 0.01198929 0.7170963  1.5004307
## 13 {255}   => {39} 0.01198929 0.7170963  1.2475707
## 14 {533}   => {39} 0.01045802 0.6200403  1.0787173
## 15 {270}   => {48} 0.01085502 0.5519031  1.1547854
## 16 {270}   => {39} 0.01354325 0.6885813  1.1979616
## 17 {2238}  => {48} 0.01083233 0.5568513  1.1651388
## 18 {2238}  => {39} 0.01459813 0.7504373  1.3055758
## 19 {110}   => {38} 0.03090901 0.9753042  5.5132579
## 20 {110}   => {39} 0.01995191 0.6295634  1.0952849
## 21 {147}   => {48} 0.01175109 0.5823496  1.2184908
## 22 {147}   => {39} 0.01289671 0.6391231  1.1119165
## 23 {271}   => {48} 0.01236360 0.5205349  1.0891514
## 24 {271}   => {39} 0.01626551 0.6848138  1.1914070
## 25 {413}   => {48} 0.01287403 0.6037234  1.2632126
## 26 {413}   => {39} 0.01281731 0.6010638  1.0457028
## 27 {36}    => {38} 0.03164629 0.9502725  5.3717570
## 28 {36}    => {39} 0.02310519 0.6938011  1.2070428
## 29 {475}   => {48} 0.01619745 0.6589755  1.3788205
## 30 {475}   => {39} 0.01701413 0.6922012  1.2042593
## 31 {170}   => {38} 0.03437989 0.9780574  5.5288215
## 32 {170}   => {39} 0.02335473 0.6644079  1.1559058
## 33 {101}   => {48} 0.01487035 0.5860527  1.2262391
## 34 {101}   => {39} 0.01587986 0.6258382  1.0888041
## 35 {310}   => {48} 0.01919194 0.6522745  1.3647994
## 36 {310}   => {39} 0.02100678 0.7139553  1.2421061
## 37 {237}   => {48} 0.01907851 0.5547493  1.1607407
## 38 {237}   => {39} 0.02188018 0.6362137  1.1068549
## 39 {225}   => {48} 0.01969102 0.5330058  1.1152453
## 40 {225}   => {39} 0.02666682 0.7218299  1.2558060
## 41 {89}    => {48} 0.03173703 0.7292155  1.5257885
## 42 {89}    => {39} 0.03118123 0.7164451  1.2464378
## 43 {65}    => {48} 0.02868583 0.5655188  1.1832744
## 44 {65}    => {39} 0.03161226 0.6232111  1.0842336
## 45 {38}    => {39} 0.11734080 0.6633111  1.1539977
## 46 {32}    => {48} 0.09112770 0.5297026  1.1083338
## 47 {32}    => {39} 0.09590300 0.5574603  0.9698434
## 48 {41}    => {48} 0.10228897 0.6034125  1.2625621
## 49 {41}    => {39} 0.12946621 0.7637337  1.3287082
## 50 {48}    => {39} 0.33055058 0.6916340  1.2032726
```

Exactly 50 rules are obtained

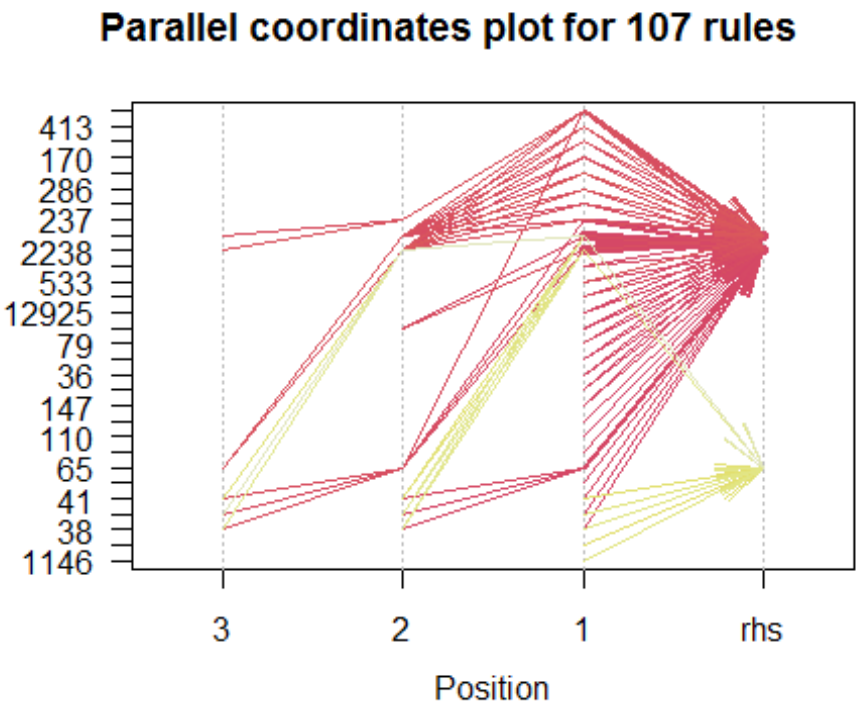
```
# Visualize association rules
plot(rules2)
```




```
plot(rules2,method = "graph", control=list(type="items"))
```




```
plot(rules2, method="paracoord", control=list(reorder=TRUE))
```




```
# Conviction
conviction <- interestMeasure(rulesPruned, "conviction", transactions=retail)
rulesConv<-as(rulesPruned, "data.frame")
rulesConv<-data.frame(rulesConv, conviction)
rulesConv
```

```
##               rules    support confidence      lift conviction
## 1     {37} => {38} 0.01186452  0.9739292 5.5054853  31.571702
## 2    {286} => {38} 0.01265852  0.9433643 5.3327062  14.533215
## 3  {12925} => {39} 0.01063950  0.6394001 1.1123985   1.179163
## 4   {1146} => {39} 0.01114993  0.6893408 1.1992830   1.368721
## 5     {79} => {48} 0.01012908  0.5581250 1.1678039   1.181495
## 6     {79} => {39} 0.01260180  0.6943750 1.2080412   1.391267
## 7   {1327} => {48} 0.01097979  0.5419933 1.1340504   1.139881
## 8   {1327} => {39} 0.01311223  0.6472564 1.1260665   1.205425
## 9    {438} => {48} 0.01162632  0.5501879 1.1511965   1.160647
## 10   {438} => {39} 0.01429187  0.6763285 1.1766448   1.313696
## 11    {60} => {39} 0.01114993  0.6601746 1.1485410   1.251248
## 12   {255} => {48} 0.01198929  0.7170963 1.5004307   1.845409
## 13   {255} => {39} 0.01198929  0.7170963 1.2475707   1.503006
## 14   {533} => {39} 0.01045802  0.6200403 1.0787173   1.119082
## 15   {270} => {48} 0.01085502  0.5519031 1.1547854   1.165090
## 16   {270} => {39} 0.01354325  0.6885813 1.1979616   1.365383
## 17  {2238} => {48} 0.01083233  0.5568513 1.1651388   1.178099
## 18  {2238} => {39} 0.01459813  0.7504373 1.3055758   1.703804
## 19   {110} => {38} 0.03090901  0.9753042 5.5132579  33.329520
## 20   {110} => {39} 0.01995191  0.6295634 1.0952849   1.147850
```

```
## 21   {147} => {48} 0.01175109  0.5823496 1.2184908   1.250024
## 22   {147} => {39} 0.01289671  0.6391231 1.1119165   1.178257
## 23   {271} => {48} 0.01236360  0.5205349 1.0891514   1.088865
## 24   {271} => {39} 0.01626551  0.6848138 1.1914070   1.349062
## 25   {413} => {48} 0.01287403  0.6037234 1.2632126   1.317446
## 26   {413} => {39} 0.01281731  0.6010638 1.0457028   1.065849
## 27    {36} => {38} 0.03164629  0.9502725 5.3717570  16.552170
## 28    {36} => {39} 0.02310519  0.6938011 1.2070428   1.388659
## 29   {475} => {48} 0.01619745  0.6589755 1.3788205   1.530896
## 30   {475} => {39} 0.01701413  0.6922012 1.2042593   1.381441
## 31   {170} => {38} 0.03437989  0.9780574 5.5288215  37.511499
## 32   {170} => {39} 0.02335473  0.6644079 1.1559058   1.267032
## 33   {101} => {48} 0.01487035  0.5860527 1.2262391   1.261207
## 34   {101} => {39} 0.01587986  0.6258382 1.0888041   1.136422
## 35   {310} => {48} 0.01919194  0.6522745 1.3647994   1.501394
## 36   {310} => {39} 0.02100678  0.7139553 1.2421061   1.486501
## 37   {237} => {48} 0.01907851  0.5547493 1.1607407   1.172537
## 38   {237} => {39} 0.02188018  0.6362137 1.1068549   1.168834
## 39   {225} => {48} 0.01969102  0.5330058 1.1152453   1.117943
## 40   {225} => {39} 0.02666682  0.7218299 1.2558060   1.528582
## 41    {89} => {48} 0.03173703  0.7292155 1.5257885   1.928002
## 42    {89} => {39} 0.03118123  0.7164451 1.2464378   1.499554
## 43    {65} => {48} 0.02868583  0.5655188 1.1832744   1.201601
## 44    {65} => {39} 0.03161226  0.6232111 1.0842336   1.128499
## 45    {38} => {39} 0.11734080  0.6633111 1.1539977   1.262904
## 46    {32} => {48} 0.09112770  0.5297026 1.1083338   1.110091
## 47    {32} => {39} 0.09590300  0.5574603 0.9698434   0.960831
## 48    {41} => {48} 0.10228897  0.6034125 1.2625621   1.316413
## 49    {41} => {39} 0.12946621  0.7637337 1.3287082   1.799689
## 50    {48} => {39} 0.33055058  0.6916340 1.2032726   1.378900
```

Answers:

A(1)

The rules that have high level of confidence and high support makes proper sense to me.
The top 5 Best rules based on high confidence and high support are:
1. {48} => {39}
2. {41} => {39}
3. {38} => {39}
4. {41} => {48}
5. {32} => {39}

The top 5 Worst rules based on low confidence and low support are:
1. {533} => {39}
2. {79} => {48}
3. {12925} => {39}
4. {2238} => {48}
5. {270} => {48}

A(2)

Initially I used the default level of confidence and support but it yielded 0 results of rules. Thus I tried with a various permuation-conbination of 'support', 'confidence' and 'minlen'. Finally I was able to obtain about 107 rules with the following values:
support = 0.01 confidence = 0.52 minlen = 2
The reason I obtained 107 rules is that most of the rules will be pruned as they might be redundant and hence I would have 50 rules in the end.

A(3)

Lift and conviction for top 5 Best rules:
No.....Rules.....Lift.....Conviction

1. {48} => {39} | 1.2032726 | 1.378900
2. {41} => {39} | 1.3287082 | 1.799689
3. {38} => {39} | 1.1539977 | 1.262904
4. {41} => {48} | 1.2625621 | 1.316413
5. {32} => {39} | 0.9698434 | 0.960831

Lift and conviction for top 5 Worst rules:
No.....Rules.....Lift.....Conviction

1. {533} => {39} | 1.0787173 | 1.119082
2. {79} => {48} | 1.1678039 | 1.181495
3. {12925} => {39}| 1.1123985 | 1.179163
4. {2238} => {48} | 1.1651388 | 1.178099
5. {270} => {48} | 1.1547854 | 1.165090

A(4)

The best rules end up being in the centre of the visualization graph. Thus this proves that they are the best ones. The worst rules are on the outer end of the graph. Hence they do not contribute to the dataframe. It is clearly seen that the rules '48', '39', '41', '32' etc are quite close by. The rules '270', '2238', '12925' are far away from the centre and hence are the worst rules.

A(5)

Yes absolutely. The model makes good sense as the rules guide which path is to be taken in order to maximize the outputs. In terms of supermarket, if customers buy X and Y together as well as Y and Z togehter, then the rules guide them in such a way that whomsoever buys X or Y would end up buying Y at the least. This is where the rules help in maximizing the transactions and understanding the relations between each data item with respect to the other.