AC / Machine Learning
Rita P. Ribeiro
Nuno Moniz

2021/2022
DCC - FCUP
DEI - FEUP

1 Hands On: Association Rules

1.1 Association Rules

1. Load the packages `arules`, `arulesViz` and the dataset `Groceries` from the package `arules` which contains 1 month of real-world point-of-sale transaction data from a typical local grocery.

```
library(arules)
library(arulesViz)
library(dplyr)
data(Groceries)
```

(a) Type `Groceries` on the R prompt. What does it return? Use the function `class` to inspect the type of data set.

```
Groceries

## transactions in sparse format with
##  9835 transactions (rows) and
##  169 items (columns)

class(Groceries)

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

(b) Use the function `summary` to get more information on the data set.

```
summary(Groceries)

## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55
##   16   17   18   19   20   21   22   23   24   26   27   28   29   32
##   46   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##        labels  level2           level1
## 1 frankfurter sausage meat and sausage
## 2     sausage sausage meat and sausage
## 3  liver loaf sausage meat and sausage
```

(c) Use the function `size` on the data set. What information does it return?

```
head(size(Groceries))

## [1] 4 3 1 4 4 5
```

(d) Use the function `inspect` to see the first five transactions.

```
inspect(Groceries[1:5])

##     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}
```

(e) Are there any duplicated transactions? Use the function `unique` or `duplicated`.

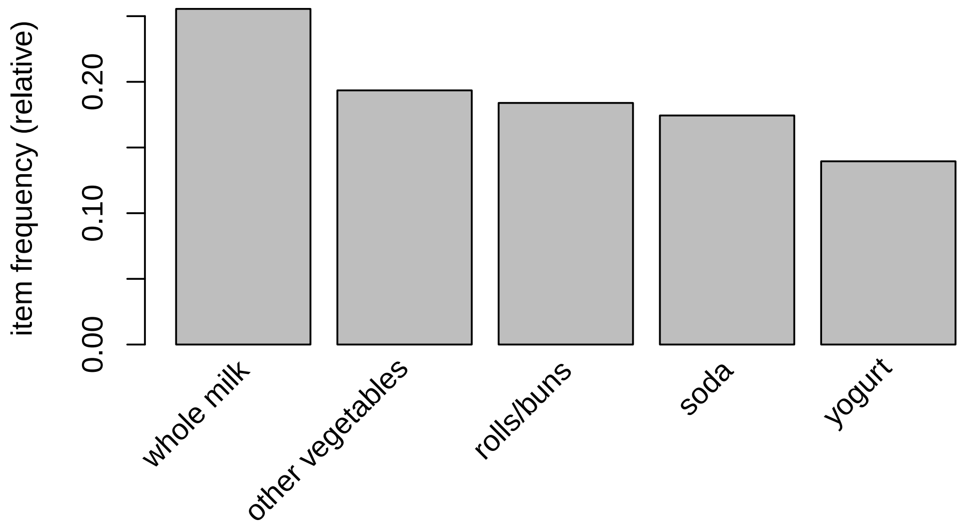
```
length(which(duplicated(Groceries)))

## [1] 2824
```

(f) Use the function `itemFrequency` to see the relative frequency of each item.

```
head(itemFrequency(Groceries))

##       frankfurter           sausage        liver loaf               ham
##       0.058973055       0.093950178       0.005083884       0.026029487
##              meat finished products
##       0.025826131       0.006507372
```

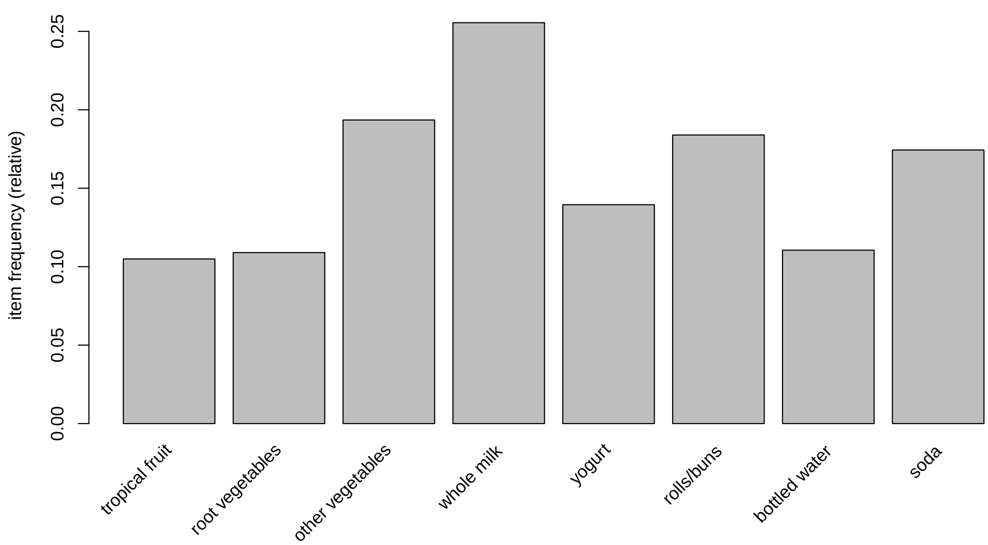
(g) Using the function `itemFrequencyPlot`, plot the top 5 more frequent items.

```
itemFrequencyPlot(Groceries, topN = 5)
```



(h) Using the same function `itemFrequencyPlot`, plot the items that have a support value of at least 0.1. How many are there?

```
itemFrequencyPlot(Groceries, support = 0.1)
```



(i) Using function `apriori`, and without generating any rules, obtain the frequent itemsets for a minimum support of 0.01. What is the class of the object returned? How many frequent itemsets were found?

```
fsets <- apriori(Groceries, parameter = list(supp = 0.01, target = "frequent itemsets"))

## Apriori
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##          NA    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen            target   ext
##      10 frequent itemsets FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [333 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
class(fsets)
```

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

(j) Inspect the 5 most frequent itemsets. What's their size?

```
inspect(sort(fsets)[1:5])
```

```
##     items              support   count
## [1] {whole milk}       0.2555160 2513
## [2] {other vegetables} 0.1934926 1903
## [3] {rolls/buns}       0.1839349 1809
## [4] {soda}             0.1743772 1715
## [5] {yogurt}           0.1395018 1372
```

(k) From the frequent itemsets obtained, select the subset of closed frequent itemsets and the subset of maximal frequent itemsets. What can you conclude?

```
fsets[is.closed(fsets)]
```

```
## set of 333 itemsets
```

```
fsets[is.maximal(fsets)]
```

```
## set of 243 itemsets
```

(l) Use the function `apriori` to generate association rules from the Groceries data set. What is the class of the returned object? How many rules were generated?

```
rules <- apriori(Groceries)
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 983
##
## set item appearances ...[0 item(s)] done [0.00s].
```

```
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [8 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

(m) Change the values of minimum support and minimum confidence and see how does that affect the number of rules generated.

```r
rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

(n) Obtain the association rules with minsup=0.01 and minconf=0.25. Using the functions `summary`, `quality`, `plot` and `inspect` acquire more information on the generated rules.

```r
rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.25))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [171 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
summary(rules)
```

```
## set of 171 rules
##
## rule length distribution (lhs + rhs):sizes
##  1  2  3
##  1 96 74
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.427   3.000   3.000
##
## summary of quality measures:
```

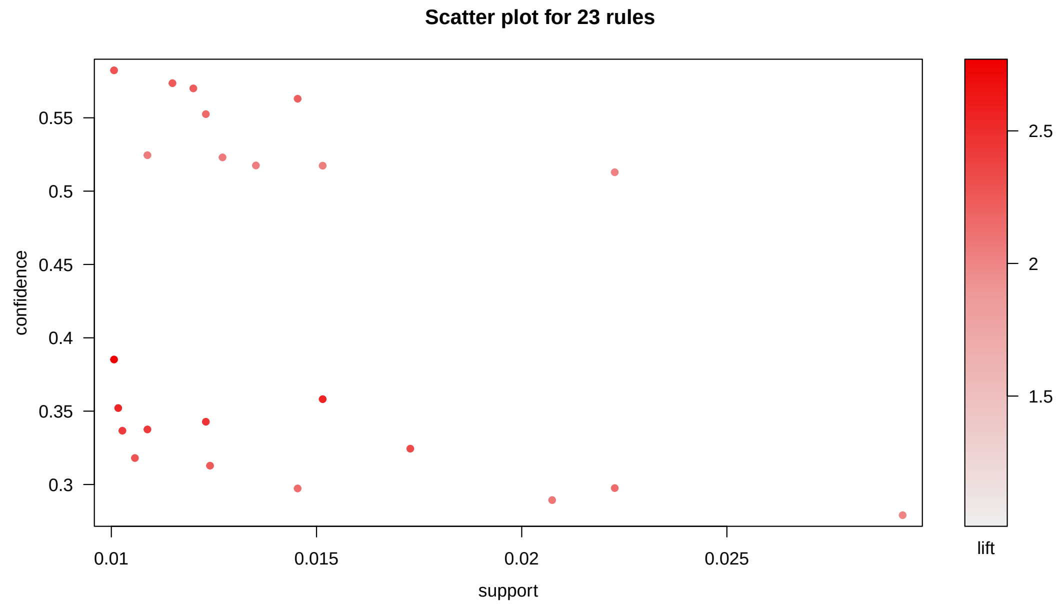
```
##     support           confidence          lift            count
##  Min.   :0.01007   Min.   :0.2517   Min.   :0.9932   Min.   :  99.0
##  1st Qu.:0.01159   1st Qu.:0.2965   1st Qu.:1.5175   1st Qu.: 114.0
##  Median :0.01454   Median :0.3582   Median :1.7716   Median : 143.0
##  Mean   :0.01961   Mean   :0.3697   Mean   :1.8695   Mean   : 192.9
##  3rd Qu.:0.02115   3rd Qu.:0.4252   3rd Qu.:2.1412   3rd Qu.: 208.0
##  Max.   :0.25552   Max.   :0.5862   Max.   :3.2950   Max.   :2513.0
##
## mining info:
##       data ntransactions support confidence
##  Groceries          9835    0.01       0.25

inspect(rules[1:5])

##     lhs              rhs                 support    confidence lift
## [1] {}            => {whole milk}       0.25551601 0.2555160  1.000000
## [2] {hard cheese} => {whole milk}       0.01006609 0.4107884  1.607682
## [3] {butter milk} => {other vegetables} 0.01037112 0.3709091  1.916916
## [4] {butter milk} => {whole milk}       0.01159126 0.4145455  1.622385
## [5] {ham}         => {whole milk}       0.01148958 0.4414062  1.727509
##     count
## [1] 2513
## [2]   99
## [3]  102
## [4]  114
## [5]  113
```

(o) Select the rules with a lift value above 2. Use the function `subset` for that.

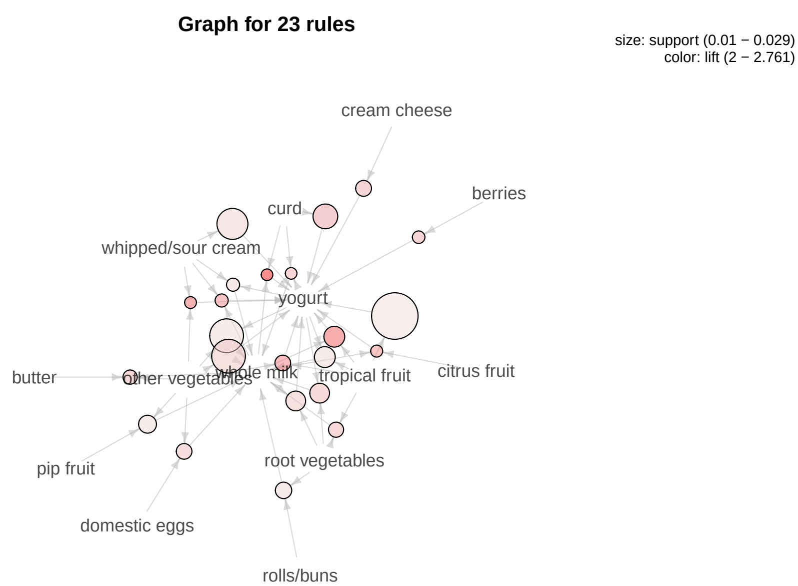
```
rules.sub <- subset(rules, subset = lift > 2)
inspect(rules.sub[1:5])

##     lhs                 rhs                support    confidence lift
## [1] {onions}         => {other vegetables} 0.01423488 0.4590164  2.372268
## [2] {berries}        => {yogurt}           0.01057448 0.3180428  2.279848
## [3] {hamburger meat} => {other vegetables} 0.01382816 0.4159021  2.149447
## [4] {cream cheese }  => {yogurt}           0.01240468 0.3128205  2.242412
## [5] {chicken}        => {root vegetables}  0.01087951 0.2535545  2.326221
##     count
## [1] 140
## [2] 104
## [3] 136
## [4] 122
## [5] 107
```

(p) Using one instruction only, select the rules that have lift value above 2 and the items "whole milk" or "yogurt" on the consequent. Inspect the selected rules by decreasing order of their lift value.

```
rules.sub <- subset(rules, subset = rhs %in% c("yogurt", "whole milk") & lift >
    2)
rules.sort <- sort(rules.sub, by = "lift")
inspect(rules.sort[1:5])

##     lhs                                         rhs      support
## [1] {whole milk,curd}                        => {yogurt} 0.01006609
## [2] {tropical fruit,whole milk}              => {yogurt} 0.01514997
## [3] {other vegetables,whipped/sour cream} => {yogurt} 0.01016777
## [4] {tropical fruit,other vegetables}     => {yogurt} 0.01230300
## [5] {whole milk,whipped/sour cream}       => {yogurt} 0.01087951
##     confidence lift     count
## [1] 0.3852140  2.761356  99
## [2] 0.3581731  2.567516 149
## [3] 0.3521127  2.524073 100
## [4] 0.3427762  2.457146 121
## [5] 0.3375394  2.419607 107

plot(rules.sub)
```




```r
plot(rules.sub, method = "graph")
```



2. Read the csv file of German Credit dataset into a data frame in R. This data set has the record of 1000 persons who took a credit by a bank.

```
df <- tbl_df(read.csv("german_credit.csv"))
```

(a) Remove the first attribute from the data frame, it is just an identifier for each record.

```
df <- df %>% select(-default)
```

(b) Try to convert the data frame into a transactions data set using the function `as`. What do you obtain?

```
dfT <- as(df, "transactions")
```

```
## Error in discretizeDF(from):  Problem with column installment_as_income_perc
## Error in discretize(x = c(4L, 2L, 2L, 2L, 3L, 2L, 3L, 2L, 2L, 4L, 3L,   :
##  The calculated breaks are:  1, 2, 4, 4
##  Some breaks are not unique.  Change the number of breaks or consider using method 'fixed'.
```

(c) Use the function `cut` to discretize the numerical attributes according to the following:

- duration_in_month: 4 equal-with intervals with labels "short","med-short","med-long","long";
- credit_amount: 4 equal-with intervals with labels "small","med-small","med-high","high";
- age: 4 equal-with intervals with labels "young adult","adult","senior","golden".
- to the rest of numerical attributes, simply use the function `as.factor`

```
df <- df %>% mutate(duration_in_month = cut(duration_in_month, 4, labels = c("short",
    "med-short", "med-long", "long")), credit_amount = cut(credit_amount, 4,
    labels = c("small", "med-small", "med-high", "high")), age = cut(age, 4,
    labels = c("young adult", "adult", "senior", "golden")))

df <- df %>% mutate_if(is.numeric, as.factor)
```

(d) Convert the data frame into a data set of transactions. What to you obtain? Use the function `itemInfo` to see what each item represents.

```
dfT <- as(df, "transactions")
item_dfT <- itemInfo(dfT)
head(item_dfT)
```

```
##                                                                     labels
## 1                                                account_check_status=< 0 DM
## 2 account_check_status=>= 200 DM / salary assignments for at least 1 year
## 3                                         account_check_status=0 <= ... < 200 DM
## 4                                       account_check_status=no checking account
## 5                                                    duration_in_month=short
## 6                                                duration_in_month=med-short
##               variables                                                 levels
## 1 account_check_status                                                   < 0 DM
## 2 account_check_status >= 200 DM / salary assignments for at least 1 year
## 3 account_check_status                                       0 <= ... < 200 DM
## 4 account_check_status                                     no checking account
## 5    duration_in_month                                                    short
## 6    duration_in_month                                                med-short
```

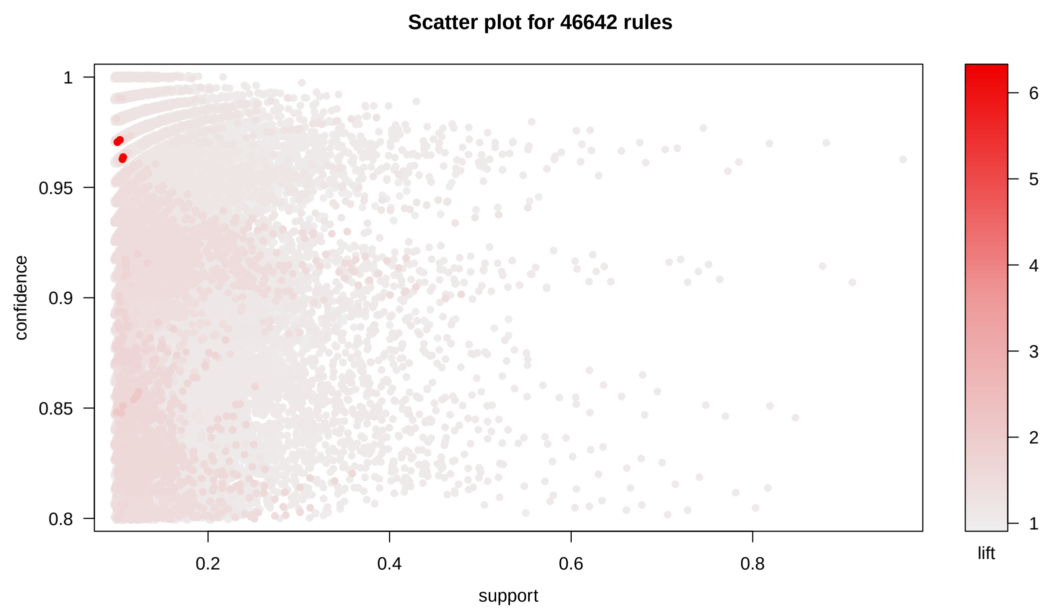
(e) Run apriori to obtain the association rules from the data set. Plot the obtained rules.

```
rules <- apriori(dfT)
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
```

```
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 100
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[80 item(s), 1000 transaction(s)] done [0.00s].
## sorting and recoding items ... [53 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.06s].
## writing ... [46642 rule(s)] done [0.01s].
## creating S4 object  ... done [0.02s].
```

```r
plot(rules)
```



(f) Observe the effect of filters and measures on the number of rules generated.

(g) Select the rules with confidence equal to 1. What does those rules tell you?

```r
rules.conf1 <- subset(rules, confidence == 1)
```

(h) Run apriori again, but this time imposing a minimum confidence equal to 0.6, minimum length of 2 and focusing only on attributes sex, age, job, housing and purpose of credit.

```r
myItems <- subset(item_dfT,variables %in% c("age","personal_status_sex","job","housing","purpose"))$labels
rules <- apriori(dfT,
                parameter = list(conf=0.6,minlen=2), # 44 rules
                appearance = list(both = myItems,
                                  default="none"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.6    0.1    1 none FALSE            TRUE       5     0.1      2
##  maxlen target   ext
##      10  rules FALSE
##
```

```
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 100
##
## set item appearances ...[25 item(s)] done [0.00s].
## set transactions ...[25 item(s), 1000 transaction(s)] done [0.00s].
## sorting and recoding items ... [15 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [44 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

(i) Identify rules $a \rightarrow b$ and $b \rightarrow a$. What do their quality values tell you?

```
# same lift and same support different confidence [19] {job=skilled employee
# / official} => {housing=own} 0.452 0.7174603 1.0062557 [20] {housing=own}
# => {job=skilled employee / official} 0.452 0.6339411 1.0062557 housing =
# own appears more often in transactions that contain job = skilled
```

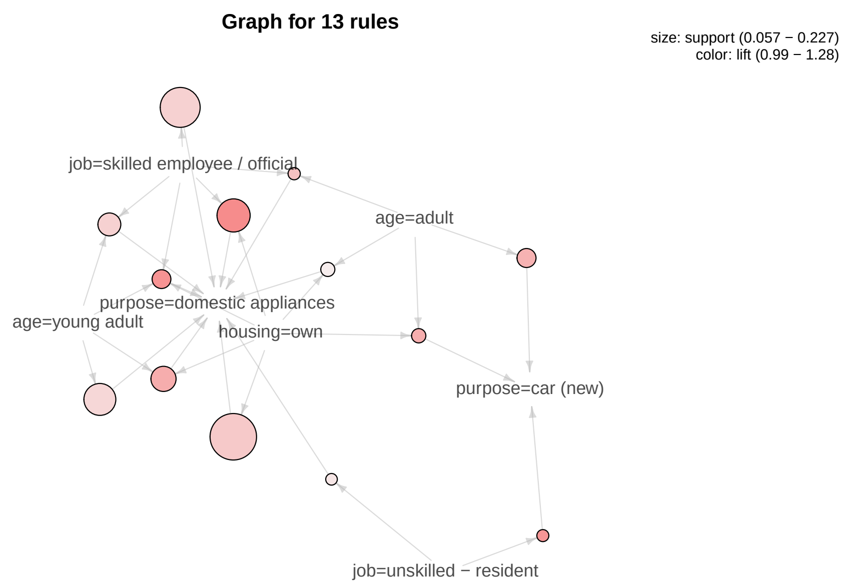
(j) Run apriori to obtain rules that relate the purpose of credit with age, job and housing. Impose a minimum support of 0.05, minimum confidence of 0.25 and a minimum length of 2. Could you propose a marketing campaign from the obtained rules?

```
my.lhs <- subset(item_dfT, variables %in% c("age", "job", "housing"))$labels
my.rhs <- subset(item_dfT, variables == "purpose")$labels
rules1 <- apriori(dfT, parameter = list(confidence = 0.25, minlen = 2, support = 0.05),
    appearance = list(lhs = my.lhs, rhs = my.rhs, default = "none"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5    0.05      2
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 50
##
## set item appearances ...[21 item(s)] done [0.00s].
## set transactions ...[21 item(s), 1000 transaction(s)] done [0.00s].
## sorting and recoding items ... [15 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [13 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].


# promote credit for domestic appliances among young adults or adults with
# skilled job and own housing
```

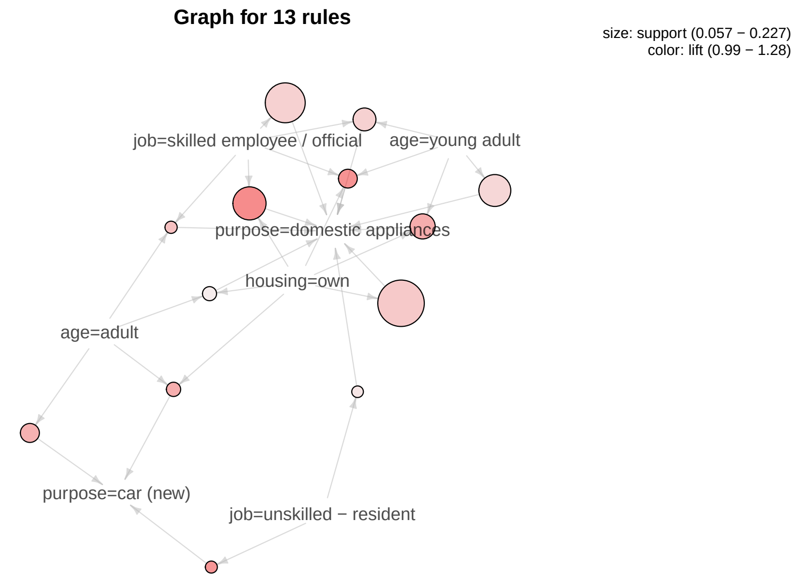
(k) Plot the previous set of rules using the method graph and graph with itemsets. What do these graphs tell you?

```
plot(rules1, method = "graph")
```




```r
plot(rules1, method = "graph", control = list(type = "itemsets"))
```

```
## Available control parameters (with default values):
## main  =  Graph for 13 rules
## nodeColors  =  c("#66CC6680", "#9999CC80")
## nodeCol  =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF",
## edgeCol  =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#515151FF", "#535353FF", "#555555FF", "#575757FF",
## alpha  =  0.5
## cex  =  1
## itemLabels  =  TRUE
## labelCol  =  #000000B3
## measureLabels  =  FALSE
## precision  =  3
## layout  =  NULL
## layoutParams  =  list()
## arrowSize  =  0.5
## engine  =  igraph
## plot  =  TRUE
## plot_options  =  list()
## max  =  100
## verbose  =  FALSE
```



(l) Plot the previous set of rules using the method grouped.

```
plot(rules1, method = "grouped")
```