Omkar Gurav

BE IT

8048

Assignment 6

Answer the following questions

i) What is Association Rule Mining?

→

- It is rule based unsupervised machine learning task for discovering interesting relations between variables in large databases or transactions.

- In given set of transactions, it finds rules that will predict the occurence of an item based on the occurences of other items in the transaction.

- Examples of the Association Rules:
  {bread} → {milk}, {Soda} → {Chips} for transactions
  1. bread, milk, soda
  2. soda, milk
  3. Soda, chips, milk
  4. Soda, bread, chips

- Association rules mining is a technique to discover how items are associated to each other. Threre are 2 common ways to measure association.

ii) Define following terms

i). Rule

→ Rule is given in the form of X→Y where X & Y are 2 itemsets.

ii. Item set

→ A collection of 1 or more items in any transaction is known as itemset.
eg. {milk, bread, jam}

iii. Support

→ ~~Frequency of~~ The Fraction of transactions that contain an itemset is known as support.

iv. Frequent Item set

→ An itemset whose support is greater than or equal to a minimum threshold are known as Frequent itemset.

v. Confidence

→ It indicates the number of times the if/then statements have been found to be true. It measures how often items in Y appear in transactions that contain X.

$$\text{Conf} = \frac{\text{Freq}(X \cup Y)}{\text{Freq}(X)}$$

vi. Lift

→ Lift is nothing but ratio of confidence to Expected confidence.

iii) Explain Apriori Algorithm.

→

- In learning association rules, Apriori is a classic machine learning algorithm.

- Apriori is designed to work on databases covering transactions.

- The algorithm is aimed to find subsets which are common to atleast a minimum number C (confidence threshold) of the itemsets.

- It follows "bottom up" approach, where frequent subsets are extended one item at a time & groups of candidates are tested against the data.

- The algorithm is continues till no further successful extensions have been found.

- Apriori uses breadth-first search & a hash tree structure to count candidate item sets efficiently.

- Apriori Property:-

  Any subset of a frequent itemset must be frequent. if {AB} is a frequent itemset, both {A} & {B} should be a frequent itemset. Iteratively find frequent itemsets with cardinality from 1 to k (k-itemset).

- Pseudocode for Apriori Algorithm:-

Join step - It is generated by joining with itself.
Prune step - Any (k-1) item set that is not frequent cannot be a subset of a frequent k-item set.

Pseudo-code -

C_k : Candidate item set of size k
L_k : Frequent item set of size k

```
        $L_1$ = {Frequent items}
For ($k=1$; $L_k| = \phi$ ; $k$++) do begin
        $C_{k+1}$ = Candidates generated from $L_k$;
    For each transaction t in database do
    Increment the count of all candidates in $C_{k+1}$
    Those are contained in t
        $L_{k+1}$ = Candidates in $C_{k+1}$ with min_support
    End
    Return $U_k L_k$;
```

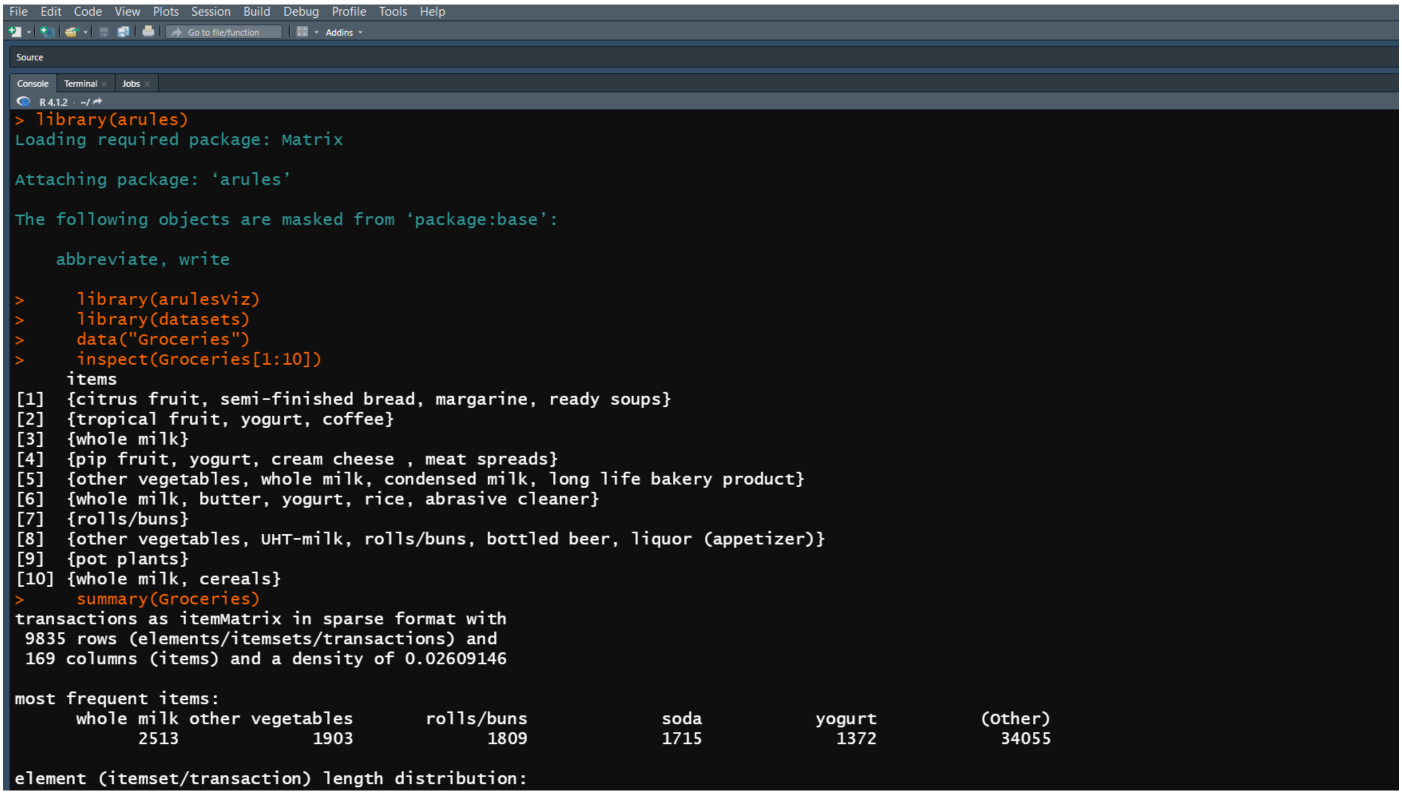
Association_Rules.R

```r
library(arules)
library(arulesViz)
library(datasets)

data("Groceries")

inspect(Groceries[1:10])

summary(Groceries)

par(mfrow = c(1,1))
# Frequency plot of top 10 items
itemFrequencyPlot(Groceries, type = 'absolute', topN = 10)
#itemFrequencyPlot(Groceries, type = 'relative', topN = 10)


 # 1.

 # Getting rules
 rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.3))

 summary(rules)

 inspect(rules[1:10])

 # Checking if there are any reduntant rules
 rules[is.redundant(rules)]
```

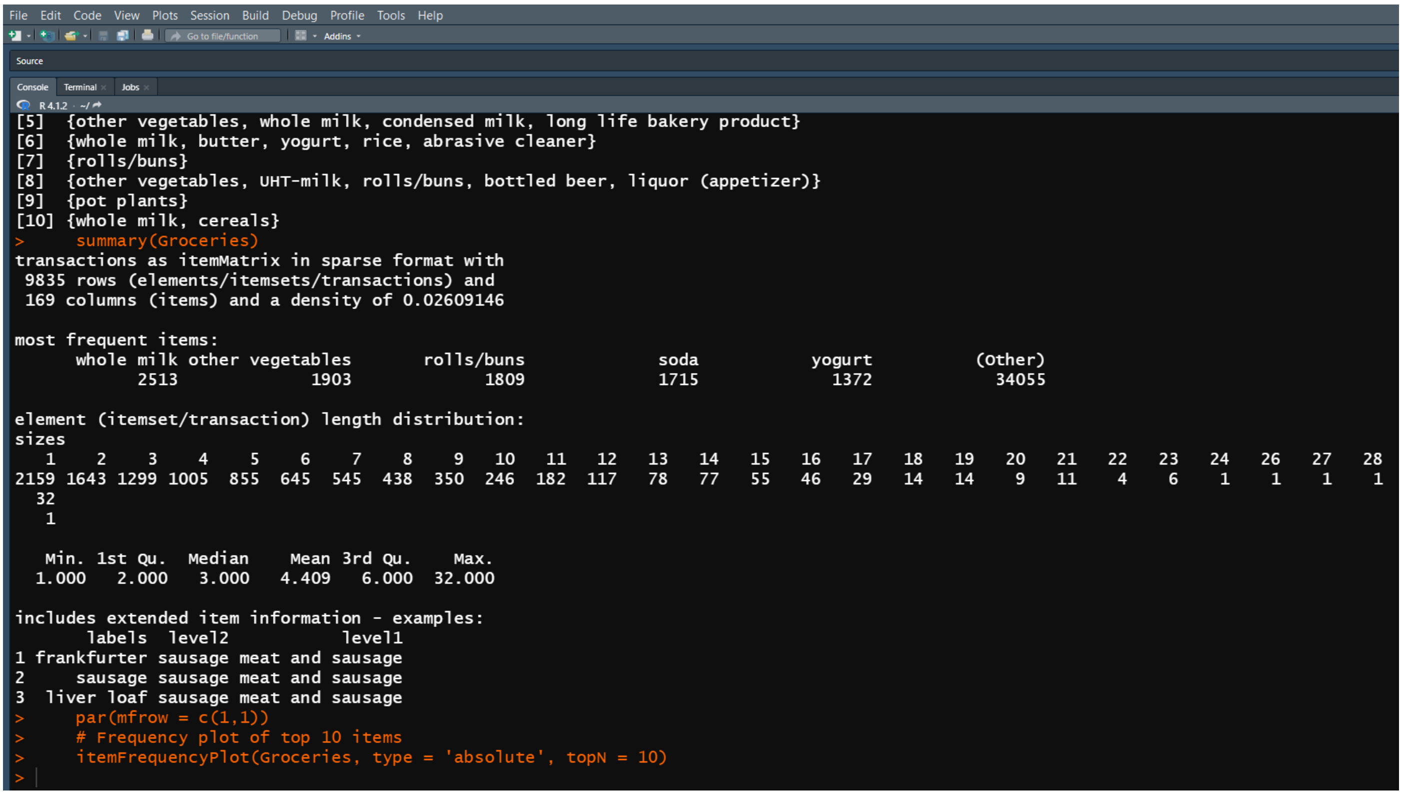
```
inspect(rules[is.redundant(rules)])

# Removing reduntant rules
rules <- rules[!is.redundant(rules)]

# Sorting rules by confidence
rules <- sort(rules, by = 'confidence')
inspect(rules[1:10])

plot(rules, method = 'graph',measure = "confidence", shading = "support",
    engine = "htmlwidget", control = list(max = 50))

plot(rules, method = 'paracoord')



# 2.

# Getting rules
rules_1 <- apriori(Groceries, parameter = list(supp = 0.03, conf = 0.3))

summary(rules_1)

inspect(rules_1)

# Checking if there are any reduntant rules
rules_1[is.redundant(rules_1)]

# Sorting rules by confidence
rules_1 <- sort(rules_1, by = 'confidence')
inspect(rules_1)
```

```
plot(rules_1, method = 'graph',measure = "support", shading = "confidence",
     engine = "htmlwidget")
plot(rules_1, method = 'paracoord')



# 3.

# Getting rules
rules_2 <- apriori(Groceries, parameter = list(supp = 0.04, conf = 0.4))

summary(rules_2)

inspect(rules_2)

# Checking if there are any reduntant rules
rules_2[is.redundant(rules_2)]

# Sorting rules by lift
rules_2 <- sort(rules_2, by = 'lift')
inspect(rules_2)

plot(rules_2, method = 'graph',measure = "lift", shading = "support",
     engine = "htmlwidget")

plot(rules_2, method = 'paracoord')
```

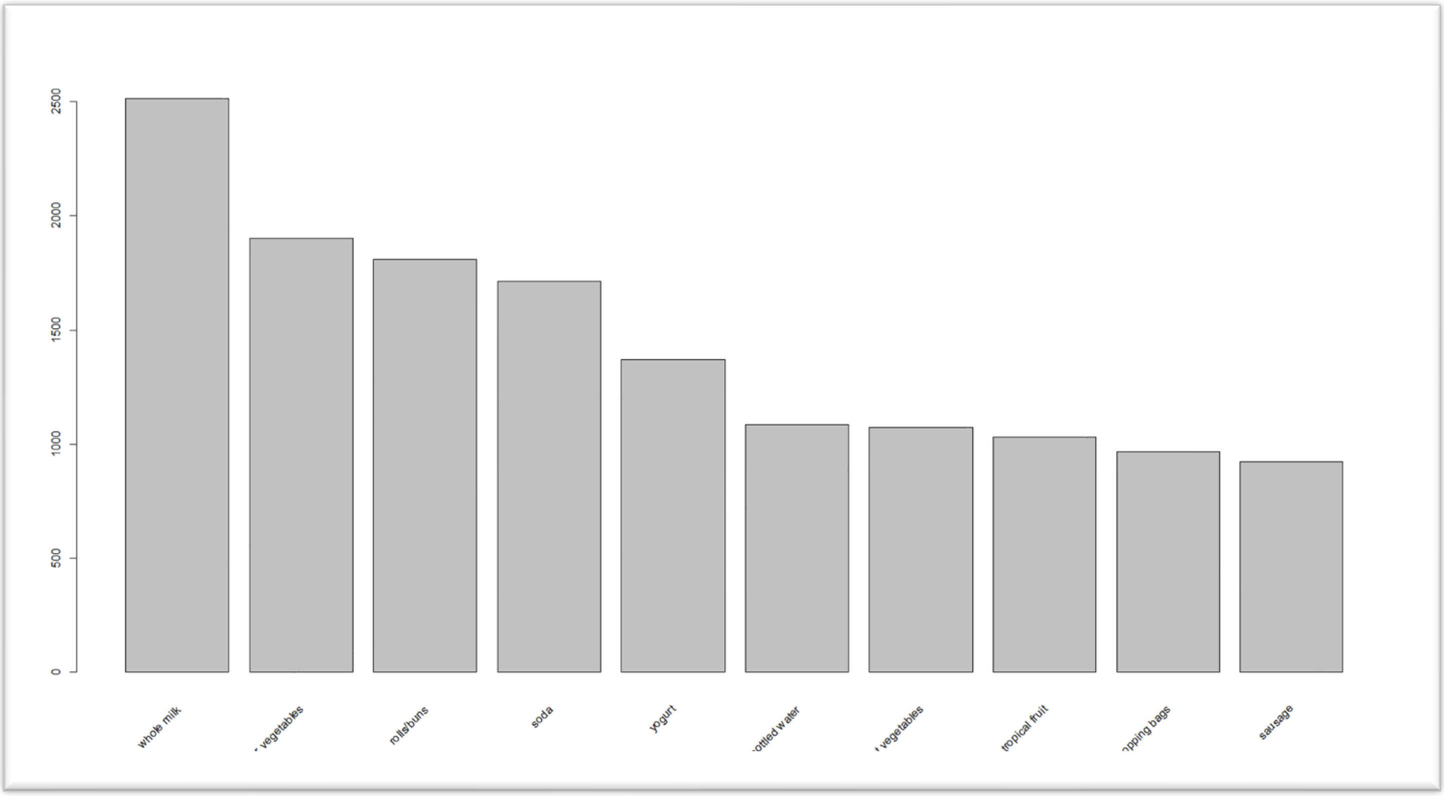
Output :

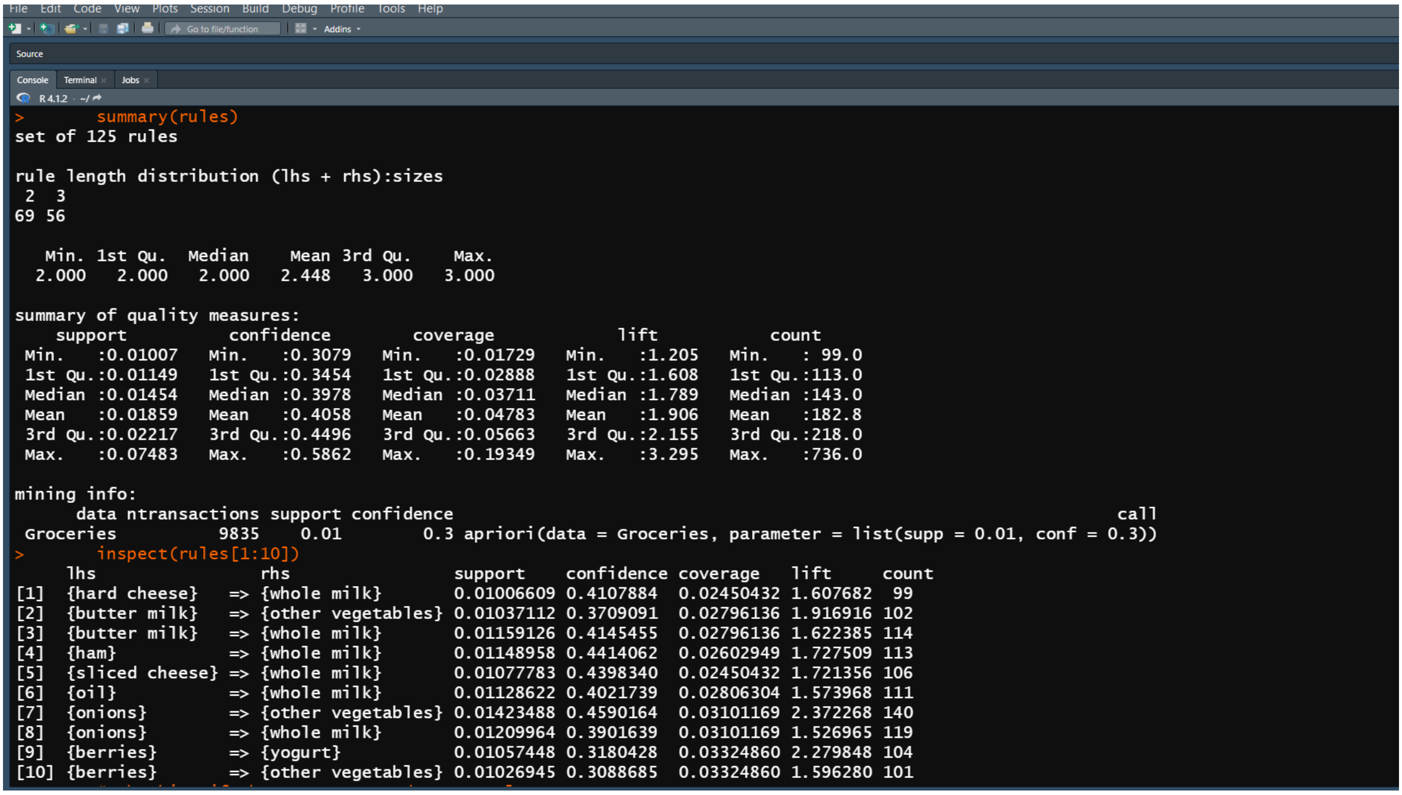

File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Jobs
R 4.1.2 · ~/

```
> library(arules)
Loading required package: Matrix

Attaching package: ‘arules’

The following objects are masked from ‘package:base’:

    abbreviate, write

>     library(arulesViz)
>     library(datasets)
>     data("Groceries")
>     inspect(Groceries[1:10])
     items
[1]  {citrus fruit, semi-finished bread, margarine, ready soups}
[2]  {tropical fruit, yogurt, coffee}
[3]  {whole milk}
[4]  {pip fruit, yogurt, cream cheese , meat spreads}
[5]  {other vegetables, whole milk, condensed milk, long life bakery product}
[6]  {whole milk, butter, yogurt, rice, abrasive cleaner}
[7]  {rolls/buns}
[8]  {other vegetables, UHT-milk, rolls/buns, bottled beer, liquor (appetizer)}
[9]  {pot plants}
[10] {whole milk, cereals}
>     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           yogurt          (Other)
            2513             1903             1809             1715             1372            34055

element (itemset/transaction) length distribution:
```

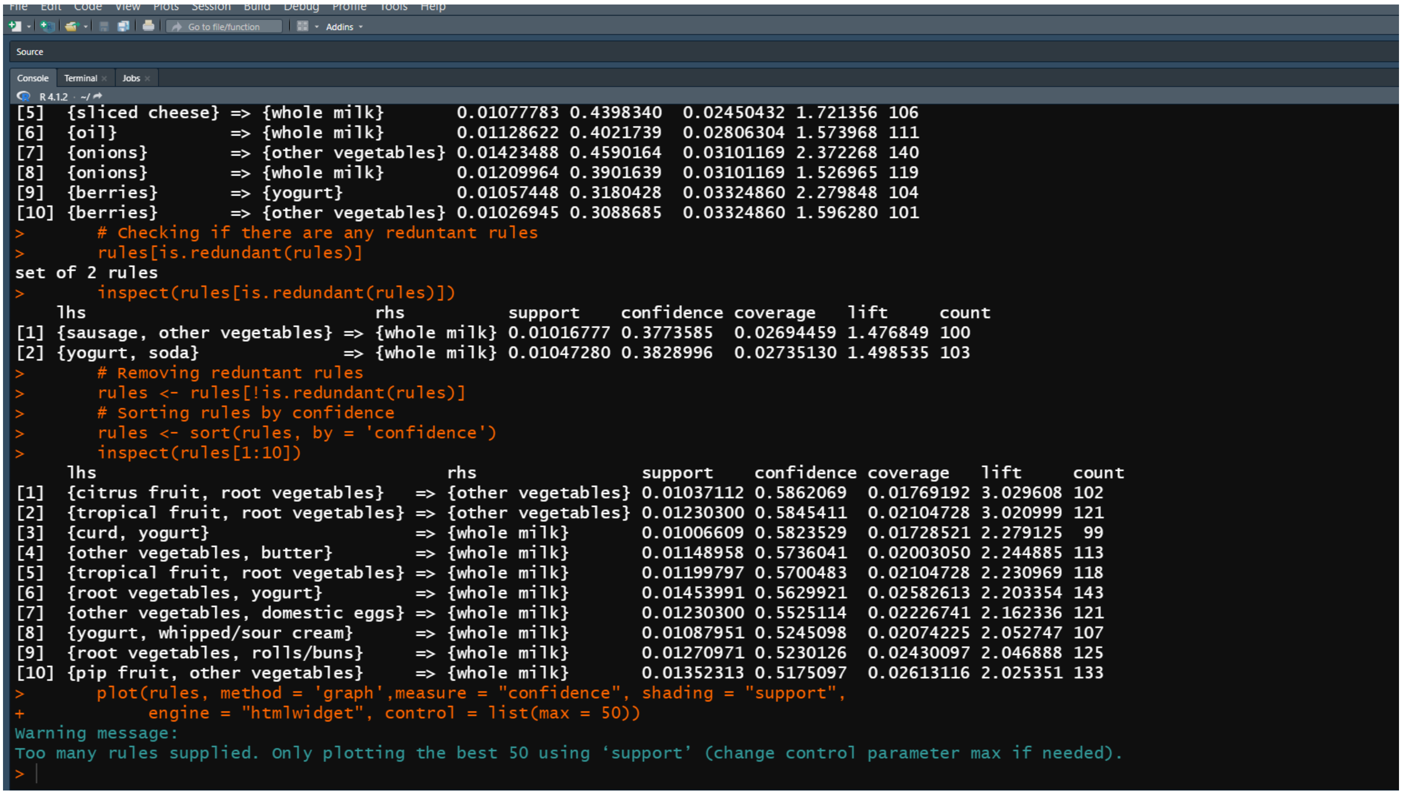

File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Jobs
R 4.1.2 · ~/

```
[5]  {other vegetables, whole milk, condensed milk, long life bakery product}
[6]  {whole milk, butter, yogurt, rice, abrasive cleaner}
[7]  {rolls/buns}
[8]  {other vegetables, UHT-milk, rolls/buns, bottled beer, liquor (appetizer)}
[9]  {pot plants}
[10] {whole milk, cereals}
>     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           yogurt          (Other)
            2513             1903             1809             1715             1372            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   26   27   28
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46   29   14   14    9   11    4    6    1    1    1    1
  32
   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
>     par(mfrow = c(1,1))
>     # Frequency plot of top 10 items
>     itemFrequencyPlot(Groceries, type = 'absolute', topN = 10)
> 
```

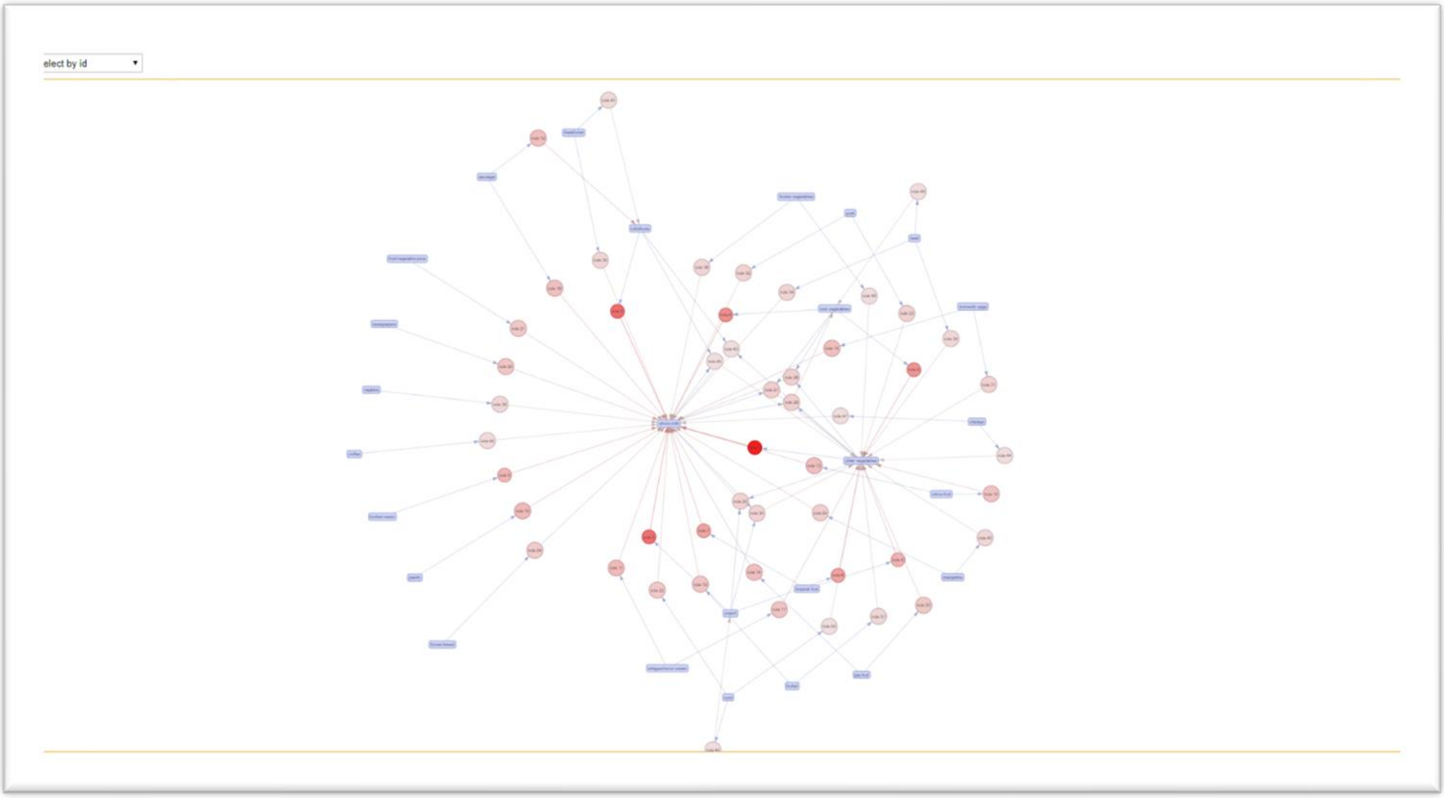


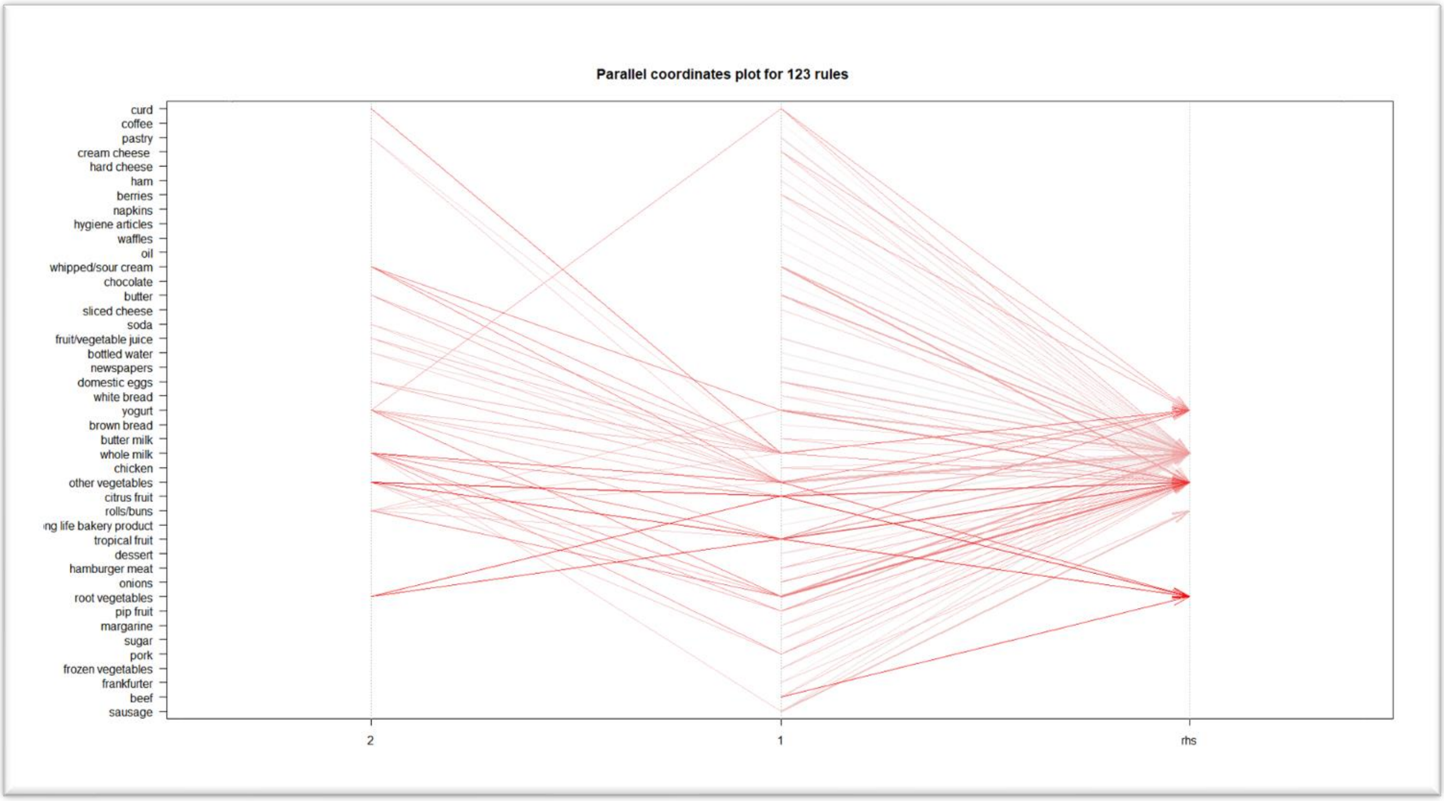

File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Jobs
R 4.1.2 · ~/

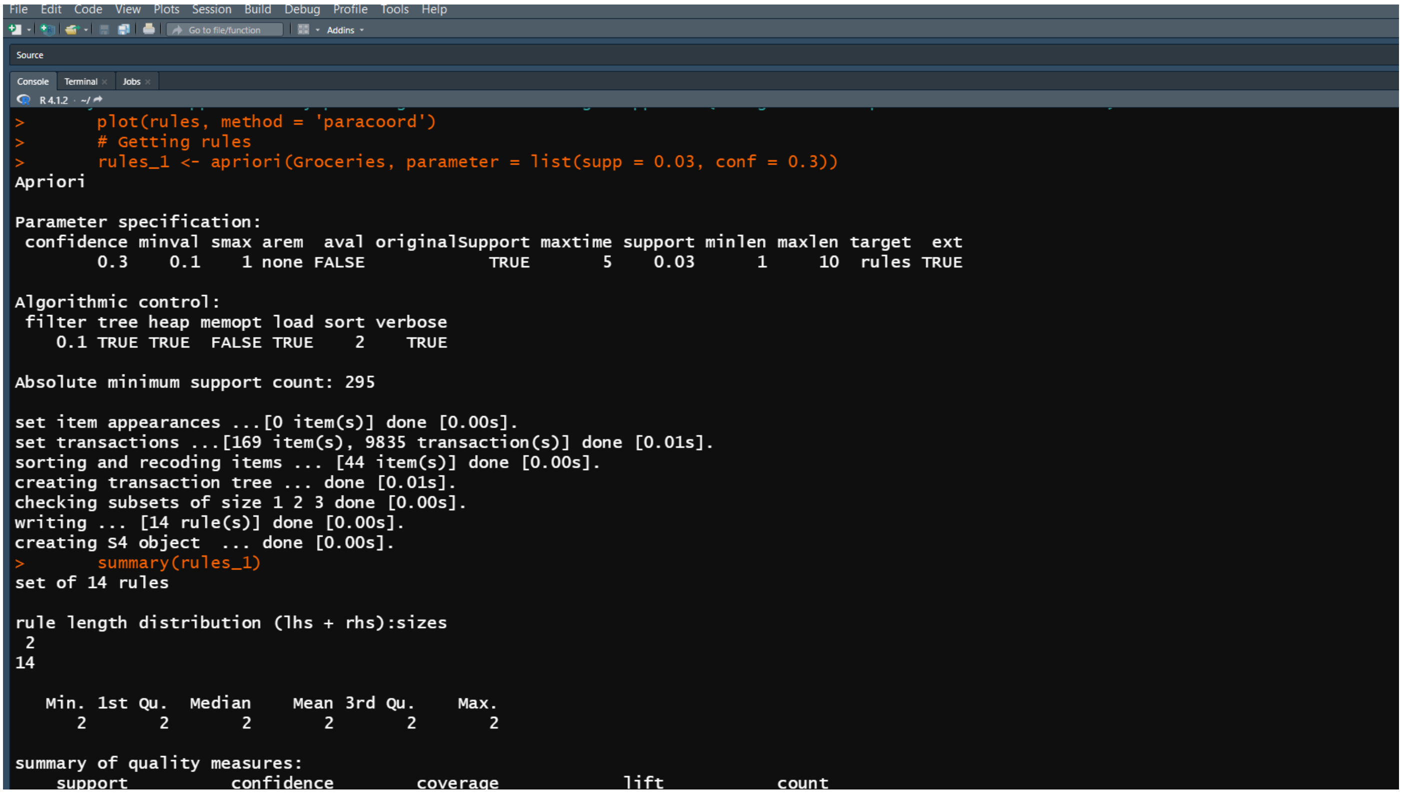
```
>       summary(rules)
set of 125 rules

rule length distribution (lhs + rhs):sizes
 2  3 
69 56 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.000   2.000   2.000   2.448   3.000   3.000 

summary of quality measures:
    support          confidence         coverage             lift           count      
 Min.   :0.01007   Min.   :0.3079   Min.   :0.01729   Min.   :1.205   Min.   : 99.0  
 1st Qu.:0.01149   1st Qu.:0.3454   1st Qu.:0.02888   1st Qu.:1.608   1st Qu.:113.0  
 Median :0.01454   Median :0.3978   Median :0.03711   Median :1.789   Median :143.0  
 Mean   :0.01859   Mean   :0.4058   Mean   :0.04783   Mean   :1.906   Mean   :182.8  
 3rd Qu.:0.02217   3rd Qu.:0.4496   3rd Qu.:0.05663   3rd Qu.:2.155   3rd Qu.:218.0  
 Max.   :0.07483   Max.   :0.5862   Max.   :0.19349   Max.   :3.295   Max.   :736.0  

mining info:
      data ntransactions support confidence                                                              call
 Groceries          9835    0.01        0.3 apriori(data = Groceries, parameter = list(supp = 0.01, conf = 0.3))
>       inspect(rules[1:10])
     lhs                  rhs                support    confidence coverage   lift     count
[1]  {hard cheese}     => {whole milk}       0.01006609 0.4107884  0.02450432 1.607682  99  
[2]  {butter milk}     => {other vegetables} 0.01037112 0.3709091  0.02796136 1.916916 102  
[3]  {butter milk}     => {whole milk}       0.01159126 0.4145455  0.02796136 1.622385 114  
[4]  {ham}             => {whole milk}       0.01148958 0.4414062  0.02602949 1.727509 113  
[5]  {sliced cheese}   => {whole milk}       0.01077783 0.4398340  0.02450432 1.721356 106  
[6]  {oil}             => {whole milk}       0.01128622 0.4021739  0.02806304 1.573968 111  
[7]  {onions}          => {other vegetables} 0.01423488 0.4590164  0.03101169 2.372268 140  
[8]  {onions}          => {whole milk}       0.01209964 0.3901639  0.03101169 1.526965 119  
[9]  {berries}         => {yogurt}           0.01057448 0.3180428  0.03324860 2.279848 104  
[10] {berries}         => {other vegetables} 0.01026945 0.3088685  0.03324860 1.596280 101  
```


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Jobs
R 4.1.2 · ~/

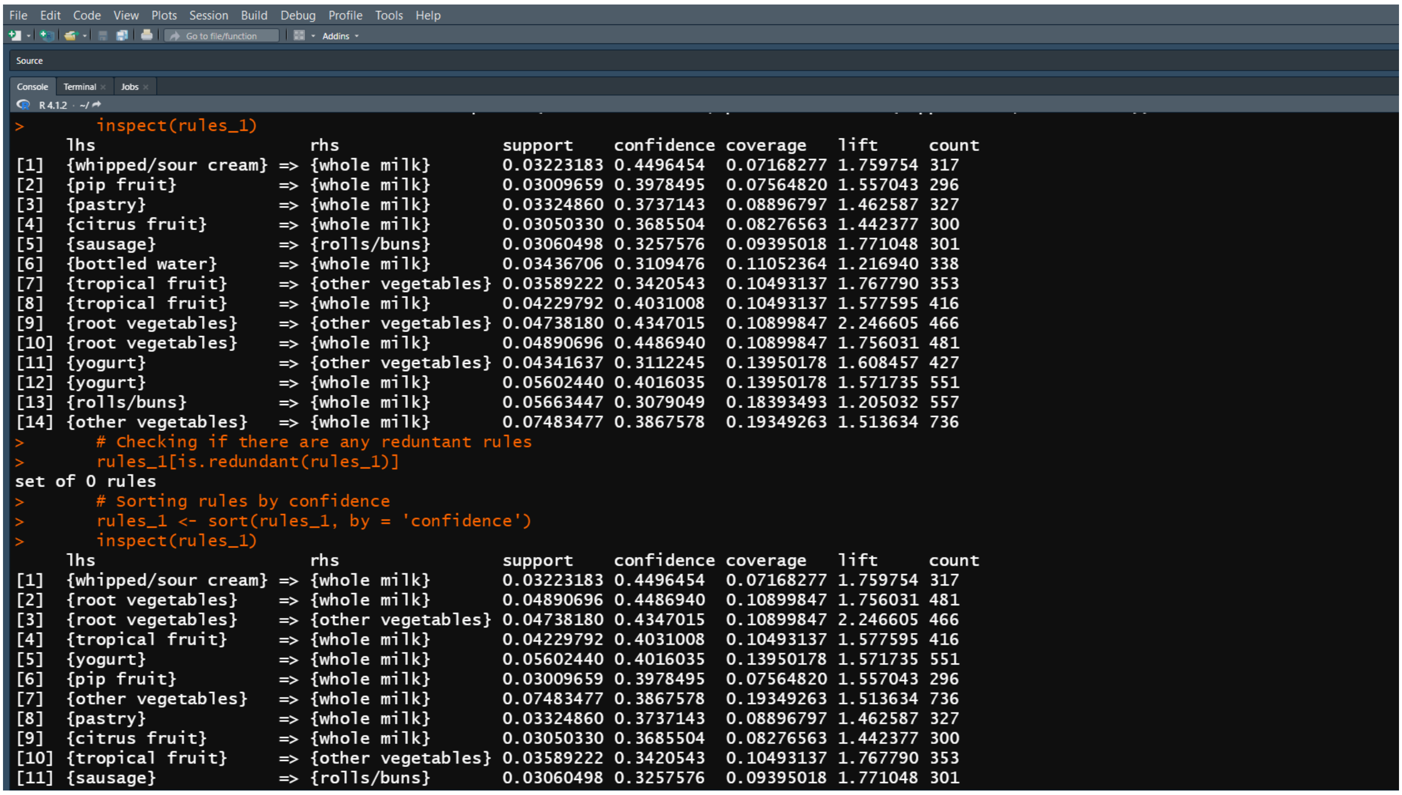
```
[5]   {sliced cheese} => {whole milk}       0.01077783 0.4398340  0.02450432 1.721356 106
[6]   {oil}           => {whole milk}       0.01128622 0.4021739  0.02806304 1.573968 111
[7]   {onions}        => {other vegetables} 0.01423488 0.4590164  0.03101169 2.372268 140
[8]   {onions}        => {whole milk}       0.01209964 0.3901639  0.03101169 1.526965 119
[9]   {berries}       => {yogurt}           0.01057448 0.3180428  0.03324860 2.279848 104
[10] {berries}        => {other vegetables} 0.01026945 0.3088685  0.03324860 1.596280 101
>       # Checking if there are any reduntant rules
>       rules[is.redundant(rules)]
set of 2 rules
>       inspect(rules[is.redundant(rules)])
    lhs                           rhs          support    confidence coverage   lift     count
[1] {sausage, other vegetables} => {whole milk} 0.01016777 0.3773585  0.02694459 1.476849 100
[2] {yogurt, soda}              => {whole milk} 0.01047280 0.3828996  0.02735130 1.498535 103
>       # Removing reduntant rules
>       rules <- rules[!is.redundant(rules)]
>       # Sorting rules by confidence
>       rules <- sort(rules, by = 'confidence')
>       inspect(rules[1:10])
     lhs                                 rhs                support    confidence coverage   lift     count
[1]  {citrus fruit, root vegetables}  => {other vegetables} 0.01037112 0.5862069  0.01769192 3.029608 102
[2]  {tropical fruit, root vegetables} => {other vegetables} 0.01230300 0.5845411  0.02104728 3.020999 121
[3]  {curd, yogurt}                   => {whole milk}       0.01006609 0.5823529  0.01728521 2.279125  99
[4]  {other vegetables, butter}       => {whole milk}       0.01148958 0.5736041  0.02003050 2.244885 113
[5]  {tropical fruit, root vegetables} => {whole milk}       0.01199797 0.5700483  0.02104728 2.230969 118
[6]  {root vegetables, yogurt}        => {whole milk}       0.01453991 0.5629921  0.02582613 2.203354 143
[7]  {other vegetables, domestic eggs} => {whole milk}       0.01230300 0.5525114  0.02226741 2.162336 121
[8]  {yogurt, whipped/sour cream}     => {whole milk}       0.01087951 0.5245098  0.02074225 2.052747 107
[9]  {root vegetables, rolls/buns}    => {whole milk}       0.01270971 0.5230126  0.02430097 2.046888 125
[10] {pip fruit, other vegetables}    => {whole milk}       0.01352313 0.5175097  0.02613116 2.025351 133
>       plot(rules, method = 'graph',measure = "confidence", shading = "support",
+            engine = "htmlwidget", control = list(max = 50))
Warning message:
Too many rules supplied. Only plotting the best 50 using ‘support’ (change control parameter max if needed).
>
```



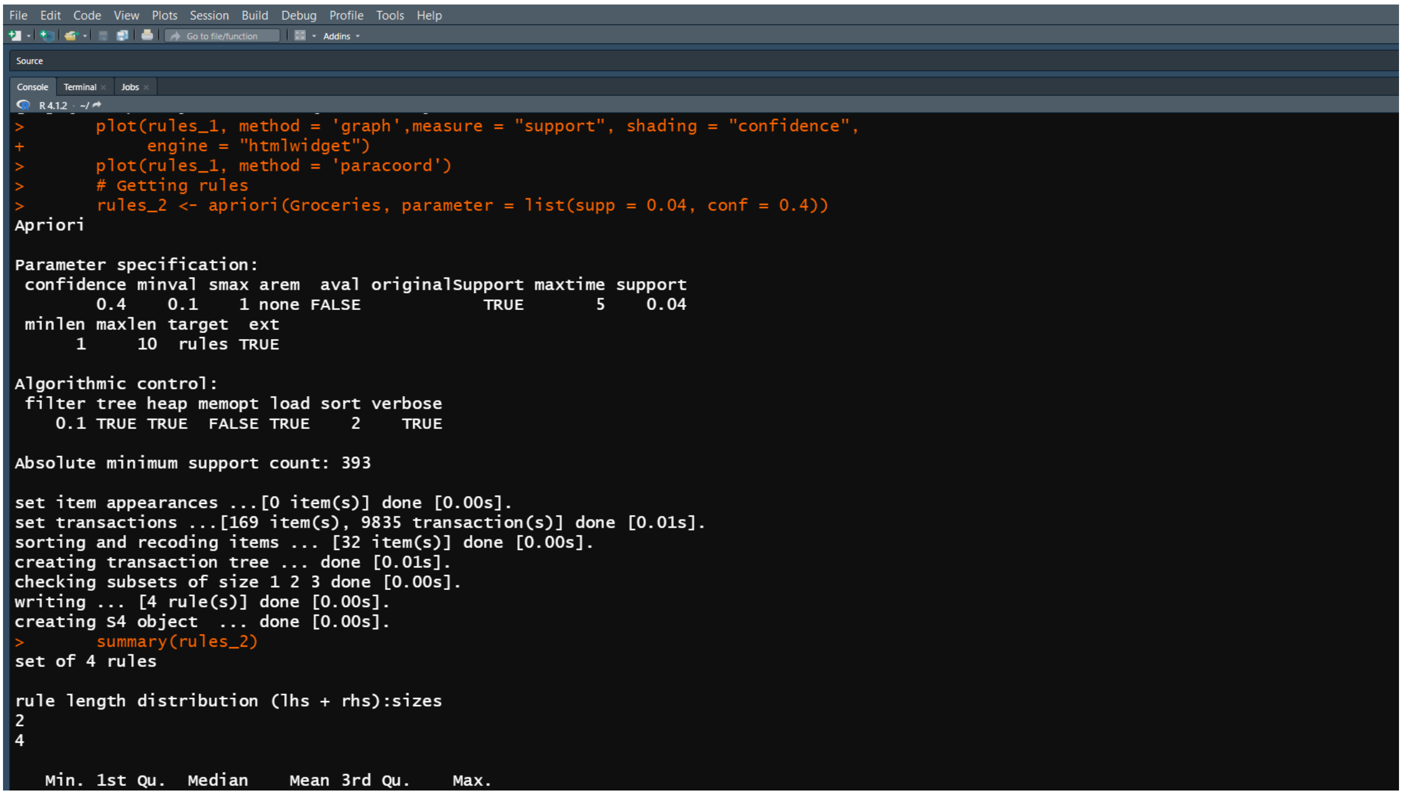



File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Jobs
R 4.1.2 · ~/

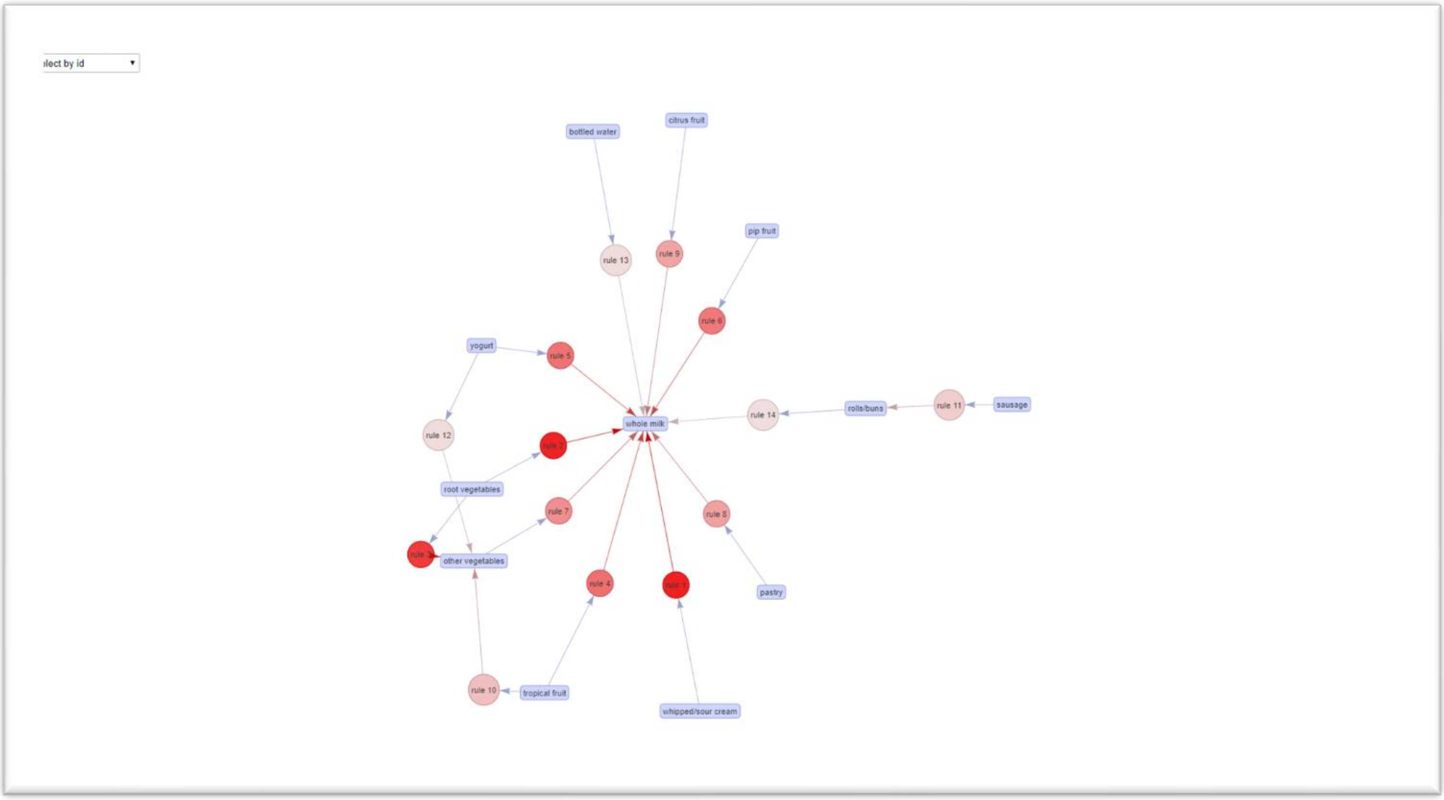
```
>       plot(rules, method = 'paracoord')
>       # Getting rules
>       rules_1 <- apriori(Groceries, parameter = list(supp = 0.03, conf = 0.3))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.3    0.1    1 none FALSE            TRUE       5    0.03      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 295

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [44 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [14 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>       summary(rules_1)
set of 14 rules

rule length distribution (lhs + rhs):sizes
 2 
14 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      2       2       2       2       2       2 

summary of quality measures:
    support           confidence        coverage            lift            count
```

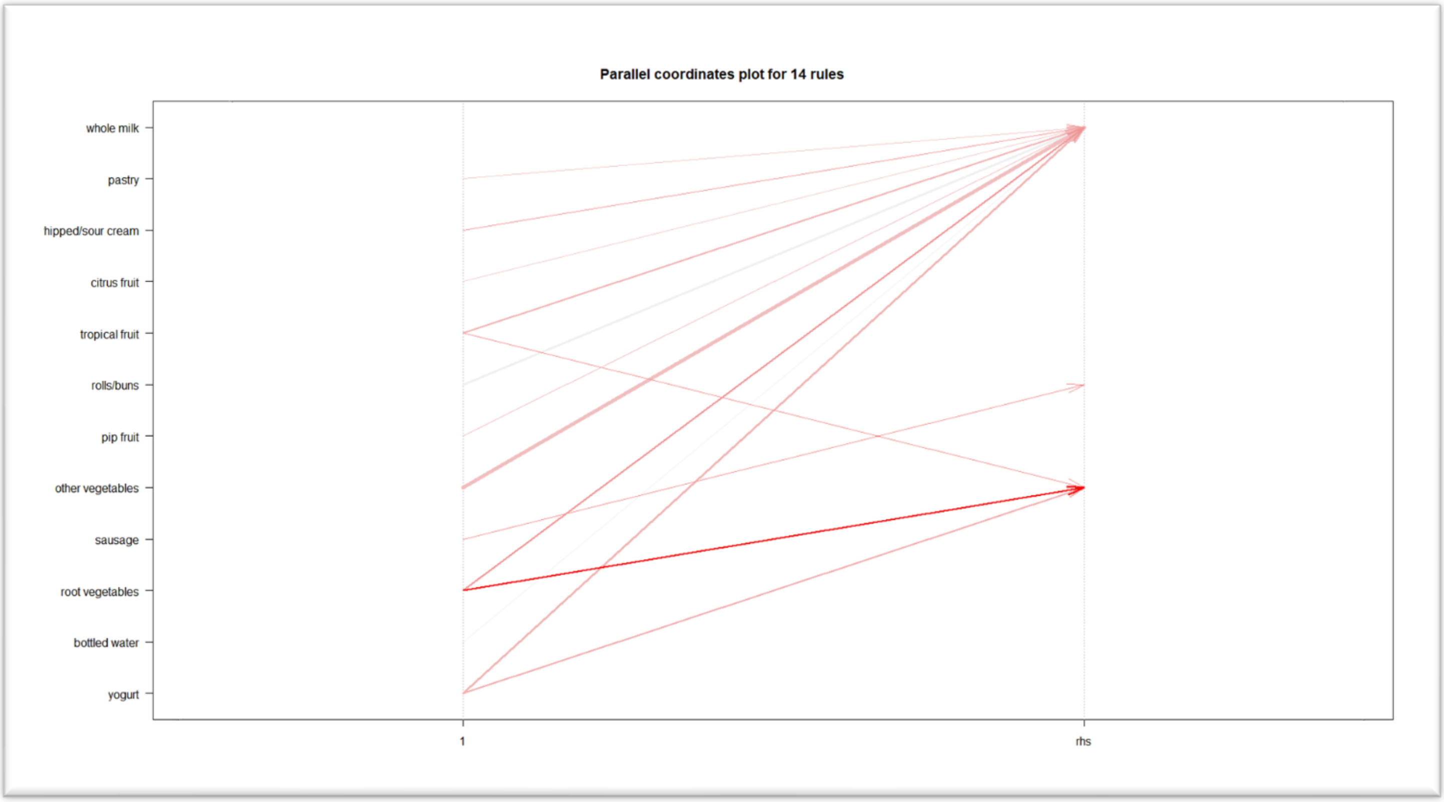

File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Jobs
R 4.1.2 · ~/

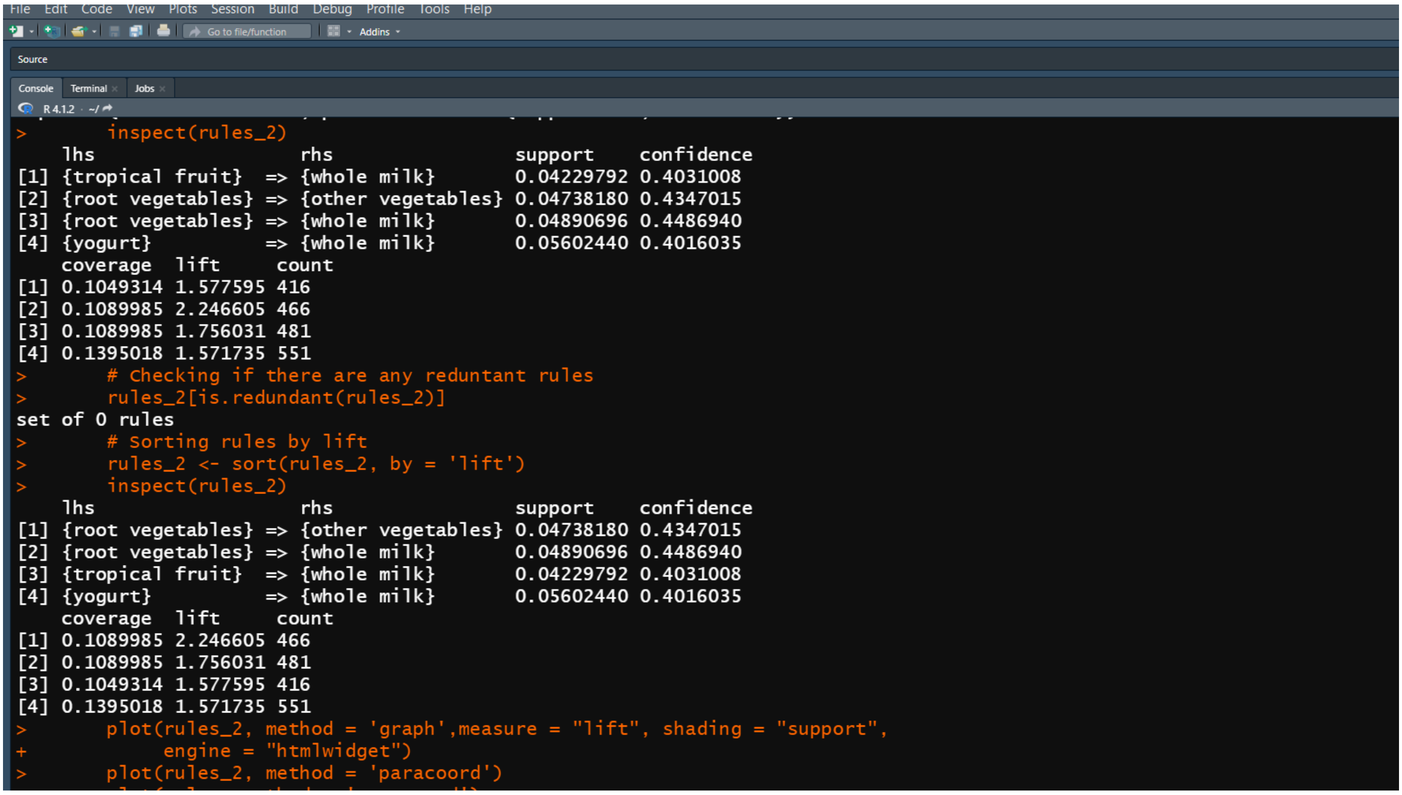
```
>       inspect(rules_1)
     lhs                    rhs                support    confidence coverage   lift     count
[1]  {whipped/sour cream} => {whole milk}       0.03223183 0.4496454  0.07168277 1.759754 317
[2]  {pip fruit}          => {whole milk}       0.03009659 0.3978495  0.07564820 1.557043 296
[3]  {pastry}             => {whole milk}       0.03324860 0.3737143  0.08896797 1.462587 327
[4]  {citrus fruit}       => {whole milk}       0.03050330 0.3685504  0.08276563 1.442377 300
[5]  {sausage}            => {rolls/buns}       0.03060498 0.3257576  0.09395018 1.771048 301
[6]  {bottled water}      => {whole milk}       0.03436706 0.3109476  0.11052364 1.216940 338
[7]  {tropical fruit}     => {other vegetables} 0.03589222 0.3420543  0.10493137 1.767790 353
[8]  {tropical fruit}     => {whole milk}       0.04229792 0.4031008  0.10493137 1.577595 416
[9]  {root vegetables}    => {other vegetables} 0.04738180 0.4347015  0.10899847 2.246605 466
[10] {root vegetables}    => {whole milk}       0.04890696 0.4486940  0.10899847 1.756031 481
[11] {yogurt}             => {other vegetables} 0.04341637 0.3112245  0.13950178 1.608457 427
[12] {yogurt}             => {whole milk}       0.05602440 0.4016035  0.13950178 1.571735 551
[13] {rolls/buns}         => {whole milk}       0.05663447 0.3079049  0.18393493 1.205032 557
[14] {other vegetables}   => {whole milk}       0.07483477 0.3867578  0.19349263 1.513634 736
>       # Checking if there are any reduntant rules
>       rules_1[is.redundant(rules_1)]
set of 0 rules
>       # Sorting rules by confidence
>       rules_1 <- sort(rules_1, by = 'confidence')
>       inspect(rules_1)
     lhs                    rhs                support    confidence coverage   lift     count
[1]  {whipped/sour cream} => {whole milk}       0.03223183 0.4496454  0.07168277 1.759754 317
[2]  {root vegetables}    => {whole milk}       0.04890696 0.4486940  0.10899847 1.756031 481
[3]  {root vegetables}    => {other vegetables} 0.04738180 0.4347015  0.10899847 2.246605 466
[4]  {tropical fruit}     => {whole milk}       0.04229792 0.4031008  0.10493137 1.577595 416
[5]  {yogurt}             => {whole milk}       0.05602440 0.4016035  0.13950178 1.571735 551
[6]  {pip fruit}          => {whole milk}       0.03009659 0.3978495  0.07564820 1.557043 296
[7]  {other vegetables}   => {whole milk}       0.07483477 0.3867578  0.19349263 1.513634 736
[8]  {pastry}             => {whole milk}       0.03324860 0.3737143  0.08896797 1.462587 327
[9]  {citrus fruit}       => {whole milk}       0.03050330 0.3685504  0.08276563 1.442377 300
[10] {tropical fruit}     => {other vegetables} 0.03589222 0.3420543  0.10493137 1.767790 353
[11] {sausage}            => {rolls/buns}       0.03060498 0.3257576  0.09395018 1.771048 301
```


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Jobs
R 4.1.2 · ~/

```
>       plot(rules_1, method = 'graph',measure = "support", shading = "confidence",
+            engine = "htmlwidget")
>       plot(rules_1, method = 'paracoord')
>       # Getting rules
>       rules_2 <- apriori(Groceries, parameter = list(supp = 0.04, conf = 0.4))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support
        0.4    0.1    1 none FALSE            TRUE       5    0.04
 minlen maxlen target  ext
      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 393

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [32 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [4 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>       summary(rules_2)
set of 4 rules

rule length distribution (lhs + rhs):sizes
2
4

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

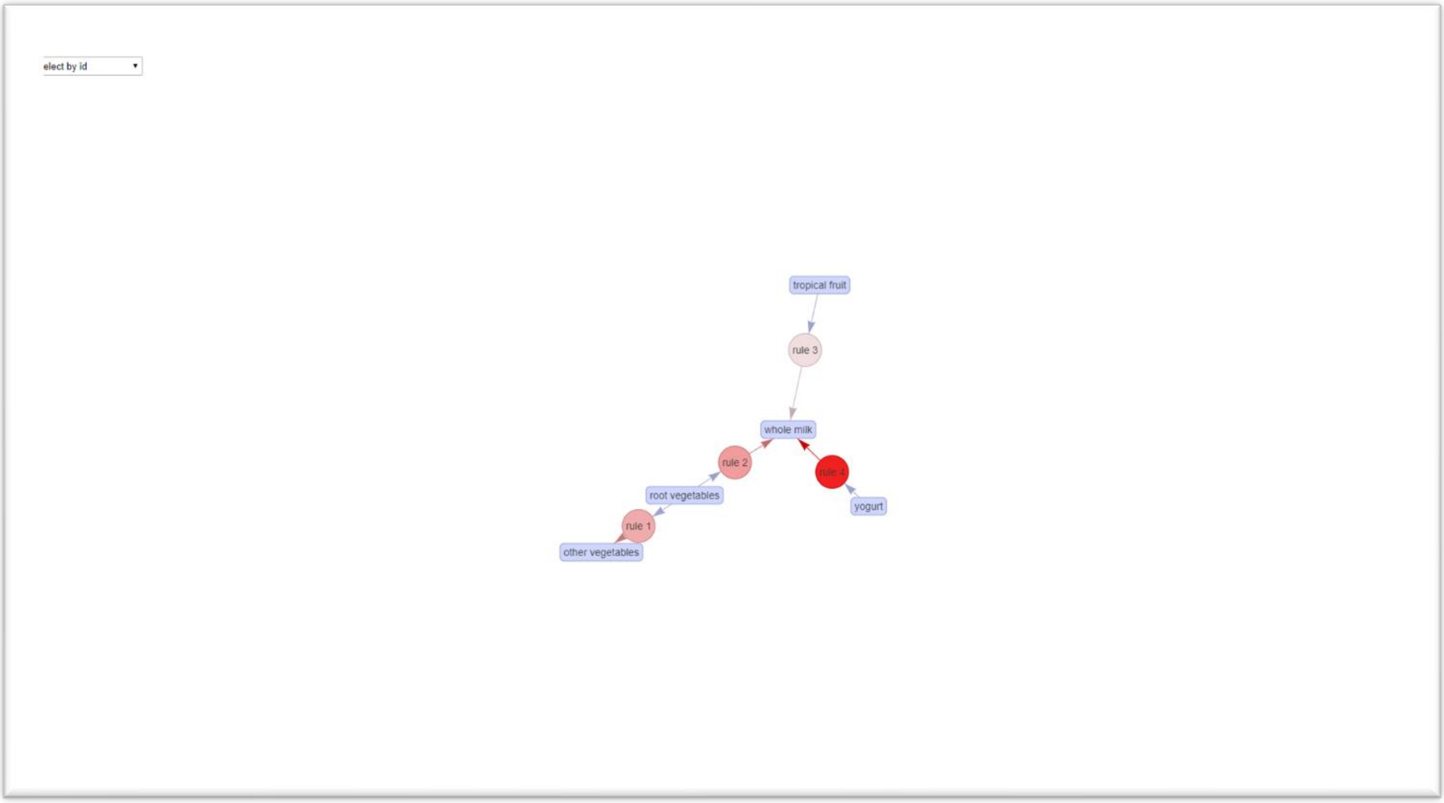

elect by id
bottled water
citrus fruit
pip fruit
rule 13
rule 9
rule 6
yogurt
rule 5
rule 12
whole milk
rule 14
rolls/buns
rule 11
sausage
rule 2
root vegetables
rule 7
rule 8
rule 3
other vegetables
rule 4
rule 1
pastry
rule 10
tropical fruit
whipped/sour cream

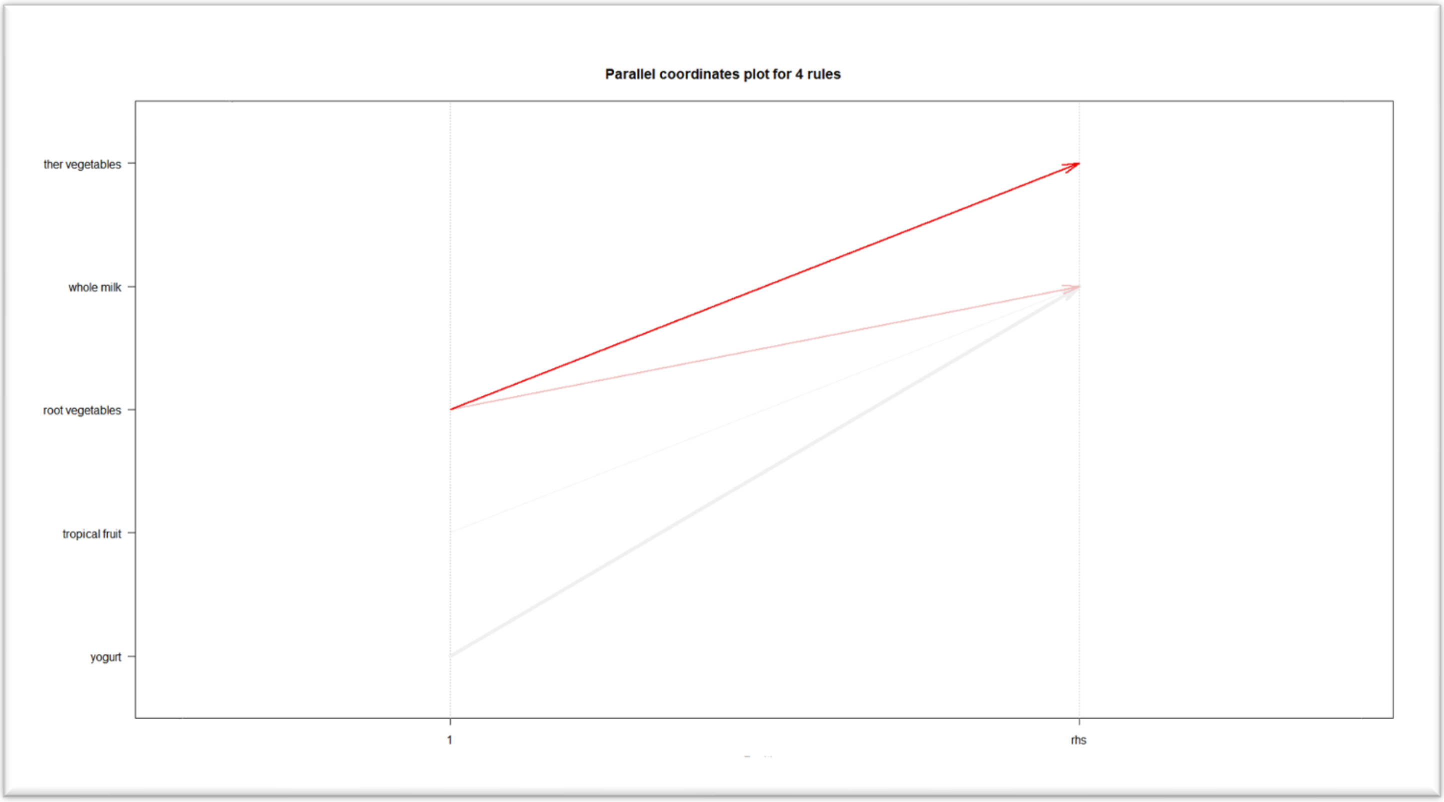

Parallel coordinates plot for 14 rules
whole milk
pastry
hipped/sour cream
citrus fruit
tropical fruit
rolls/buns
pip fruit
other vegetables
sausage
root vegetables
bottled water
yogurt
1
rhs



File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Jobs
R 4.1.2 · ~/

```
>       inspect(rules_2)
    lhs                   rhs                support    confidence
[1] {tropical fruit}  => {whole milk}       0.04229792 0.4031008
[2] {root vegetables} => {other vegetables} 0.04738180 0.4347015
[3] {root vegetables} => {whole milk}       0.04890696 0.4486940
[4] {yogurt}          => {whole milk}       0.05602440 0.4016035
    coverage  lift     count
[1] 0.1049314 1.577595 416
[2] 0.1089985 2.246605 466
[3] 0.1089985 1.756031 481
[4] 0.1395018 1.571735 551
>       # Checking if there are any reduntant rules
>       rules_2[is.redundant(rules_2)]
set of 0 rules
>       # Sorting rules by lift
>       rules_2 <- sort(rules_2, by = 'lift')
>       inspect(rules_2)
    lhs                   rhs                support    confidence
[1] {root vegetables} => {other vegetables} 0.04738180 0.4347015
[2] {root vegetables} => {whole milk}       0.04890696 0.4486940
[3] {tropical fruit}  => {whole milk}       0.04229792 0.4031008
[4] {yogurt}          => {whole milk}       0.05602440 0.4016035
    coverage  lift     count
[1] 0.1089985 2.246605 466
[2] 0.1089985 1.756031 481
[3] 0.1049314 1.577595 416
[4] 0.1395018 1.571735 551
>       plot(rules_2, method = 'graph',measure = "lift", shading = "support",
+            engine = "htmlwidget")
>       plot(rules_2, method = 'paracoord')
```




Parallel coordinates plot for 4 rules
ther vegetables
whole milk
root vegetables
tropical fruit
yogurt
1
rhs