Parin Patel

Loan Prediction: Association Rules to Predict PEP Request

Introduction:

Personal Equity Plans (PEP) is a tax-free investment plan that allows those over the age of 18 to invest shares in companies. The overall goal of the plan is the encourage individual investment, through an approved plan and investment trust. One of the main incentives of choosing this investment path was that PEP garner higher capital growth at a greater rate than many other forms of investment.

To determine if a banks customer will want to obtain a PEP, exploration of the bank dataset must occur. This dataset contains attributes related to each person's demographic and banking information. By running Association Rules on the bank dataset, we can better predict which customers will want to obtain a PEP. Specifically, we will utilize an algorithm to find relationships among the variables by finding how often (frequent) a combination of items appears in the data. The frequent combinations in the data are called rules. By finding the rules from data, we can create reliable predictions that will ultimately help us better serve our customers and expand our network of buyers.

Note: Association Rule discovery using Weka was conducted on the bank dataset. However, due to a coffee-driven curiosity, Association Rule discovery was also conducted in R to generate plots and other visualizations to better demonstrate to the client. The R-walk-through for the Apriori algorithm on the bank dataset can be found at the end of each of the following sections.

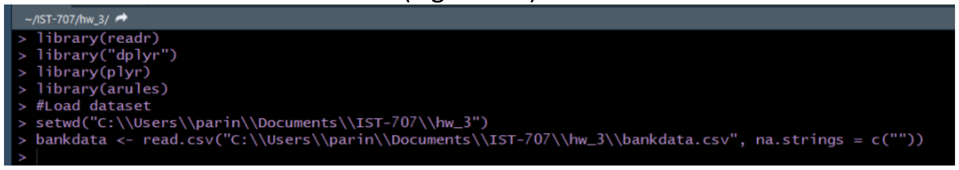
Analysis:

The Data:

The bank dataset contained information regarding bank customer demographic, income, car ownership, account ownership, marital status, and personal equity plan purchase decisions. Additionally, each customer is given a unique identification number.

Weka: In Weka, the data was loaded as a .csv file, and then saved as a .arff file.

R: First the environment was set up. We installed and loaded the following packages: readr, arules, dplyr, and plyr. Second, we set the working directory and loaded the .csv bank dataset and saved it as a new data frame (Figure 1.1)



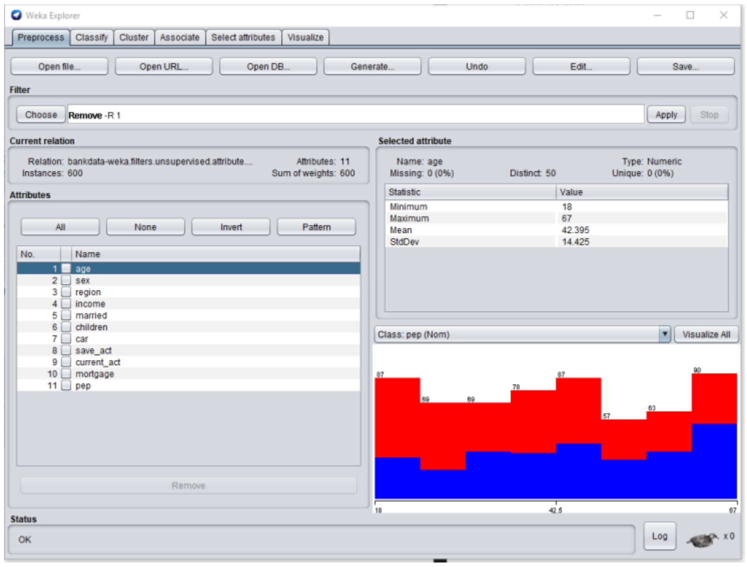
```
~/IST-707/hw_3/
> library(readr)
> library("dplyr")
> library(plyr)
> library(arules)
> #Load dataset
> setwd("C:\\Users\\parin\\Documents\\IST-707\\hw_3")
> bankdata <- read.csv("C:\\Users\\parin\\Documents\\IST-707\\hw_3\\bankdata.csv", na.strings = c(""))
>
```

Figure 1.1

Parin Patel

Data Preprocessing:

<u>Weka</u>: First, the ID column was removed using the remove-R1 filter method in the application. Figure 1.2 shows the attributes after.



Filter: Remove -R 1

Current relation

Relation: bankdata-weka.filters.unsupervised.attribute.... Attributes: 11
Instances: 600 Sum of weights: 600

Selected attribute

Name: age Type: Numeric
Missing: 0 (0%) Distinct: 50 Unique: 0 (0%)

Statistic	Value
Minimum	18
Maximum	67
Mean	42.395
StdDev	14.425

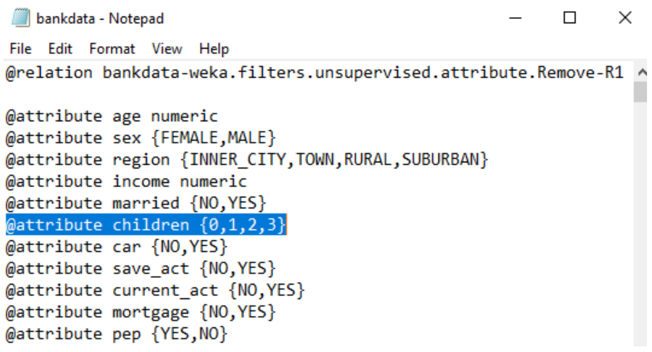
Attributes

No.	Name
1	age
2	sex
3	region
4	income
5	married
6	children
7	car
8	save_act
9	current_act
10	mortgage
11	pep

Status: OK

Figure 1.2

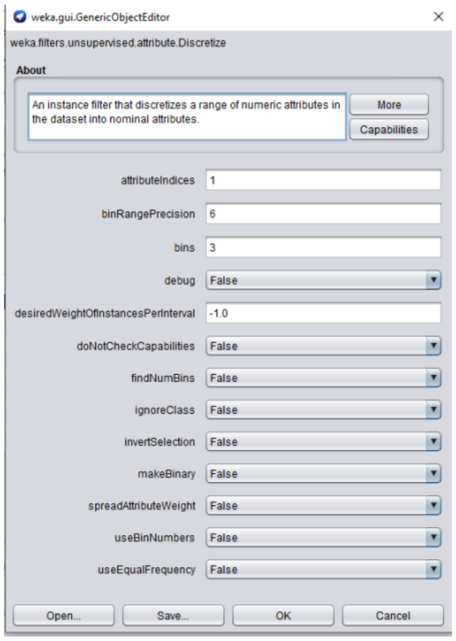
Next, we discretize income, age, and children. To discretize children, we will need to open the file as a notepad document. Next, we will simply remove the keyword "numeric" as the type for the "children" attribute in the ARFF file and replacing it with the set of discrete values as seen in Figure 1.3.



```
@relation bankdata-weka.filters.unsupervised.attribute.Remove-R1

@attribute age numeric
@attribute sex {FEMALE,MALE}
@attribute region {INNER_CITY,TOWN,RURAL,SUBURBAN}
@attribute income numeric
@attribute married {NO,YES}
@attribute children {0,1,2,3}
@attribute car {NO,YES}
@attribute save_act {NO,YES}
@attribute current_act {NO,YES}
@attribute mortgage {NO,YES}
@attribute pep {YES,NO}
```

Figure 1.3

To discretize income and age, the filter.unsupervised.attribute.Discretize function in the Weka Explorer was used. We chose to divide both of these attributes into three intervals (bins). Figure 1.4.1 shows the object editor, while Figure 1.4.2 will show the object header for age. Figure 1.4.3 will show the object header for income.

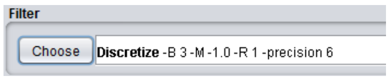


weka.filters.unsupervised.attribute.Discretize

An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes.

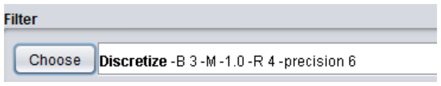
attributeIndices	1
binRangePrecision	6
bins	3
debug	False
desiredWeightOfInstancesPerInterval	-1.0
doNotCheckCapabilities	False
findNumBins	False
ignoreClass	False
invertSelection	False
makeBinary	False
spreadAttributeWeight	False
useBinNumbers	False
useEqualFrequency	False

Figure 1.4 1



Discretize -B 3 -M -1.0 -R 1 -precision 6

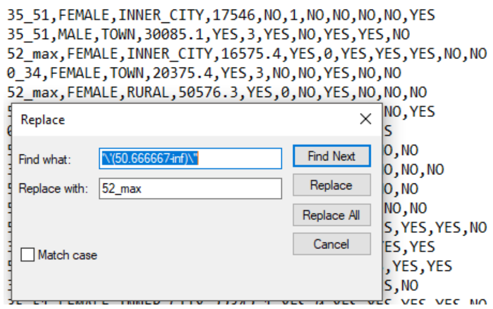
Figure 1.4 2



Discretize -B 3 -M -1.0 -R 4 -precision 6

Figure 1.4 3

Additionally, to maintain clear labels for standard naming conventions, a cleaning step on the notepad data of the newly discretized bank data was needed. The find-and-replace button was used to update the original labels with more condensed versions (Figure 1.5).



```
35_51,FEMALE,INNER_CITY,17546,NO,1,NO,NO,NO,NO,YES
35_51,MALE,TOWN,30085.1,YES,3,YES,NO,YES,YES,NO
52_max,FEMALE,INNER_CITY,16575.4,YES,0,YES,YES,YES,NO,NO
0_34,FEMALE,TOWN,20375.4,YES,3,NO,NO,YES,NO,NO
52_max,FEMALE,RURAL,50576.3,YES,0,NO,YES,NO,NO,NO
```

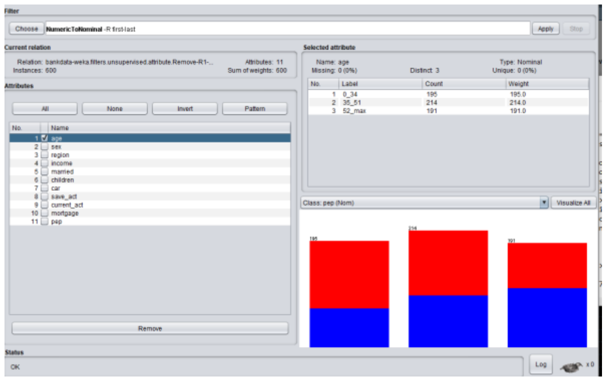
Find what: '\'(50.666667-inf)\''

Replace with: 52_max

Figure 1.5

Finally, we run a filter function called NumericToNominal on all attributes.

Parin Patel



Filter: NumericToNominal -R first-last

Current relation — Relation: bankdata-weka.filters.unsupervised.attribute.Remove-R1-... Instances: 600 Attributes: 11 Sum of weights: 600

Selected attribute — Name: age Type: Nominal Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

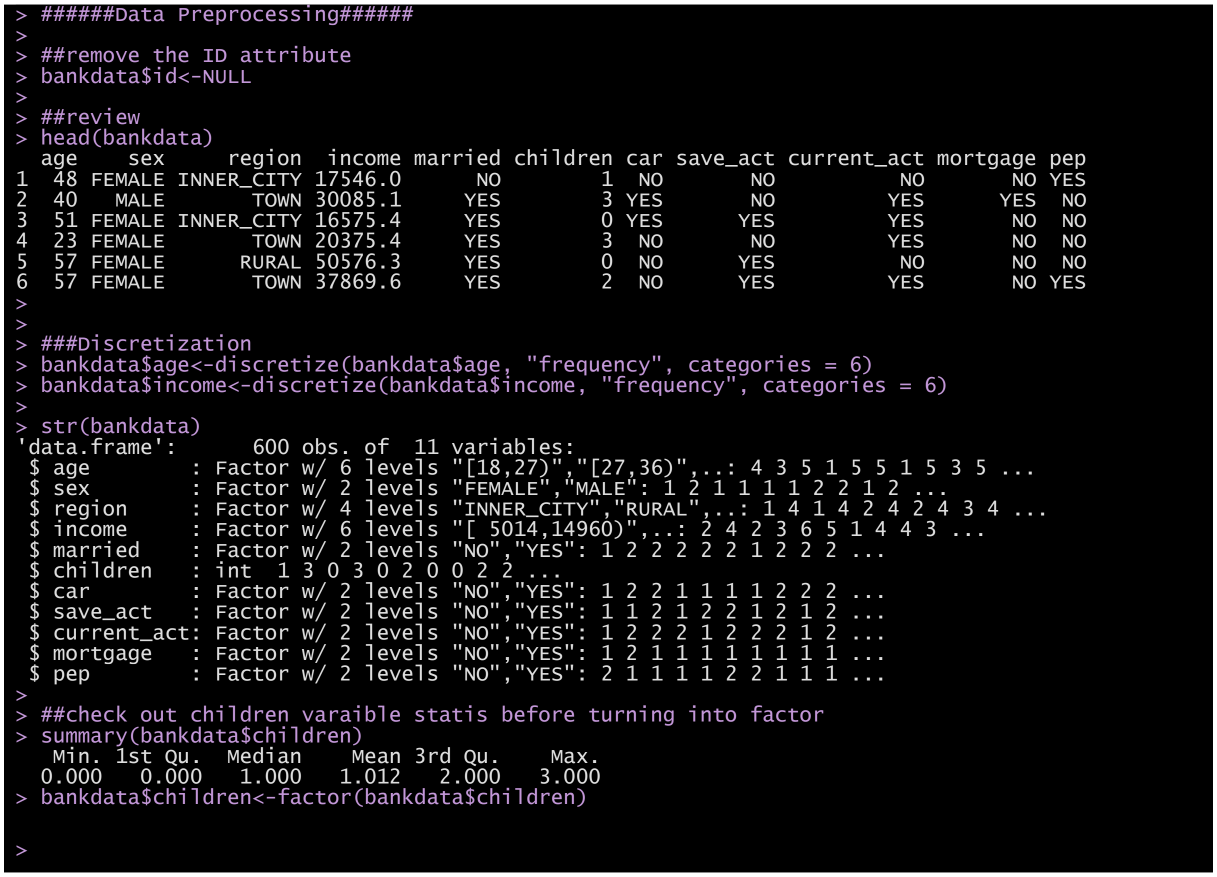
No.	Label	Count	Weight
1	0_34	195	195.0
2	35_51	214	214.0
3	52_max	191	191.0

Attributes:

No.	Name
1	age
2	sex
3	region
4	income
5	married
6	children
7	car
8	save_act
9	current_act
10	mortgage
11	pep

Class: pep (Nom)

<u>In R:</u> The steps conducted in R are similar to what was done in Weka. We removed the ID attribute first, then worked toward discretization of age and income. To convert them to factor variables we run the unsupervised discretion filter with equal binning frequency. And finally converted the children variable into factor since the Apirori rules method requires all variables be discrete or factor.



```
> ######Data Preprocessing######
>
> ##remove the ID attribute
> bankdata$id<-NULL
>
> ##review
> head(bankdata)
  age    sex     region  income married children car save_act current_act mortgage pep
1  48 FEMALE INNER_CITY 17546.0      NO        1  NO       NO          NO       NO YES
2  40   MALE       TOWN 30085.1     YES        3 YES       NO         YES      YES  NO
3  51 FEMALE INNER_CITY 16575.4     YES        0 YES      YES         YES       NO  NO
4  23 FEMALE       TOWN 20375.4     YES        3  NO       NO         YES       NO  NO
5  57 FEMALE      RURAL 50576.3     YES        0  NO      YES          NO       NO  NO
6  57 FEMALE       TOWN 37869.6     YES        2  NO      YES         YES       NO YES
>
>
> ###Discretization
> bankdata$age<-discretize(bankdata$age, "frequency", categories = 6)
> bankdata$income<-discretize(bankdata$income, "frequency", categories = 6)
>
> str(bankdata)
'data.frame':      600 obs. of  11 variables:
 $ age        : Factor w/ 6 levels "[18,27)","[27,36)",..: 4 3 5 1 5 5 1 5 3 5 ...
 $ sex        : Factor w/ 2 levels "FEMALE","MALE": 1 2 1 1 1 1 2 2 1 2 ...
 $ region     : Factor w/ 4 levels "INNER_CITY","RURAL",..: 1 4 1 4 2 4 2 4 3 4 ...
 $ income     : Factor w/ 6 levels "[ 5014,14960)",..: 2 4 2 3 6 5 1 4 4 3 ...
 $ married    : Factor w/ 2 levels "NO","YES": 1 2 2 2 2 2 1 2 2 2 ...
 $ children   : int  1 3 0 3 0 2 0 0 2 2 ...
 $ car        : Factor w/ 2 levels "NO","YES": 1 2 2 1 1 1 1 2 2 2 ...
 $ save_act   : Factor w/ 2 levels "NO","YES": 1 1 2 1 2 2 1 2 1 2 ...
 $ current_act: Factor w/ 2 levels "NO","YES": 1 2 2 2 1 2 2 2 1 2 ...
 $ mortgage   : Factor w/ 2 levels "NO","YES": 1 2 1 1 1 1 1 1 1 1 ...
 $ pep        : Factor w/ 2 levels "NO","YES": 2 1 1 1 1 2 2 1 1 1 ...
>
> ##check out children varaible statis before turning into factor
> summary(bankdata$children)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   1.000   1.012   2.000   3.000
> bankdata$children<-factor(bankdata$children)

>
```

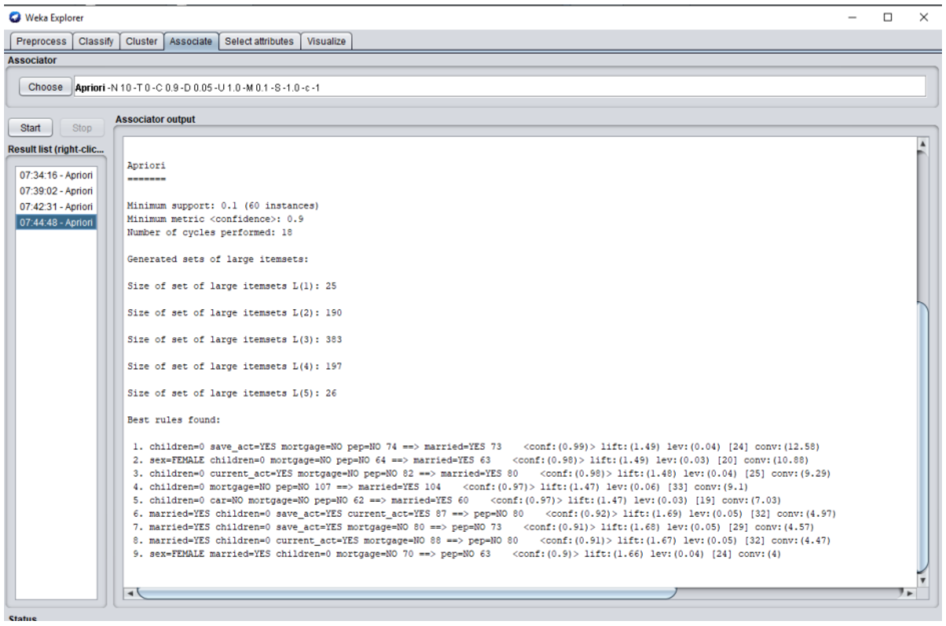
Association Rule Discovery & Description of Parameters and Experiments:

Weka: In Weka we can adjust our parameters to get more meaningful results. We have the option to adjust the following parameters:

- Delta – which is the decrease rate of minimum support. The minimum support value will get lowered by this rate until the lowerBoundMinSupport or the numRules is reached.
- lowerBoundMinSupport- lowest bound for minimum support
- metricType- the approach to ranking rule.
- minMetric- the threshold for selecting rules; your confidence value
- numRules- the confidence threshold, the number of best rules to be shown
- outputItemSets- whether to output the item sets and their frequencies.
- upperBoundMinSupport- the upper bound for minimum support

In general, rules that satisfy both a minimum support threshold and a minimum confidence threshold are considered strong rules. These rules must be frequent itemsets and must satisfy min_support and min_confidence.

In this example (Figure 2.1), we have chosen to the metric confidence as the criteria. We chose to set the metricType to confidence and set the minMetric to 0.9. The higher the confidence, the more positive the rule. From the output we can see that the algorithm went through 18 cycles, with 60 instances. It ultimately returned 9 rules, if we wanted more we could have lowered our support and minMentric to 0.7.

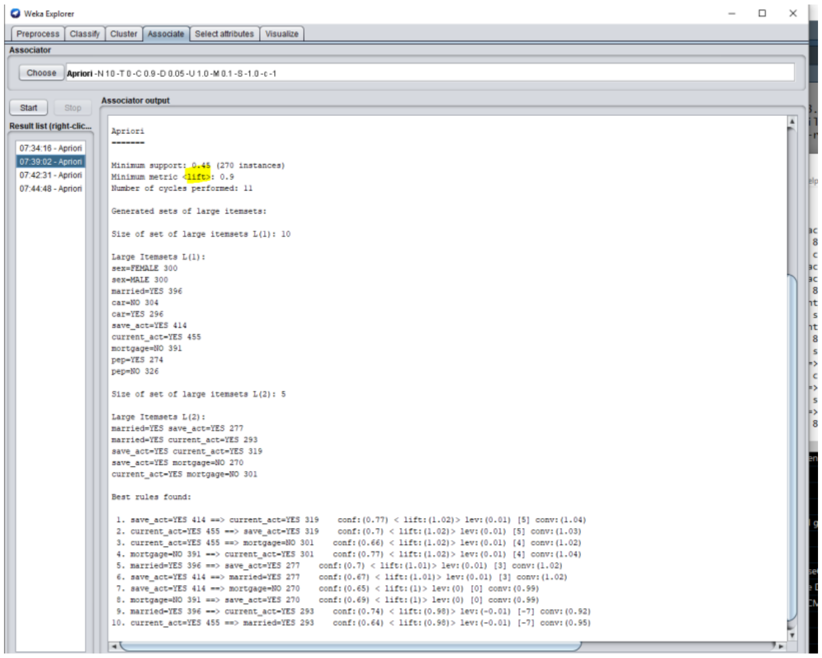


```
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
```

```
Apriori
=======

Minimum support: 0.1 (60 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 25

Size of set of large itemsets L(2): 190

Size of set of large itemsets L(3): 383

Size of set of large itemsets L(4): 197

Size of set of large itemsets L(5): 26

Best rules found:

 1. children=0 save_act=YES mortgage=NO pep=NO 74 ==> married=YES 73    <conf:(0.99)> lift:(1.49) lev:(0.04) [24] conv:(12.58)
 2. sex=FEMALE children=0 mortgage=NO pep=NO 64 ==> married=YES 63    <conf:(0.98)> lift:(1.49) lev:(0.03) [20] conv:(10.88)
 3. children=0 current_act=YES mortgage=NO pep=NO 82 ==> married=YES 80    <conf:(0.98)> lift:(1.48) lev:(0.04) [25] conv:(9.29)
 4. children=0 mortgage=NO pep=NO 107 ==> married=YES 104    <conf:(0.97)> lift:(1.47) lev:(0.06) [33] conv:(9.1)
 5. children=0 car=NO mortgage=NO pep=NO 62 ==> married=YES 60    <conf:(0.97)> lift:(1.47) lev:(0.03) [19] conv:(7.03)
 6. married=YES children=0 save_act=YES current_act=YES 87 ==> pep=NO 80    <conf:(0.92)> lift:(1.69) lev:(0.05) [32] conv:(4.97)
 7. married=YES children=0 save_act=YES mortgage=NO 80 ==> pep=NO 73    <conf:(0.91)> lift:(1.68) lev:(0.05) [29] conv:(4.57)
 8. married=YES children=0 current_act=YES mortgage=NO 88 ==> pep=NO 80    <conf:(0.91)> lift:(1.67) lev:(0.05) [32] conv:(4.47)
 9. sex=FEMALE married=YES children=0 mortgage=NO 70 ==> pep=NO 63    <conf:(0.9)> lift:(1.66) lev:(0.04) [24] conv:(4)
```

Figure 2 1

For the second metricType, we chose to adjust the parameter to measure Lift. We set the confidence value at 0.9 again. The itemset generated 270 instances over 11 cycles. Additionally, it found 10 rules (Figure 2.2)

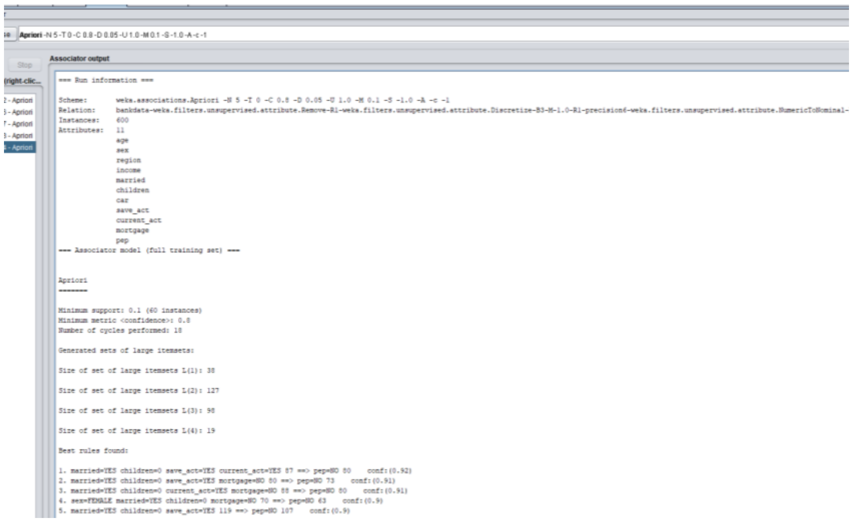


```
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
```

```
Apriori
=======

Minimum support: 0.45 (270 instances)
Minimum metric <lift>: 0.9
Number of cycles performed: 11

Generated sets of large itemsets:

Size of set of large itemsets L(1): 10

Large Itemsets L(1):
sex=FEMALE 300
sex=MALE 300
married=YES 396
car=NO 304
car=YES 296
save_act=YES 414
current_act=YES 455
mortgage=NO 391
pep=YES 274
pep=NO 326

Size of set of large itemsets L(2): 5

Large Itemsets L(2):
married=YES save_act=YES 277
married=YES current_act=YES 293
save_act=YES current_act=YES 319
save_act=YES mortgage=NO 270
current_act=YES mortgage=NO 301

Best rules found:

 1. save_act=YES 414 ==> current_act=YES 319    conf:(0.77) < lift:(1.02)> lev:(0.01) [5] conv:(1.04)
 2. current_act=YES 455 ==> save_act=YES 319    conf:(0.7) < lift:(1.02)> lev:(0.01) [5] conv:(1.03)
 3. current_act=YES 455 ==> mortgage=NO 301    conf:(0.66) < lift:(1.02)> lev:(0.01) [4] conv:(1.02)
 4. mortgage=NO 391 ==> current_act=YES 301    conf:(0.77) < lift:(1.02)> lev:(0.01) [4] conv:(1.04)
 5. married=YES 396 ==> save_act=YES 277    conf:(0.7) < lift:(1.01)> lev:(0.01) [3] conv:(1.02)
 6. save_act=YES 414 ==> married=YES 277    conf:(0.67) < lift:(1.01)> lev:(0.01) [3] conv:(1.02)
 7. save_act=YES 414 ==> mortgage=NO 270    conf:(0.65) < lift:(1)> lev:(0) [0] conv:(0.99)
 8. mortgage=NO 391 ==> save_act=YES 270    conf:(0.69) < lift:(1)> lev:(0) [0] conv:(0.99)
 9. married=YES 396 ==> current_act=YES 293    conf:(0.74) < lift:(0.98)> lev:(-0.01) [-7] conv:(0.92)
10. current_act=YES 455 ==> married=YES 293    conf:(0.64) < lift:(0.98)> lev:(-0.01) [-7] conv:(0.95)
```

Figure 2 2

Finally, in Weka, we looked at the association rules for classifictioon. We restricted the right hand side of the rules to be the label PEP. "Car" was turned on in the Object Editor, and classIndex was kept as the default -1, since PEP was the last column. Finally, Metric Type wa set to confidence. The confidence value was set to 0.8 to reach 4 rules. Previously, when set at 0.9, only four rules qere generated.



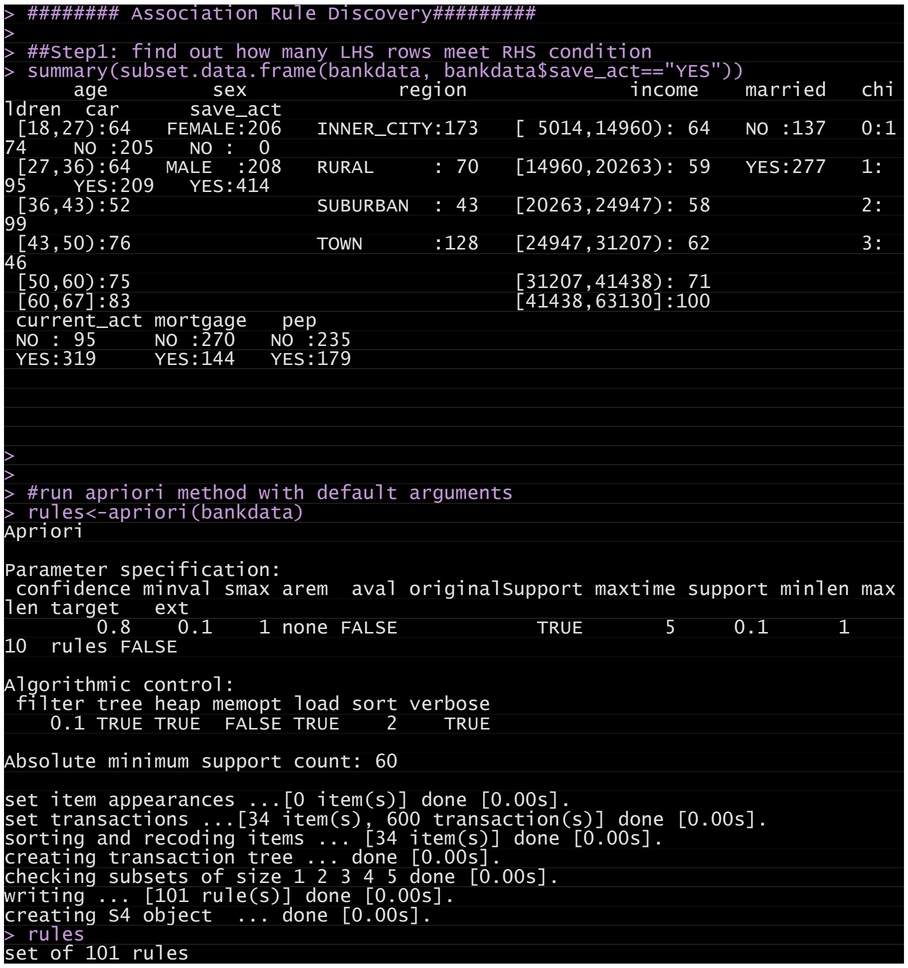
```
Apriori -N 5 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -A -c -1
```

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 5 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -A -c -1
Relation:     bankdata-weka.filters.unsupervised.attribute.Remove-R1-weka.filters.unsupervised.attribute.Discretize-B3-M-1.0-R1-precision6-weka.filters.unsupervised.attribute.NumericToNominal-
Instances:    600
Attributes:   11
              age
              sex
              region
              income
              married
              children
              car
              save_act
              current_act
              mortgage
              pep
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.1 (60 instances)
Minimum metric <confidence>: 0.8
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 38

Size of set of large itemsets L(2): 127

Size of set of large itemsets L(3): 98

Size of set of large itemsets L(4): 19

Best rules found:

1. married=YES children=0 save_act=YES current_act=YES 87 ==> pep=NO 80    conf:(0.92)
2. married=YES children=0 save_act=YES mortgage=NO 80 ==> pep=NO 73    conf:(0.91)
3. married=YES children=0 current_act=YES mortgage=NO 88 ==> pep=NO 80    conf:(0.91)
4. sex=FEMALE married=YES children=0 mortgage=NO 70 ==> pep=NO 63    conf:(0.9)
5. married=YES children=0 save_act=YES 119 ==> pep=NO 107    conf:(0.9)
```

Figure 2 3

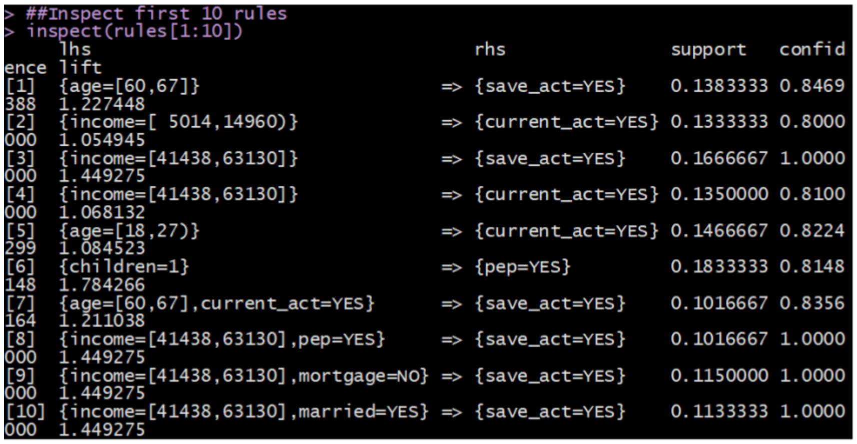
<u>*In R:*</u> Association rule discovery was worked out in a similar method as what was done in Weka. We first started out by trying to find the rule support. This is the proportion of data rows where both the left and right sides of the rule meet the condition. In step 1 (Figure 2.4), we looked at the proportion of data with an age between 60 and 67 and save_act=Yes. We then ran the apriori algorithm with the default argument values and saved the rules as a variable. The output

of the algorithm shows the same parameters as the Weka output (input parameters, number of items, number of data rows, and number of rules generated). We also see that 101 rules are generated. For the sake of space, the output for these rules are shown in appendix A at the end of this document. Figure 2.5 shows the first 10 rules.



```
> ######## Association Rule Discovery#########
>
> ##Step1: find out how many LHS rows meet RHS condition
> summary(subset.data.frame(bankdata, bankdata$save_act=="YES"))
      age          sex              region              income    married   chi
ldren  car       save_act
 [18,27):64   FEMALE:206   INNER_CITY:173   [ 5014,14960): 64   NO :137   0:1
74    NO :205   NO :  0
 [27,36):64   MALE  :208   RURAL     : 70   [14960,20263): 59   YES:277   1:
95    YES:209   YES:414
 [36,43):52                SUBURBAN  : 43   [20263,24947): 58             2:
99
 [43,50):76                TOWN      :128   [24947,31207): 62             3:
46
 [50,60):75                                 [31207,41438): 71
 [60,67]:83                                 [41438,63130]:100
 current_act mortgage   pep
 NO : 95     NO :270   NO :235
 YES:319     YES:144   YES:179




>
>
> #run apriori method with default arguments
> rules<-apriori(bankdata)
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen max
len target   ext
        0.8    0.1    1 none FALSE            TRUE       5     0.1      1
10  rules FALSE

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

Absolute minimum support count: 60

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[34 item(s), 600 transaction(s)] done [0.00s].
sorting and recoding items ... [34 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 done [0.00s].
writing ... [101 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules
set of 101 rules
```

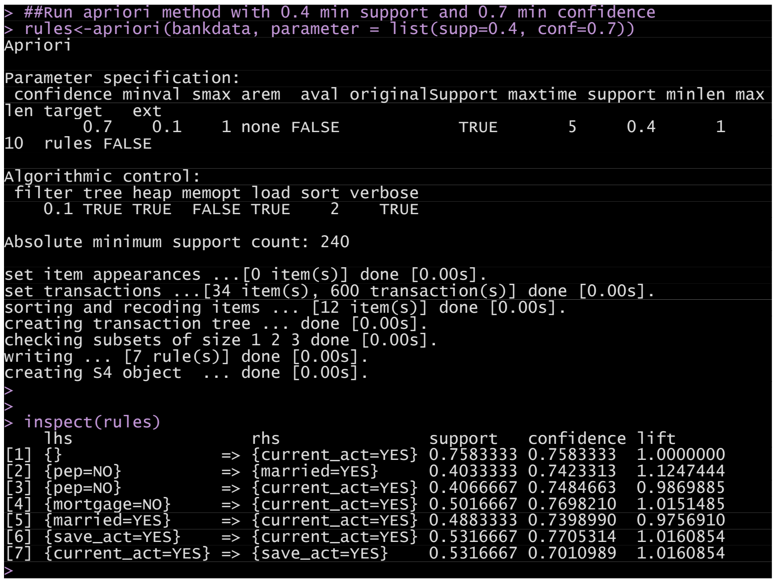
Figure 2.4



```
> ##Inspect first 10 rules
> inspect(rules[1:10])
    lhs                                   rhs               support   confid
ence lift
[1]  {age=[60,67]}                     => {save_act=YES}    0.1383333 0.8469
388  1.227448
[2]  {income=[ 5014,14960)}            => {current_act=YES} 0.1333333 0.8000
000  1.054945
[3]  {income=[41438,63130]}            => {save_act=YES}    0.1666667 1.0000
000  1.449275
[4]  {income=[41438,63130]}            => {current_act=YES} 0.1350000 0.8100
000  1.068132
[5]  {age=[18,27)}                     => {current_act=YES} 0.1466667 0.8224
299  1.084523
[6]  {children=1}                      => {pep=YES}         0.1833333 0.8148
148  1.784266
[7]  {age=[60,67],current_act=YES}     => {save_act=YES}    0.1016667 0.8356
164  1.211038
[8]  {income=[41438,63130],pep=YES}    => {save_act=YES}    0.1016667 1.0000
000  1.449275
[9]  {income=[41438,63130],mortgage=NO} => {save_act=YES}   0.1150000 1.0000
000  1.449275
[10] {income=[41438,63130],married=YES} => {save_act=YES}   0.1133333 1.0000
000  1.449275
```

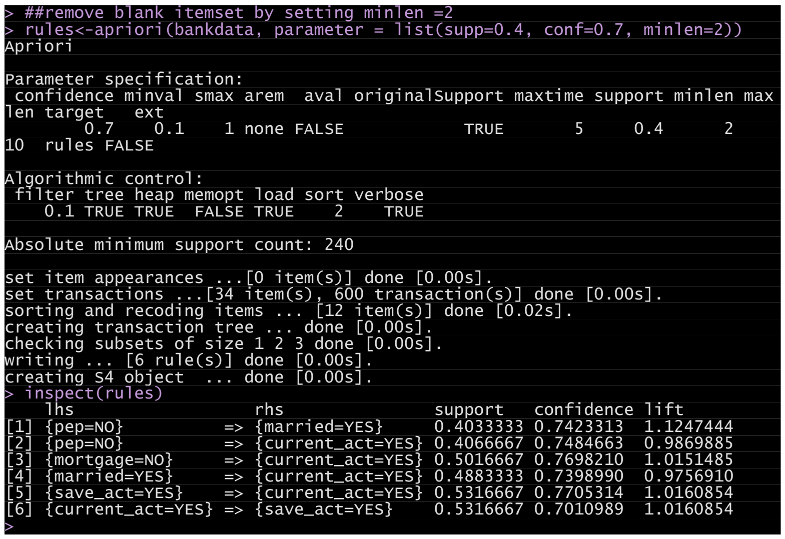
Figure 2.5

Next, we chose to generate rules with specified minimum support and minimum confidence. We chose a support value of 0.4 and with a confidence of 0.7 (Figure 2.6). The method generated 7 rules. Looking at the rules, since the first one has a blank left item set. This means that the lift metric for that rule is one, and the left- and right-hand sides are independent. Therefore, we need to eliminate the blank dataset. This is done in Figure 2.7 using the minlen parameter. The rules are generated again, and the function comes back with 6 generated rules. We kept the confidence at 0.7 and the support at 0.4.



```
> ##Run apriori method with 0.4 min support and 0.7 min confidence
> rules<-apriori(bankdata, parameter = list(supp=0.4, conf=0.7))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen max
len target   ext
        0.7    0.1    1 none FALSE            TRUE       5     0.4      1
10  rules FALSE

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

Absolute minimum support count: 240

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[34 item(s), 600 transaction(s)] done [0.00s].
sorting and recoding items ... [12 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [7 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>
>
> inspect(rules)
    lhs                  rhs               support   confidence lift
[1] {}                => {current_act=YES} 0.7583333 0.7583333  1.0000000
[2] {pep=NO}          => {married=YES}     0.4033333 0.7423313  1.1247444
[3] {pep=NO}          => {current_act=YES} 0.4066667 0.7484663  0.9869885
[4] {mortgage=NO}     => {current_act=YES} 0.5016667 0.7698210  1.0151485
[5] {married=YES}     => {current_act=YES} 0.4883333 0.7398990  0.9756910
[6] {save_act=YES}    => {current_act=YES} 0.5316667 0.7705314  1.0160854
[7] {current_act=YES} => {save_act=YES}    0.5316667 0.7010989  1.0160854
>
```

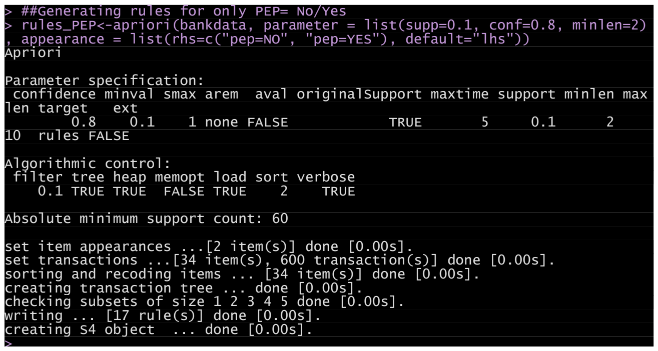
Figure 2.6



```
> ##remove blank itemset by setting minlen =2
> rules<-apriori(bankdata, parameter = list(supp=0.4, conf=0.7, minlen=2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen max
len target   ext
        0.7    0.1    1 none FALSE            TRUE       5     0.4      2
10  rules FALSE

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

Absolute minimum support count: 240

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[34 item(s), 600 transaction(s)] done [0.00s].
sorting and recoding items ... [12 item(s)] done [0.02s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [6 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> inspect(rules)
    lhs                   rhs               support   confidence lift
[1] {pep=NO}           => {married=YES}     0.4033333 0.7423313  1.1247444
[2] {pep=NO}           => {current_act=YES} 0.4066667 0.7484663  0.9869885
[3] {mortgage=NO}      => {current_act=YES} 0.5016667 0.7698210  1.0151485
[4] {married=YES}      => {current_act=YES} 0.4883333 0.7398990  0.9756910
[5] {save_act=YES}     => {current_act=YES} 0.5316667 0.7705314  1.0160854
[6] {current_act=YES} => {save_act=YES}     0.5316667 0.7010989  1.0160854
>
```

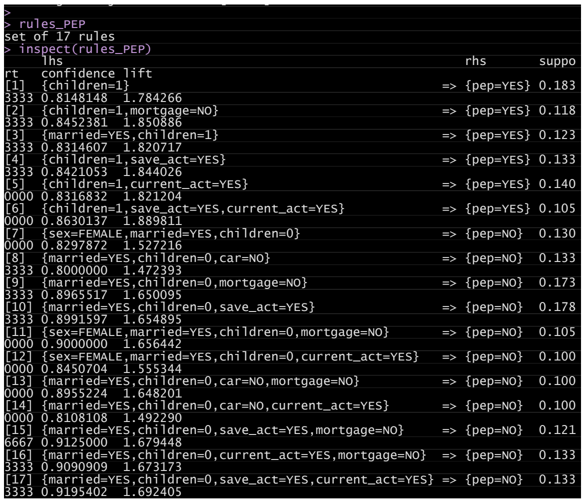
Figure 2.7

Finally, we chose to generate rules with only pep=NO or pep=YES on the right hand side. We used filters to generate rules with only the specified item sets. This method generated 17 Rules, with the minimum allowed rule length is 2 and the maximum length is 10. (Figure 2.8, 2.9). Finally, we sorted through the rules by the descending lift metric (Figure 2.10). This was done again for the Confidence metric to show rules generated when pep=No.The top five rules for when pep=no is pasted in results.



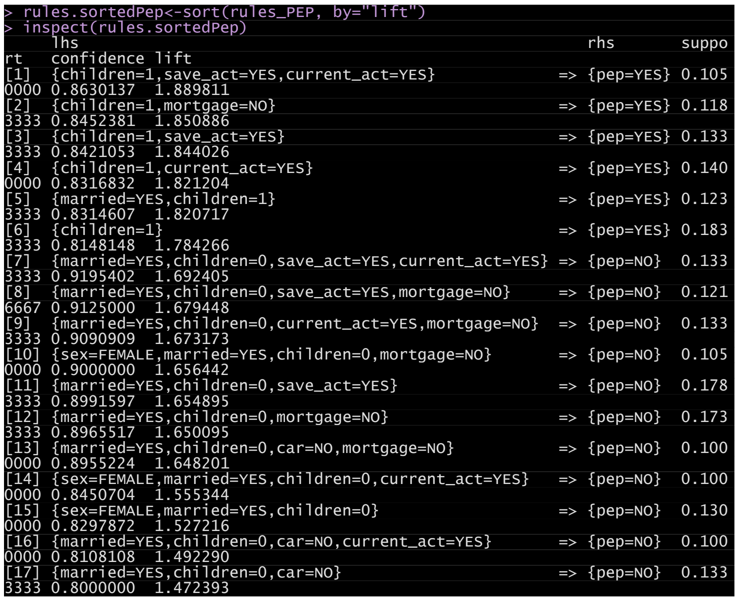
```
> ##Generating rules for only PEP= No/Yes
> rules_PEP<-apriori(bankdata, parameter = list(supp=0.1, conf=0.8, minlen=2)
, appearance = list(rhs=c("pep=NO", "pep=YES"), default="lhs"))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen max
len target   ext
        0.8    0.1    1 none FALSE            TRUE       5     0.1      2
10  rules FALSE

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

Absolute minimum support count: 60

set item appearances ...[2 item(s)] done [0.00s].
set transactions ...[34 item(s), 600 transaction(s)] done [0.00s].
sorting and recoding items ... [34 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 done [0.00s].
writing ... [17 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>
```

Figure 2.8



```
>
> rules_PEP
set of 17 rules
> inspect(rules_PEP)
     lhs                                                 rhs       suppo
rt   confidence lift
[1]  {children=1}                                     => {pep=YES} 0.183
3333 0.8148148  1.784266
[2]  {children=1,mortgage=NO}                         => {pep=YES} 0.118
3333 0.8452381  1.850886
[3]  {married=YES,children=1}                         => {pep=YES} 0.123
3333 0.8314607  1.820717
[4]  {children=1,save_act=YES}                        => {pep=YES} 0.133
3333 0.8421053  1.844026
[5]  {children=1,current_act=YES}                     => {pep=YES} 0.140
0000 0.8316832  1.821204
[6]  {children=1,save_act=YES,current_act=YES}        => {pep=YES} 0.105
0000 0.8630137  1.889811
[7]  {sex=FEMALE,married=YES,children=0}              => {pep=NO}  0.130
0000 0.8297872  1.527216
[8]  {married=YES,children=0,car=NO}                  => {pep=NO}  0.133
3333 0.8000000  1.472393
[9]  {married=YES,children=0,mortgage=NO}             => {pep=NO}  0.173
3333 0.8965517  1.650095
[10] {married=YES,children=0,save_act=YES}            => {pep=NO}  0.178
3333 0.8991597  1.654895
[11] {sex=FEMALE,married=YES,children=0,mortgage=NO}  => {pep=NO}  0.105
0000 0.9000000  1.656442
[12] {sex=FEMALE,married=YES,children=0,current_act=YES}   => {pep=NO}  0.100
0000 0.8450704  1.555344
[13] {married=YES,children=0,car=NO,mortgage=NO}      => {pep=NO}  0.100
0000 0.8955224  1.648201
[14] {married=YES,children=0,car=NO,current_act=YES}  => {pep=NO}  0.100
0000 0.8108108  1.492290
[15] {married=YES,children=0,save_act=YES,mortgage=NO}     => {pep=NO}  0.121
6667 0.9125000  1.679448
[16] {married=YES,children=0,current_act=YES,mortgage=NO}  => {pep=NO}  0.133
3333 0.9090909  1.673173
[17] {married=YES,children=0,save_act=YES,current_act=YES} => {pep=NO}  0.133
3333 0.9195402  1.692405
```

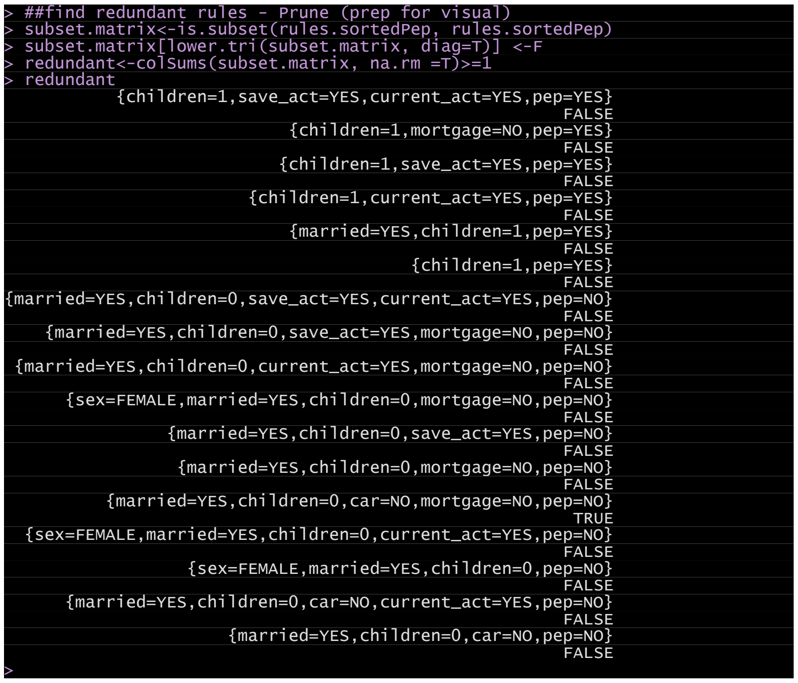
Figure 2.9



```
> rules.sortedPep<-sort(rules_PEP, by="lift")
> inspect(rules.sortedPep)
     lhs                                                      rhs       suppo
rt   confidence lift
[1]  {children=1,save_act=YES,current_act=YES}             => {pep=YES} 0.105
0000 0.8630137  1.889811
[2]  {children=1,mortgage=NO}                              => {pep=YES} 0.118
3333 0.8452381  1.850886
[3]  {children=1,save_act=YES}                             => {pep=YES} 0.133
3333 0.8421053  1.844026
[4]  {children=1,current_act=YES}                          => {pep=YES} 0.140
0000 0.8316832  1.821204
[5]  {married=YES,children=1}                              => {pep=YES} 0.123
3333 0.8314607  1.820717
[6]  {children=1}                                          => {pep=YES} 0.183
3333 0.8148148  1.784266
[7]  {married=YES,children=0,save_act=YES,current_act=YES} => {pep=NO}  0.133
3333 0.9195402  1.692405
[8]  {married=YES,children=0,save_act=YES,mortgage=NO}     => {pep=NO}  0.121
6667 0.9125000  1.679448
[9]  {married=YES,children=0,current_act=YES,mortgage=NO}  => {pep=NO}  0.133
3333 0.9090909  1.673173
[10] {sex=FEMALE,married=YES,children=0,mortgage=NO}       => {pep=NO}  0.105
0000 0.9000000  1.656442
[11] {married=YES,children=0,save_act=YES}                 => {pep=NO}  0.178
3333 0.8991597  1.654895
[12] {married=YES,children=0,mortgage=NO}                  => {pep=NO}  0.173
3333 0.8965517  1.650095
[13] {married=YES,children=0,car=NO,mortgage=NO}           => {pep=NO}  0.100
0000 0.8955224  1.648201
[14] {sex=FEMALE,married=YES,children=0,current_act=YES}   => {pep=NO}  0.100
0000 0.8450704  1.555344
[15] {sex=FEMALE,married=YES,children=0}                   => {pep=NO}  0.130
0000 0.8297872  1.527216
[16] {married=YES,children=0,car=NO,current_act=YES}       => {pep=NO}  0.100
0000 0.8108108  1.492290
[17] {married=YES,children=0,car=NO}                       => {pep=NO}  0.133
3333 0.8000000  1.472393
```

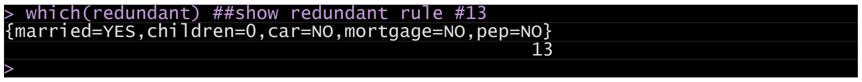
Figure 2.10

In Figure 2.10, we see that some itemset on the left-hand side contain three items while others contain 1 item, etc. Therefore, we prepare for pruning the redundant rules. In Figure 2.11 we build a matrix where the first column is the itemset from the left and right side of the rules. The headings of the other columns are the rule numbers. Each intersection is either True or False. We can also use the which(redundant) command, like in Figure 2.12, to see that the redundant rule is number 13. Finally, in Figure 2.13, we remove the redundant rule and store the final pruned rules. We can see that we have 16 rules that remain.



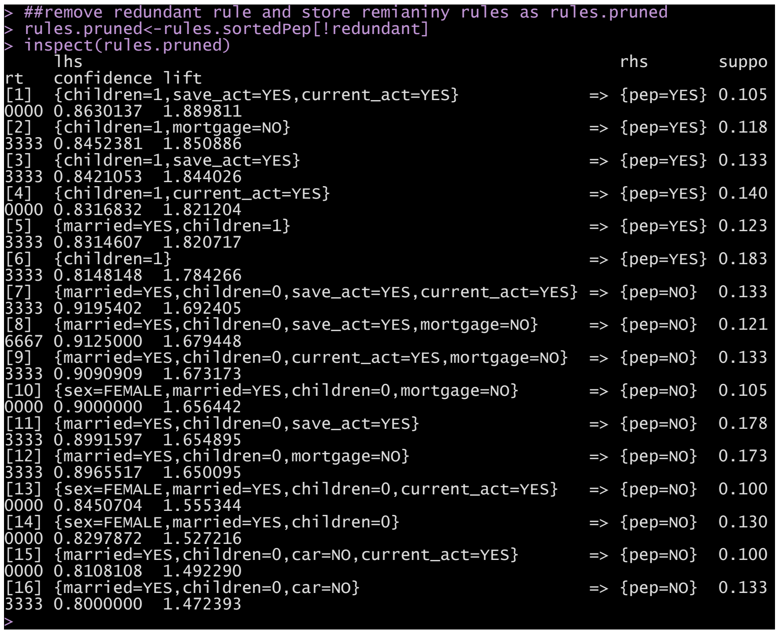
```
> ##find redundant rules - Prune (prep for visual)
> subset.matrix<-is.subset(rules.sortedPep, rules.sortedPep)
> subset.matrix[lower.tri(subset.matrix, diag=T)] <-F
> redundant<-colSums(subset.matrix, na.rm =T)>=1
> redundant
           {children=1,save_act=YES,current_act=YES,pep=YES}
                                                       FALSE
                            {children=1,mortgage=NO,pep=YES}
                                                       FALSE
                           {children=1,save_act=YES,pep=YES}
                                                       FALSE
                        {children=1,current_act=YES,pep=YES}
                                                       FALSE
                            {married=YES,children=1,pep=YES}
                                                       FALSE
                                        {children=1,pep=YES}
                                                       FALSE
{married=YES,children=0,save_act=YES,current_act=YES,pep=NO}
                                                       FALSE
    {married=YES,children=0,save_act=YES,mortgage=NO,pep=NO}
                                                       FALSE
 {married=YES,children=0,current_act=YES,mortgage=NO,pep=NO}
                                                       FALSE
      {sex=FEMALE,married=YES,children=0,mortgage=NO,pep=NO}
                                                       FALSE
                 {married=YES,children=0,save_act=YES,pep=NO}
                                                       FALSE
                  {married=YES,children=0,mortgage=NO,pep=NO}
                                                       FALSE
           {married=YES,children=0,car=NO,mortgage=NO,pep=NO}
                                                        TRUE
  {sex=FEMALE,married=YES,children=0,current_act=YES,pep=NO}
                                                       FALSE
                   {sex=FEMALE,married=YES,children=0,pep=NO}
                                                       FALSE
      {married=YES,children=0,car=NO,current_act=YES,pep=NO}
                                                       FALSE
                       {married=YES,children=0,car=NO,pep=NO}
                                                       FALSE
>
```

Figure 2.11



```
> which(redundant) ##show redundant rule #13
{married=YES,children=0,car=NO,mortgage=NO,pep=NO}
                                                13
>
```

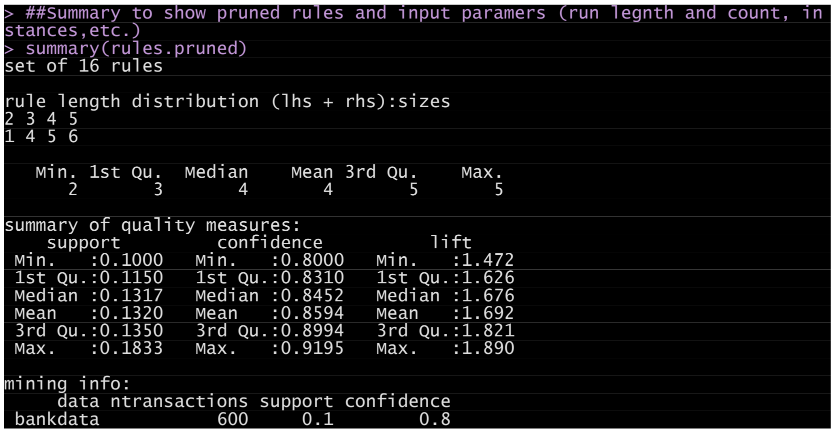
Figure 2.12



```
> ##remove redundant rule and store remianiny rules as rules.pruned
> rules.pruned<-rules.sortedPep[!redundant]
> inspect(rules.pruned)
     lhs                                                       rhs       suppo
rt   confidence lift
[1]  {children=1,save_act=YES,current_act=YES}             => {pep=YES} 0.105
0000 0.8630137  1.889811
[2]  {children=1,mortgage=NO}                              => {pep=YES} 0.118
3333 0.8452381  1.850886
[3]  {children=1,save_act=YES}                             => {pep=YES} 0.133
3333 0.8421053  1.844026
[4]  {children=1,current_act=YES}                          => {pep=YES} 0.140
0000 0.8316832  1.821204
[5]  {married=YES,children=1}                              => {pep=YES} 0.123
3333 0.8314607  1.820717
[6]  {children=1}                                          => {pep=YES} 0.183
3333 0.8148148  1.784266
[7]  {married=YES,children=0,save_act=YES,current_act=YES} => {pep=NO}  0.133
3333 0.9195402  1.692405
[8]  {married=YES,children=0,save_act=YES,mortgage=NO}     => {pep=NO}  0.121
6667 0.9125000  1.679448
[9]  {married=YES,children=0,current_act=YES,mortgage=NO}  => {pep=NO}  0.133
3333 0.9090909  1.673173
[10] {sex=FEMALE,married=YES,children=0,mortgage=NO}       => {pep=NO}  0.105
0000 0.9000000  1.656442
[11] {married=YES,children=0,save_act=YES}                 => {pep=NO}  0.178
3333 0.8991597  1.654895
[12] {married=YES,children=0,mortgage=NO}                  => {pep=NO}  0.173
3333 0.8965517  1.650095
[13] {sex=FEMALE,married=YES,children=0,current_act=YES}   => {pep=NO}  0.100
0000 0.8450704  1.555344
[14] {sex=FEMALE,married=YES,children=0}                   => {pep=NO}  0.130
0000 0.8297872  1.527216
[15] {married=YES,children=0,car=NO,current_act=YES}       => {pep=NO}  0.100
0000 0.8108108  1.492290
[16] {married=YES,children=0,car=NO}                       => {pep=NO}  0.133
3333 0.8000000  1.472393
>
```

Figure 2.13

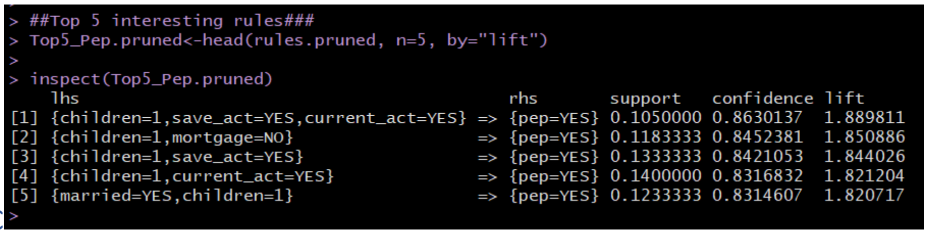
Finally, in Figure 2.14, we run a summary of the pruned rules. This is done in the end to show the statistical metrics and input parameters. We can see the method generated 16 rules totals. The rule length and count are listed on the left side (2 3 4 5 and 1 4 5 6). Additionally, the summary of quality measures includes the minimum, 1st quartile, mean, median, 3rd quartile, and max value for the three measures: support, confidence, and lift. The mining inf shows the dataset name, data rows count, minimum support and max confidence of the Apriori method parameters.



```
> ##Summary to show pruned rules and input paramers (run legnth and count, in
stances,etc.)
> summary(rules.pruned)
set of 16 rules

rule length distribution (lhs + rhs):sizes
2 3 4 5
1 4 5 6

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      2       3       4       4       5       5

summary of quality measures:
    support          confidence          lift
 Min.   :0.1000   Min.   :0.8000   Min.   :1.472
 1st Qu.:0.1150   1st Qu.:0.8310   1st Qu.:1.626
 Median :0.1317   Median :0.8452   Median :1.676
 Mean   :0.1320   Mean   :0.8594   Mean   :1.692
 3rd Qu.:0.1350   3rd Qu.:0.8994   3rd Qu.:1.821
 Max.   :0.1833   Max.   :0.9195   Max.   :1.890

mining info:
     data ntransactions support confidence
 bankdata           600     0.1        0.8
```

Figure 2.14

Finally, in Figure 2.15, we generate the top 5 interesting rules from the 16 generated above.



```
> ##Top 5 interesting rules###
> Top5_Pep.pruned<-head(rules.pruned, n=5, by="lift")
>
> inspect(Top5_Pep.pruned)
    lhs                                           rhs       support   confidence lift
[1] {children=1,save_act=YES,current_act=YES} => {pep=YES} 0.1050000 0.8630137  1.889811
[2] {children=1,mortgage=NO}                  => {pep=YES} 0.1183333 0.8452381  1.850886
[3] {children=1,save_act=YES}                 => {pep=YES} 0.1333333 0.8421053  1.844026
[4] {children=1,current_act=YES}              => {pep=YES} 0.1400000 0.8316832  1.821204
[5] {married=YES,children=1}                  => {pep=YES} 0.1233333 0.8314607  1.820717
>
```

C

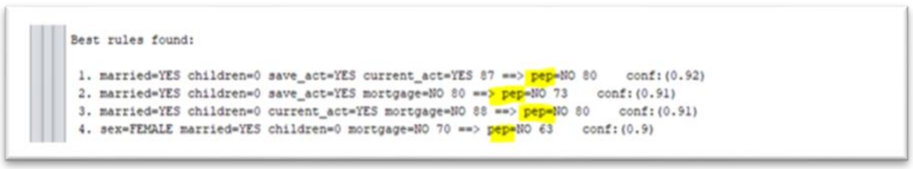
Figure 2.15

Results:

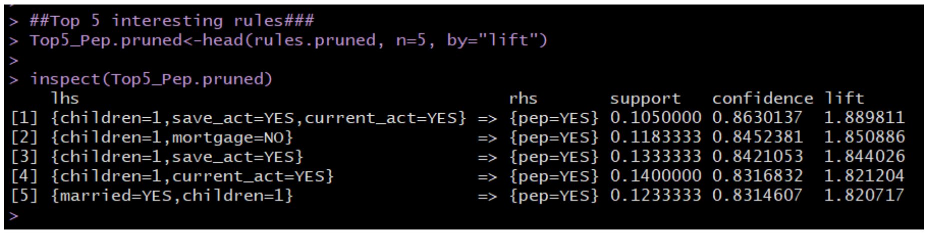
Top 5 Most Interesting Rules & 3 items Listed Above For Each Rule:

We will us use the Top 5 Most Interesting Rules produced by the R code and Weka. The Weka was used to generate when pep=NO, which generated 4 results. The R code was used for when pep=YES, and generated 16 results (Figure 2.12), but we have taken the top five of the sixteen for simplicity. In addition, for standardization, we generated the R code for when pep=No. It is similar to the Weka rules, but it was included for completeness. For the final five most interesting, we took 3 from pep=YES and 2 from pep=NO.

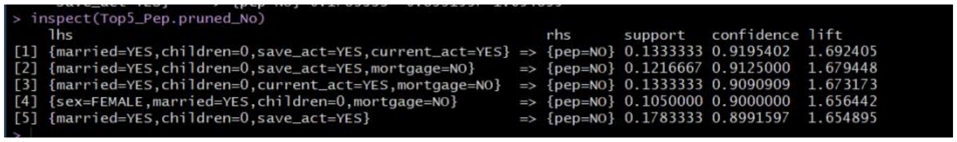
Parin Patel



```
Best rules found:

 1. married=YES children=0 save_act=YES current_act=YES 87 ==> pep=NO 80    conf:(0.92)
 2. married=YES children=0 save_act=YES mortgage=NO 80 ==> pep=NO 73    conf:(0.91)
 3. married=YES children=0 current_act=YES mortgage=NO 88 ==> pep=NO 80    conf:(0.91)
 4. sex=FEMALE married=YES children=0 mortgage=NO 70 ==> pep=NO 63    conf:(0.9)
```



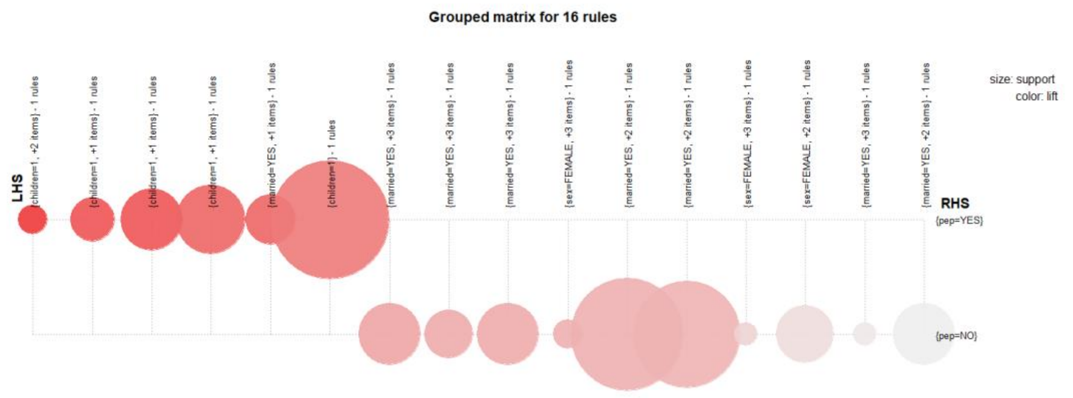
```
> ##Top 5 interesting rules###
> Top5_Pep.pruned<-head(rules.pruned, n=5, by="lift")
>
> inspect(Top5_Pep.pruned)
    lhs                                         rhs         support   confidence lift
[1] {children=1,save_act=YES,current_act=YES} => {pep=YES} 0.1050000 0.8630137  1.889811
[2] {children=1,mortgage=NO}                  => {pep=YES} 0.1183333 0.8452381  1.850886
[3] {children=1,save_act=YES}                 => {pep=YES} 0.1333333 0.8421053  1.844026
[4] {children=1,current_act=YES}              => {pep=YES} 0.1400000 0.8316832  1.821204
[5] {married=YES,children=1}                  => {pep=YES} 0.1233333 0.8314607  1.820717
>
```



```
> inspect(Top5_Pep.pruned_No)
    lhs                                                     rhs        support   confidence lift
[1] {married=YES,children=0,save_act=YES,current_act=YES} => {pep=NO} 0.1333333 0.9195402  1.692405
[2] {married=YES,children=0,save_act=YES,mortgage=NO}     => {pep=NO} 0.1216667 0.9125000  1.679448
[3] {married=YES,children=0,current_act=YES,mortgage=NO}  => {pep=NO} 0.1333333 0.9090909  1.673173
[4] {sex=FEMALE,married=YES,children=0,mortgage=NO}       => {pep=NO} 0.1050000 0.9000000  1.656442
[5] {married=YES,children=0,save_act=YES}                 => {pep=NO} 0.1783333 0.8991597  1.654895
```

1. Customers who are married with no children, have a savings account and a current account are not likely to buy PEP.
   a. In this first rule, the support is 0.133, and the confidence is 0.920.
2. Customers who are married, with no children who have a savings account, but no mortgage are not likely to buy PEP.
   a. In this rule, the support is 0.122, and the confidence is 0.912.
3. Customers who have one child, a savings account, and a current account ARE likely to buy a PEP investment account.
   a. In this rule, the support is 0.105 and the confidence is 0.863.
4. Customers who are have one child and a savings account ARE likely to buy PEP
5. Customers who have one child and no mortgage are likely to buy PEP.

From the rules, it can be concluded that customers who have one child are a very important target demographic because they will likely buy PEP. In addition, whether a customer has a mortgage account does not play a major factor in if they will open a PEP account. The below grouped matrix shows the spread of rules and their relation to whether pep=No or Yes.

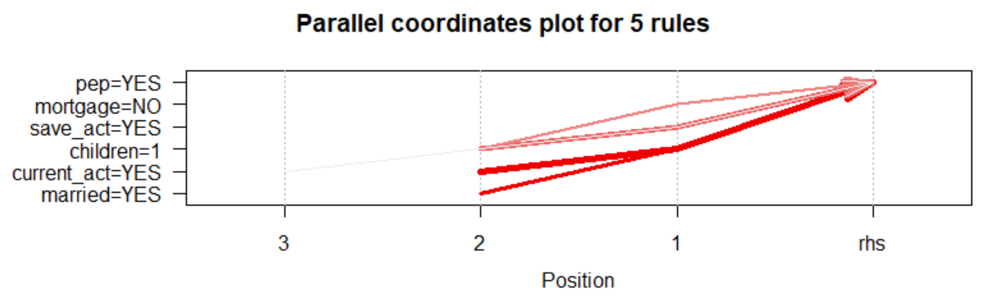


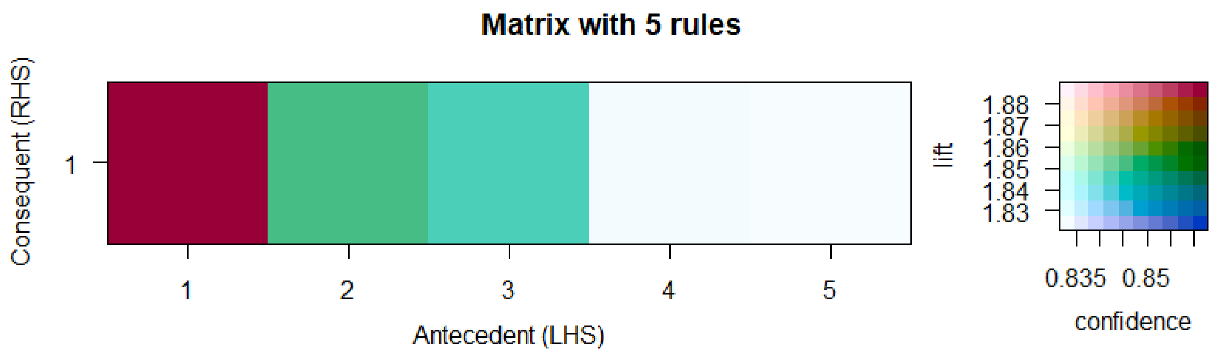
Discussion of One Rule:

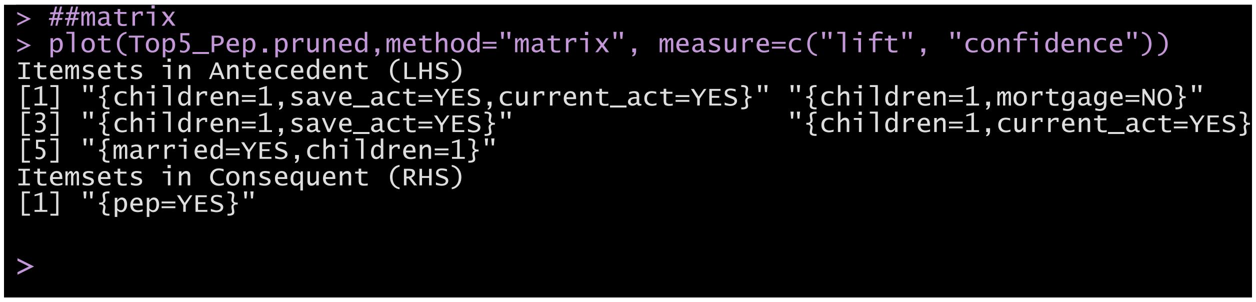
In the third rule, that states customers who have one child, a savings account, and a current account are likely to open a PEP Investment account, the support is 0.105. Considering that the dataset contains 600 instances, this means that customers that fit this demographic are represented in 63 transactions (600*0.105). In addition, the confidence is 0.836, this means that this demographic of customers has a probability of 0.836 of opening a PEP investment account. The lift is 1.8898, this references that the probability for customers with one child, a savings and current account, that open a PEP account is improved by 1.8898.

Conclusion:

When talking to my client, who knows very little about data mining, I would him them to look through the graphics and visuals to better understand which of their customers are buying PEP investment accounts. Looking at the graphics below you can see that the those who are married, with one child, a savings account, no mortgage, and a current account are more likely to open PEP. In addition, the Matrix With 5 Rules visual explains how confident we are with the first rule, which is that the those who are married, with one child, a savings account, no mortgage, and a current account are more likely to open PEP.





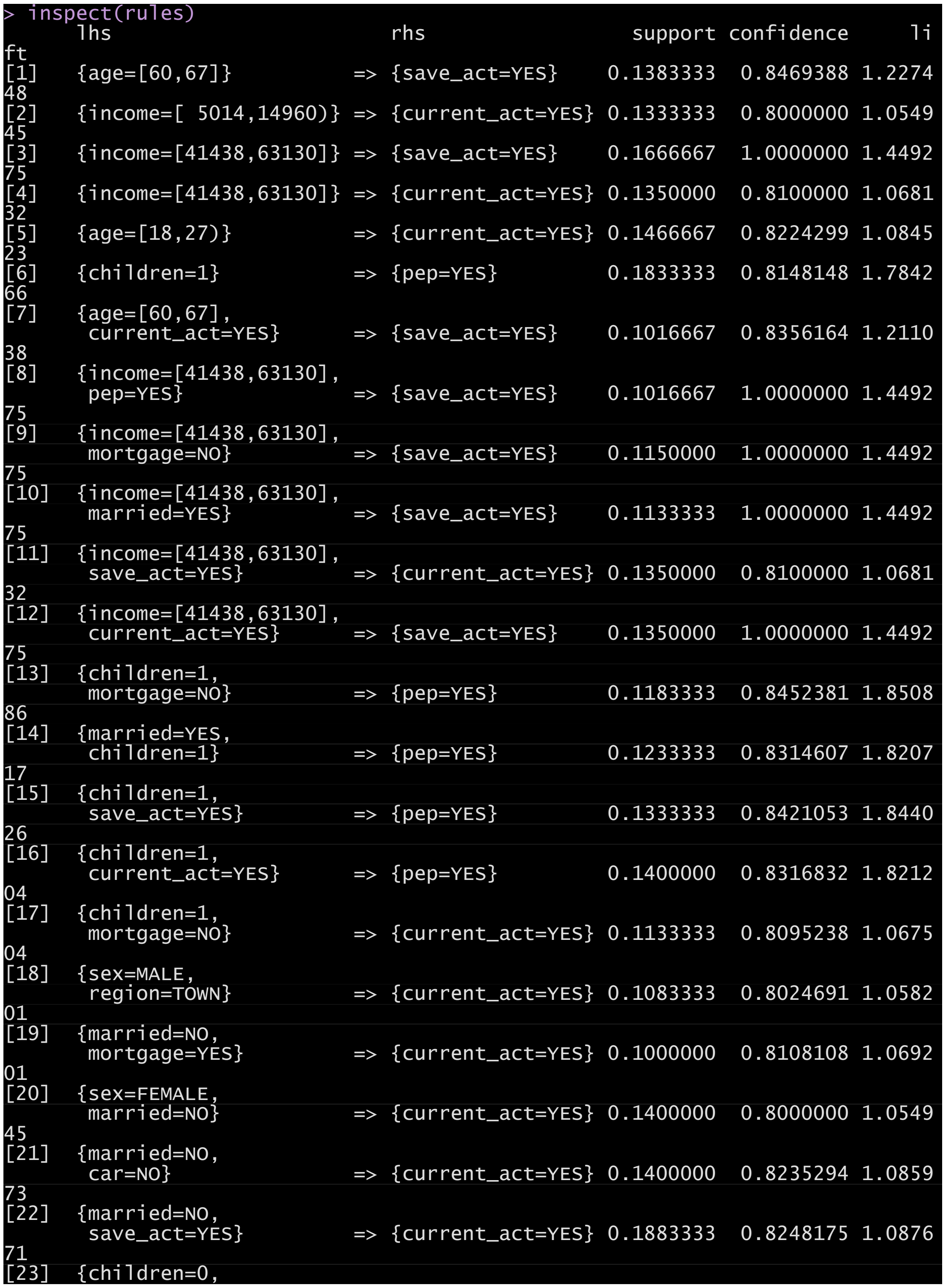


```
> ##matrix
> plot(Top5_Pep.pruned,method="matrix", measure=c("lift", "confidence"))
Itemsets in Antecedent (LHS)
[1] "{children=1,save_act=YES,current_act=YES}" "{children=1,mortgage=NO}"
[3] "{children=1,save_act=YES}"                 "{children=1,current_act=YES}
[5] "{married=YES,children=1}"
Itemsets in Consequent (RHS)
[1] "{pep=YES}"

>
```

Appendix A:

All 101 Initial Rules

Parin Patel



```
> inspect(rules)
      lhs                          rhs                  support confidence     li
ft
[1]   {age=[60,67]}             => {save_act=YES}    0.1383333  0.8469388 1.2274
48
[2]   {income=[ 5014,14960)} => {current_act=YES} 0.1333333  0.8000000 1.0549
45
[3]   {income=[41438,63130]} => {save_act=YES}    0.1666667  1.0000000 1.4492
75
[4]   {income=[41438,63130]} => {current_act=YES} 0.1350000  0.8100000 1.0681
32
[5]   {age=[18,27)}             => {current_act=YES} 0.1466667  0.8224299 1.0845
23
[6]   {children=1}              => {pep=YES}         0.1833333  0.8148148 1.7842
66
[7]   {age=[60,67],
       current_act=YES}       => {save_act=YES}    0.1016667  0.8356164 1.2110
38
[8]   {income=[41438,63130],
       pep=YES}               => {save_act=YES}    0.1016667  1.0000000 1.4492
75
[9]   {income=[41438,63130],
       mortgage=NO}           => {save_act=YES}    0.1150000  1.0000000 1.4492
75
[10]  {income=[41438,63130],
       married=YES}           => {save_act=YES}    0.1133333  1.0000000 1.4492
75
[11]  {income=[41438,63130],
       save_act=YES}          => {current_act=YES} 0.1350000  0.8100000 1.0681
32
[12]  {income=[41438,63130],
       current_act=YES}       => {save_act=YES}    0.1350000  1.0000000 1.4492
75
[13]  {children=1,
       mortgage=NO}           => {pep=YES}         0.1183333  0.8452381 1.8508
86
[14]  {married=YES,
       children=1}            => {pep=YES}         0.1233333  0.8314607 1.8207
17
[15]  {children=1,
       save_act=YES}          => {pep=YES}         0.1333333  0.8421053 1.8440
26
[16]  {children=1,
       current_act=YES}       => {pep=YES}         0.1400000  0.8316832 1.8212
04
[17]  {children=1,
       mortgage=NO}           => {current_act=YES} 0.1133333  0.8095238 1.0675
04
[18]  {sex=MALE,
       region=TOWN}           => {current_act=YES} 0.1083333  0.8024691 1.0582
01
[19]  {married=NO,
       mortgage=YES}          => {current_act=YES} 0.1000000  0.8108108 1.0692
01
[20]  {sex=FEMALE,
       married=NO}            => {current_act=YES} 0.1400000  0.8000000 1.0549
45
[21]  {married=NO,
       car=NO}                => {current_act=YES} 0.1400000  0.8235294 1.0859
73
[22]  {married=NO,
       save_act=YES}          => {current_act=YES} 0.1883333  0.8248175 1.0876
71
[23]  {children=0,
```