# Association Rule Mining

*May 23, 2016*

## Contents



# Preparing the data

In project 3 we're working with a new data set from the College Scorecard. According to the data documentation:

> The College Scorecard is designed to increase transparency, putting the power in the hands of the public — from those choosing colleges to those improving college quality — to see how well different schools are serving their students.

There are a lot of variables available to us in this data set and we can use association mining to discovery trends and patterns.

```
install.packages('arules', dependencies = TRUE)
library(arules)

colleges = read.delim('./data/colleges.tsv',
                      sep = '\t',
                      header = TRUE)
```

## Feature engineering

Feature engineering is the process of using domain knowledge or expertise to construct new variables for use in data exploration and modeling. In association rule mining, we're looking for associations between *categories* of features, so the inputs all need to be discrete. Many of the variables in your data set are already discrete, like `state` or `locale`, but we'll need to use some feature engineering to coerce other variables of interest.

### discretize

The arules package provides a function that can help you create discrete variables from continuous ones, called discretize:

```
colleges =
  colleges %>%
    mutate(cost_category = discretize(cost))

# the default results in only 3 bins of uneven size
table(colleges$cost_category)
```

```
##
## [-1643,28095) [28095,57832) [57832,87570]
##          5971           320             3
```

```
colleges =
  colleges %>%
    mutate(cost_category = discretize(cost, method = 'frequency', categories = 4))

# that's better
table(colleges$cost_category)
```

```
##
## [-1643, 9387) [ 9387,15114) [15114,20541) [20541,87570]
##          1575          1572          1574          1573
```

discretize can break your variable up with several different methods, *e.g.* equal interval length, equal frequency, clustering-based, and custom interval discretization. Also you can specify categories to get as many or as few as you like. See ?discretize for details.

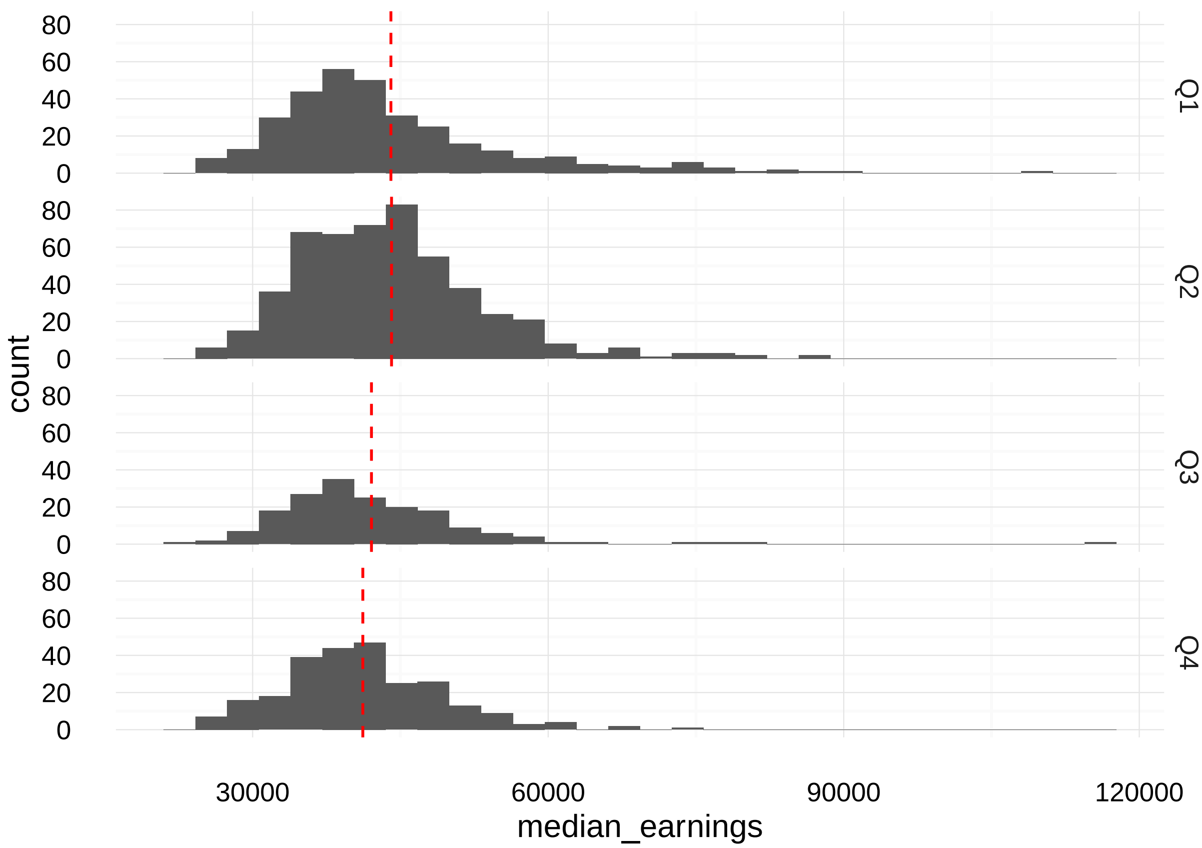
```
colleges$earnings_quartiles = discretize(colleges$median_earnings,
                                         method = 'frequency',
                                         categories = 4,
                                         labels = c('Q1', 'Q2', 'Q3', 'Q4'))

colleges$debt_quartiles = discretize(colleges$median_debt,
                                     method = 'frequency',
                                     categories = 4,
                                     labels = c('Q1', 'Q2', 'Q3', 'Q4'))
```

### SAT performance spread

Let's make a couple new features based on the variables available. I'm curious about the spread in student performance at schools, so to investigate that I can compute the difference between the top 75% and the bottom 25% of performance on the SATs.

```
# difference between Q3 and Q1.
colleges =
  colleges %>%
    mutate(sat_verbal_spread = discretize(sat_verbal_quartile_3 - sat_verbal_quartile_1,
```

```
                                    method = 'frequency',
                                    categories = 4,
                                    labels = c('Q1', 'Q2', 'Q3', 'Q4')),
         sat_math_spread = discretize(sat_math_quartile_3 - sat_math_quartile_1,
                                  method = 'frequency',
                                  categories = 4,
                                  labels = c('Q1', 'Q2', 'Q3', 'Q4')),
         sat_writing_spread = discretize(sat_writing_quartile_3 - sat_writing_quartile_1,
                                     method = 'frequency',
                                     categories = 4,
                                     labels = c('Q1', 'Q2', 'Q3', 'Q4')))
```




**STEM schools**

I have a feeling that there are interesting patterns earnings patterns for schools where students primarily study science, technology, engineering, and math (STEM) fields. So, I can make a variable indicating whether a school has a large proportion of STEM students.

```
colleges =
  colleges %>%
    mutate(stem_perc = architecture_major_perc + comm_tech_major_perc +
                       computer_science_major_perc + engineering_major_perc +
                       eng_tech_major_perc + bio_science_major_perc +
                       math_stats_major_perc,
           high_stem = ifelse(stem_perc >= 0.30, TRUE, FALSE))
```

## Generating rules

Before we start association rule mining, we need to select out the columns to mine. Remember, this algorithm works on discrete, categorical data so we need to remove or convert numerical columns.

```
college_features =
  colleges %>%
    select(state, locale, control, pred_deg, historically_black,
           men_only, women_only, religious, online_only,
           earnings_quartiles, debt_quartiles, high_stem, sat_verbal_spread,
           sat_math_spread, sat_writing_spread)
```

Then we're going to load our data into a `transaction` object in the `arules` package. Each row of `college_features` represents a 'transaction.'

```
college_features = as(college_features, 'transactions')
```

Once our data is a `transaction` object, we can inspect the `itemsets` which are just the characteristics corresponding with each college.

```
# view the itemsets
inspect(college_features[1:5])
```

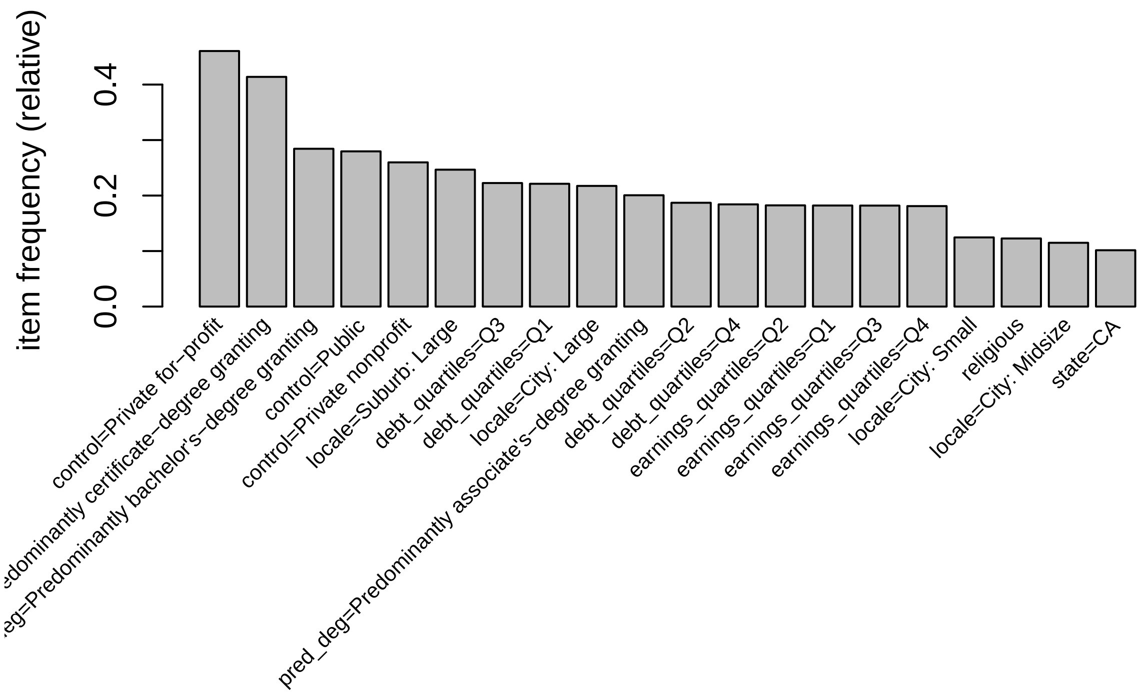
```
##   items                                                transactionID
## 1 {state=AL,
##    locale=City: Midsize,
##    control=Public,
##    pred_deg=Predominantly bachelor's-degree granting,
##    historically_black,
##    earnings_quartiles=Q2,
##    debt_quartiles=Q4,
##    high_stem,
##    sat_verbal_spread=Q1,
##    sat_math_spread=Q2}                                             1
## 2 {state=AL,
##    locale=City: Midsize,
##    control=Public,
##    pred_deg=Predominantly bachelor's-degree granting,
##    earnings_quartiles=Q4,
##    debt_quartiles=Q3,
##    sat_verbal_spread=Q3,
##    sat_math_spread=Q4}                                             2
## 3 {state=AL,
##    locale=City: Midsize,
##    control=Private nonprofit,
##    pred_deg=Predominantly bachelor's-degree granting,
##    religious,
##    earnings_quartiles=Q3}                                          3
## 4 {state=AL,
##    locale=City: Midsize,
##    control=Public,
##    pred_deg=Predominantly bachelor's-degree granting,
```

```
##    earnings_quartiles=Q4,
##    debt_quartiles=Q3,
##    high_stem,
##    sat_verbal_spread=Q4,
##    sat_math_spread=Q4}                                                  4
## 5 {state=AL,
##    locale=City: Midsize,
##    control=Public,
##    pred_deg=Predominantly bachelor's-degree granting,
##    historically_black,
##    earnings_quartiles=Q2,
##    debt_quartiles=Q4,
##    sat_verbal_spread=Q2,
##    sat_math_spread=Q2}                                                  5
```

```r
summary(college_features)
```

```
## transactions as itemMatrix in sparse format with
##  7308 rows (elements/itemsets/transactions) and
##  105 columns (items) and a density of 0.05871713
##
## most frequent items:
##                         control=Private for-profit
##                                               3365
## pred_deg=Predominantly certificate-degree granting
##                                               3025
##  pred_deg=Predominantly bachelor's-degree granting
##                                               2078
##                                     control=Public
##                                               2044
##                          control=Private nonprofit
##                                               1899
##                                            (Other)
##                                              32645
##
## element (itemset/transaction) length distribution:
## sizes
##    3    4    5    6    7    8    9   10   11   12
##   14  679 1858 3051  392  283  639  370   21    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   5.000   6.000   6.165   6.000  12.000
##
## includes extended item information - examples:
##     labels variables levels
## 1 state=AK     state     AK
## 2 state=AL     state     AL
## 3 state=AR     state     AR
##
## includes extended transaction information - examples:
##   transactionID
## 1             1
## 2             2
## 3             3
```

```r
# plot the most frequent items
itemFrequencyPlot(college_features, topN = 20, cex = 0.70)
```




The summary of the data set provides an overview of the most frequent items along with other distributional details. Further, we can see which characteristics are the most common in the data set the `itemFrequencyPlot()`.

Next, use the `apriori()` function to find all rules with a specified minimum support and confidence, *e.g.* support = 0.01 and confidence = 0.6.

```r
# run the apriori algorithm
rules = apriori(college_features,
                parameter = list(sup = 0.01, conf = 0.6, target = 'rules'))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.6    0.1    1 none FALSE            TRUE    0.01      1     10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 73
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[105 item(s), 7308 transaction(s)] done [0.00s].
```

```
## sorting and recoding items ... [76 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.00s].
## writing ... [930 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# print distribution information
summary(rules)
```

```
## set of 930 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4   5   6
##  55 376 376 115   8
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.000   4.000   3.618   4.000   6.000
##
## summary of quality measures:
##     support          confidence          lift
##  Min.   :0.01013   Min.   :0.6000   Min.   : 1.314
##  1st Qu.:0.01177   1st Qu.:0.6667   1st Qu.: 2.517
##  Median :0.01574   Median :0.8045   Median : 3.389
##  Mean   :0.02188   Mean   :0.8118   Mean   : 3.748
##  3rd Qu.:0.02217   3rd Qu.:0.9638   3rd Qu.: 3.848
##  Max.   :0.30624   Max.   :1.0000   Max.   :18.213
##
## mining info:
##              data ntransactions support confidence
##  college_features          7308    0.01        0.6
```

```r
items(rules)
```

```
## itemMatrix in sparse format with
##  930 rows (elements/transactions) and
##  105 columns (items)
```

```r
# view the rules
inspect(head(rules))
```

```
##   lhs                         rhs                                                  support confidenc
## 1 {state=PR}               => {earnings_quartiles=Q1}                            0.01272578  0.69402
## 2 {state=AZ}               => {control=Private for-profit}                       0.01135742  0.66400
## 3 {state=CO}               => {control=Private for-profit}                       0.01039956  0.60800
## 4 {historically_black}     => {pred_deg=Predominantly bachelor's-degree granting} 0.01122058  0.82828
## 5 {state=NJ}               => {locale=Suburb: Large}                             0.01587302  0.72955
## 6 {sat_writing_spread=Q4} => {sat_verbal_spread=Q4}                             0.01067323  0.65546
```

```r
# sort the rules by lift
inspect(head(sort(rules, by = 'lift')))
```

```
##   lhs                                                   rhs                       support confidence
## 1 {sat_writing_spread=Q4}                            => {sat_verbal_spread=Q4} 0.01067323  0.6554622
## 2 {religious,
##    sat_math_spread=Q4}                               => {sat_verbal_spread=Q4} 0.01053640  0.6260163
## 3 {control=Private nonprofit,
##    religious,
##    sat_math_spread=Q4}                               => {sat_verbal_spread=Q4} 0.01053640  0.6260163
## 4 {pred_deg=Predominantly bachelor's-degree granting,
##    religious,
##    sat_math_spread=Q4}                               => {sat_verbal_spread=Q4} 0.01026273  0.6198347
## 5 {control=Private nonprofit,
##    pred_deg=Predominantly bachelor's-degree granting,
##    religious,
##    sat_math_spread=Q4}                               => {sat_verbal_spread=Q4} 0.01026273  0.6198347
## 6 {pred_deg=Predominantly bachelor's-degree granting,
##    sat_math_spread=Q1,
##    sat_writing_spread=Q1}                            => {sat_verbal_spread=Q1} 0.01313629  0.7500000
```

```r
# view quality metrics
quality(rules) = cbind(quality(rules), coverage = coverage(rules))
head(quality(rules))
```

```
##      support confidence      lift   coverage
## 1 0.01272578  0.6940299  3.810646 0.01833607
## 2 0.01135742  0.6640000  1.442054 0.01710454
## 3 0.01039956  0.6080000  1.320435 0.01710454
## 4 0.01122058  0.8282828  2.912941 0.01354680
## 5 0.01587302  0.7295597  2.957084 0.02175698
## 6 0.01067323  0.6554622 18.213375 0.01628352
```

The result of mining the colleges data associations between the characteristics expressed in the form of rules. We end up producing 930 such associations, or rules.

For an overview of the rules `summary()` can be used. It shows the number of rules, the most frequent items contained in the left-hand-side and the right-hand-side and their respective length distributions and summary statistics for the quality measures returned by the mining algorithm.

## Increasing rule size with `maxlen`

By default, rules will not exceed 6 objects in size, but you can change that length constraint with `maxlen`.

```r
# allow rules to up to size 10
long_rules = apriori(college_features,
                     parameter = list(sup = 0.001, conf = 0.6,
                                      target = 'rules', maxlen = 10))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.6    0.1    1 none FALSE            TRUE   0.001      1     10
##  target   ext
```

```
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 7
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[105 item(s), 7308 transaction(s)] done [0.00s].
## sorting and recoding items ... [98 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 done [0.01s].
## writing ... [25755 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
inspect(head(sort(long_rules, by = 'lift'), 2))
```

```
##   lhs                                                      rhs            support confidence     lift
## 1 {state=NY,
##    pred_deg=Predominantly bachelor's-degree granting,
##    earnings_quartiles=Q1}                               => {men_only} 0.001231527        0.9 99.65455
## 2 {state=NY,
##    control=Private nonprofit,
##    pred_deg=Predominantly bachelor's-degree granting,
##    earnings_quartiles=Q1}                               => {men_only} 0.001231527        0.9 99.65455
```

### Investigate interesting patterns

It's common for the result of association rule mining to produce a huge amount of rules. The `subset()` function can be used to narrow in on the results by filtering on characteristics of interest.

For example, let's filter for control=Private for profit to see what is associated with that.

```
high_earners = subset(rules, subset = rhs %in% 'earnings_quartiles=Q4' & lift > 1)

inspect(head(high_earners, n = 5, by = 'lift'))
```

```
##   lhs                                                      rhs                        support confiden
## 1 {locale=Suburb: Large,
##    pred_deg=Predominantly bachelor's-degree granting,
##    sat_math_spread=Q2}                                  => {earnings_quartiles=Q4} 0.01176793  0.81904
## 2 {locale=Suburb: Large,
##    sat_math_spread=Q2}                                  => {earnings_quartiles=Q4} 0.01190476  0.80555
## 3 {locale=Suburb: Large,
##    pred_deg=Predominantly bachelor's-degree granting,
##    sat_verbal_spread=Q2}                                => {earnings_quartiles=Q4} 0.01122058  0.80392
## 4 {locale=City: Large,
##    pred_deg=Predominantly bachelor's-degree granting,
##    sat_verbal_spread=Q2}                                => {earnings_quartiles=Q4} 0.01163109  0.80188
## 5 {locale=City: Large,
##    sat_verbal_spread=Q2}                                => {earnings_quartiles=Q4} 0.01190476  0.79816
```

## Visualizing with `arulesViz`

The authors of `arules` followed on that package with `arulesViz`, which provides lots of great pre-packaged visualization for exploring your association rules.

```
# arules plot template
plot(x,
     method = NULL,
     measure = 'support',
     shading = 'lift',
     interactive = FALSE,
     data,
     control = ...)
```

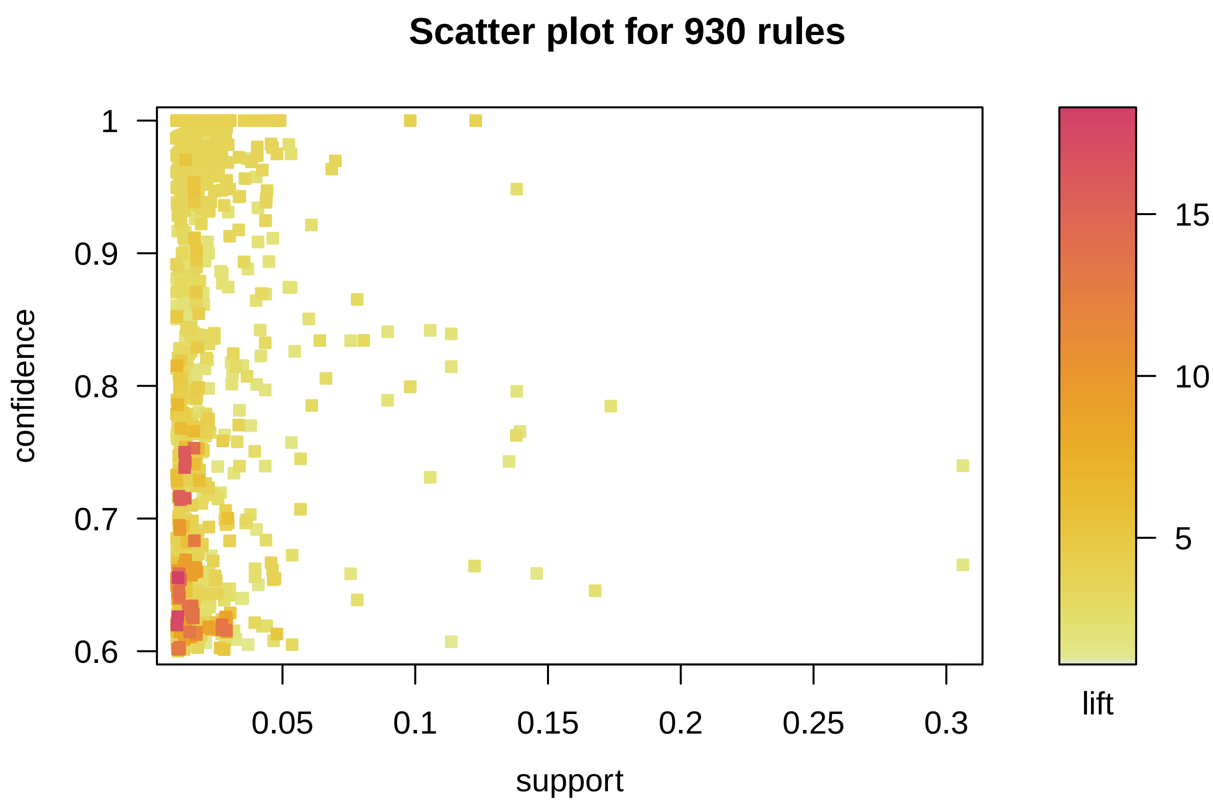
where:

- *x*: is the set of rules to be visualized
- *method*: the visualization method
- *measure*: and shading contain the interest measures used by the plot
- *interactive*: indicates whether you want to interactively explore
- *data*: can contain the transaction data set used to mine the rules
- *control*: list with further control arguments to customize the plot

```
install.packages('arulesViz', dependencies = TRUE)

# you might need to install Rgraphviz from this repository
source('http://bioconductor.org/biocLite.R')
biocLite('Rgraphviz')

library(arulesViz)
```

We can start with a visualization of association rules as a scatter plot with two measures on the axes. The default `plot()` for association rules is a scatter plot using support and confidence on the axes. Lift is used as the color of the points.

```
plot(rules)
```

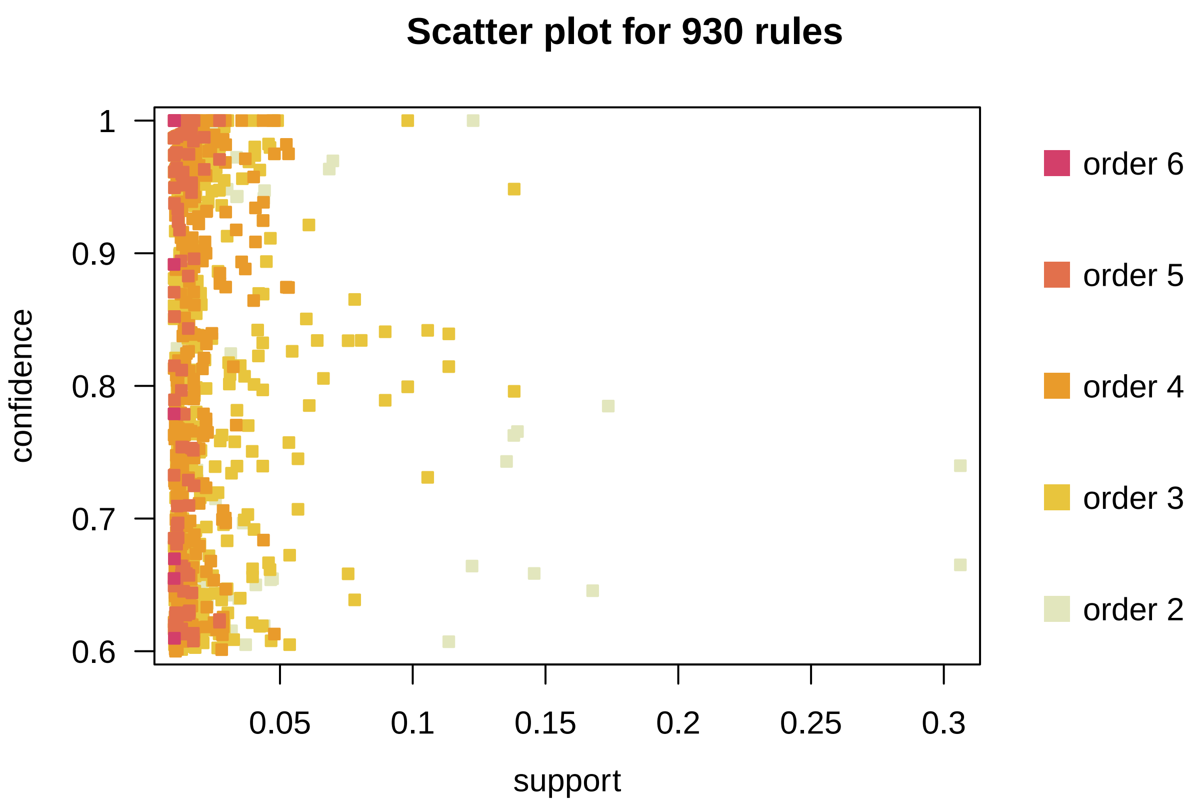



```r
head(quality(rules))
```

```
##      support confidence      lift   coverage
## 1 0.01272578  0.6940299  3.810646 0.01833607
## 2 0.01135742  0.6640000  1.442054 0.01710454
## 3 0.01039956  0.6080000  1.320435 0.01710454
## 4 0.01122058  0.8282828  2.912941 0.01354680
## 5 0.01587302  0.7295597  2.957084 0.02175698
## 6 0.01067323  0.6554622 18.213375 0.01628352
```

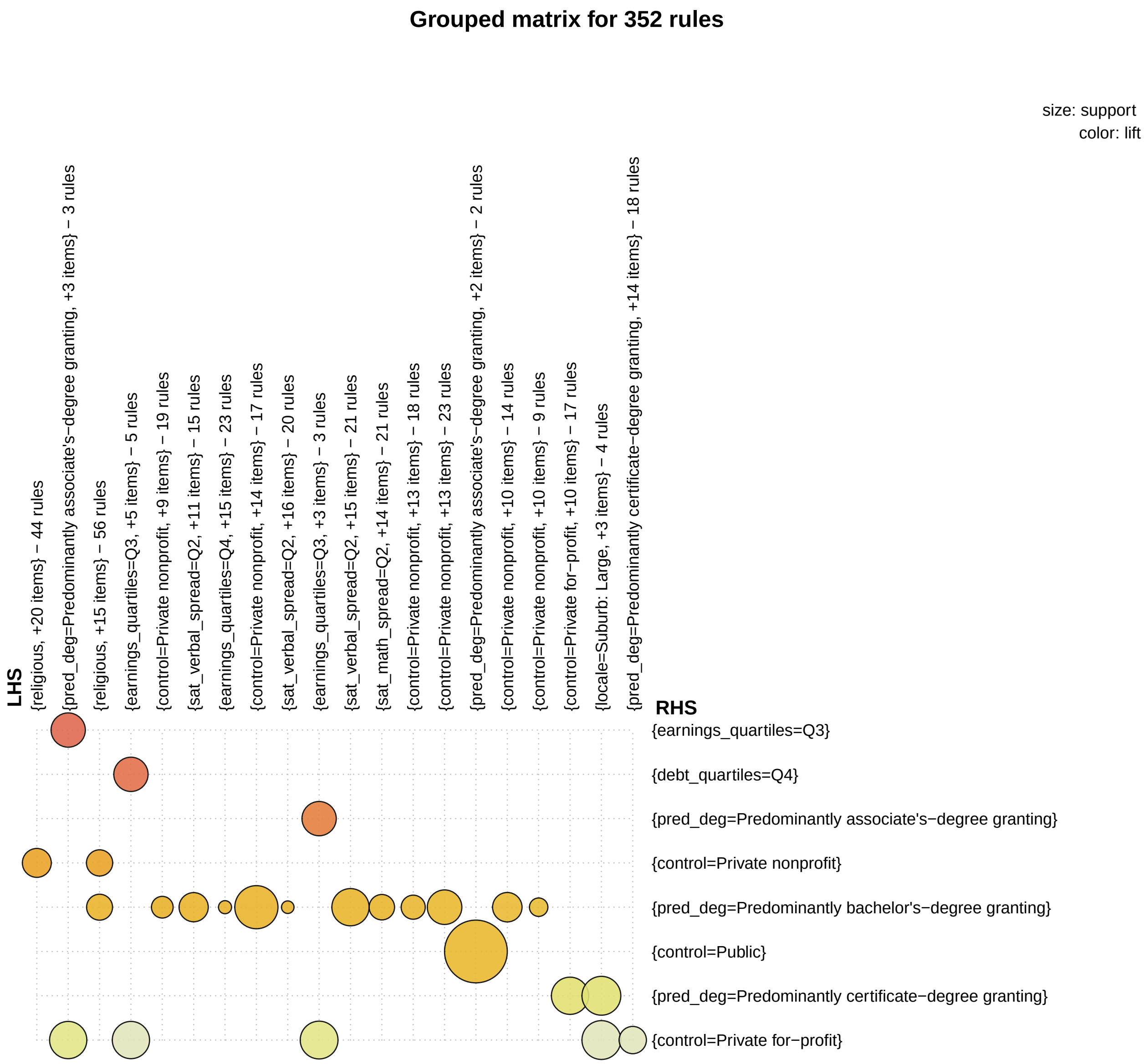
Another version of the scatter plot called two-key plot. Here support and confidence are used for the x and y-axes and the color of the points indicates the 'order,' *i.e.*, the number of items contained in the rule:

```r
plot(rules, shading = 'order')
```

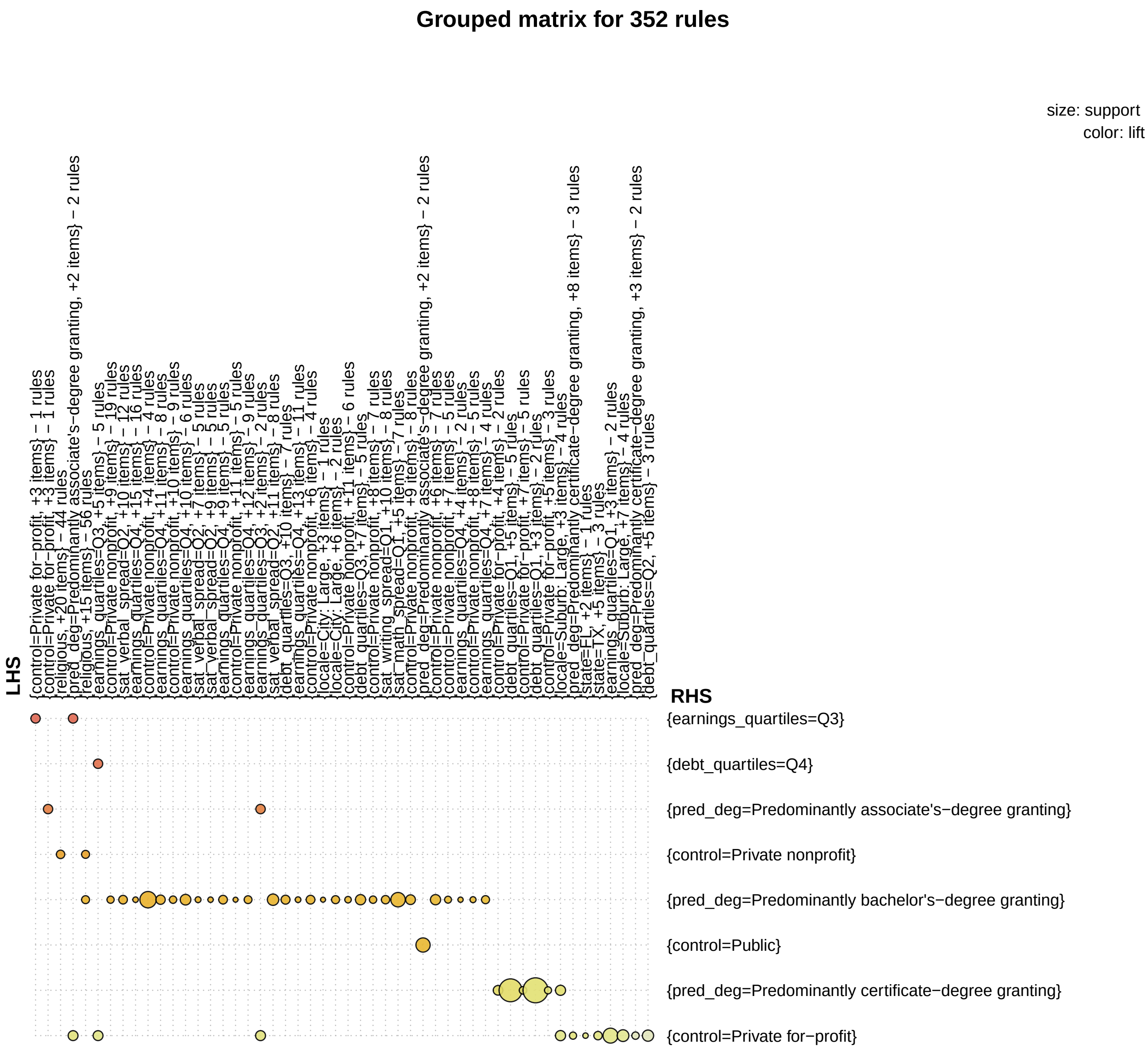



We can filter to narrow in on rules of interest:

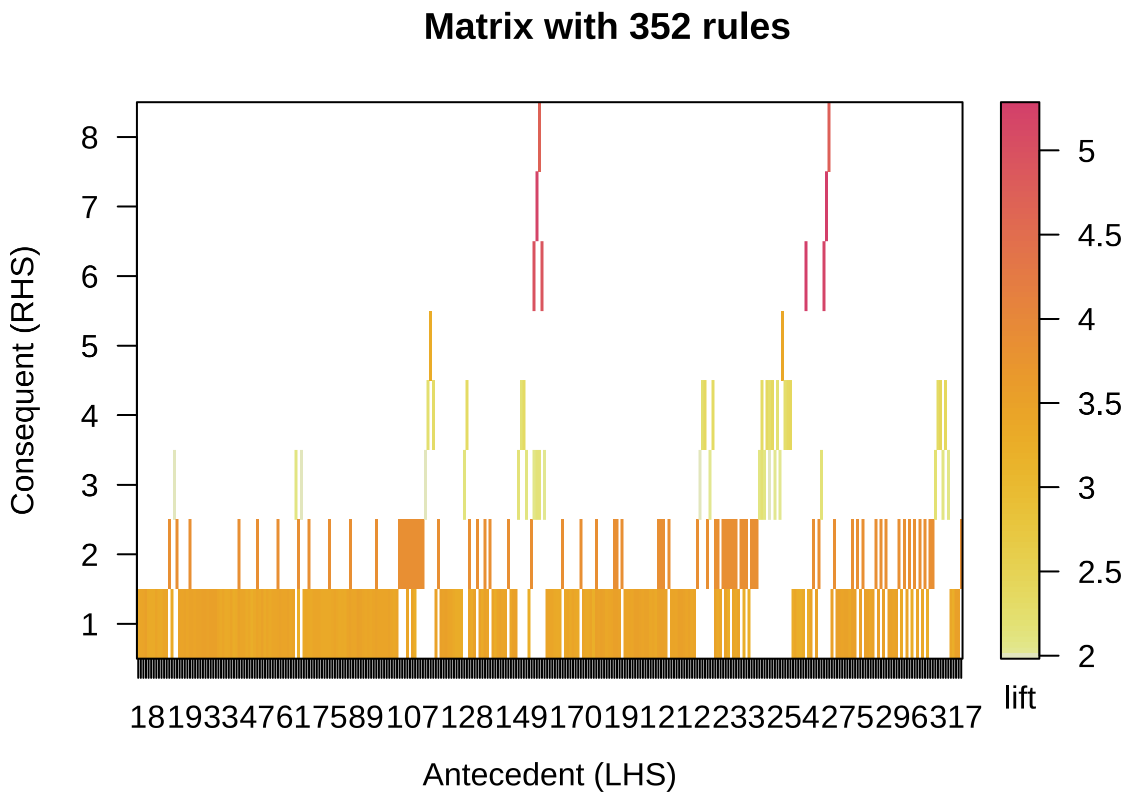
```
subrules = rules[quality(rules)$confidence > 0.9]

plot(subrules, method = 'grouped')
```



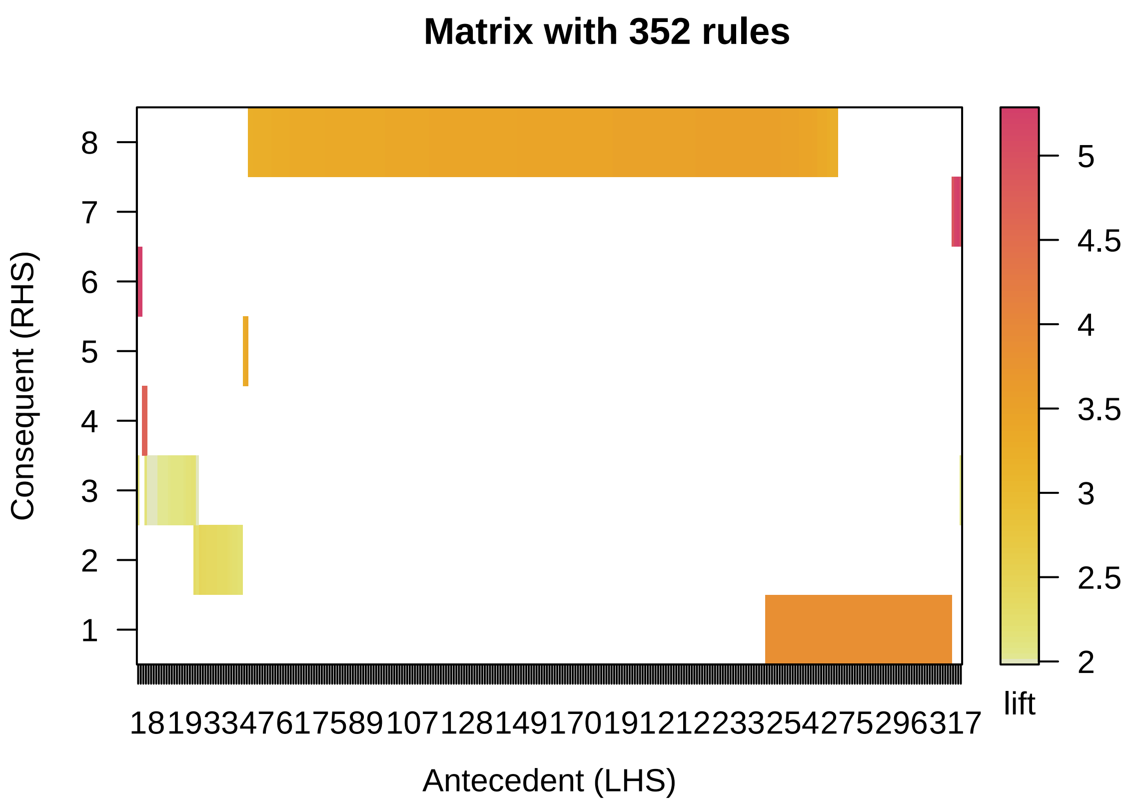

```r
plot(subrules, method = 'grouped', control = list(k = 50))
```



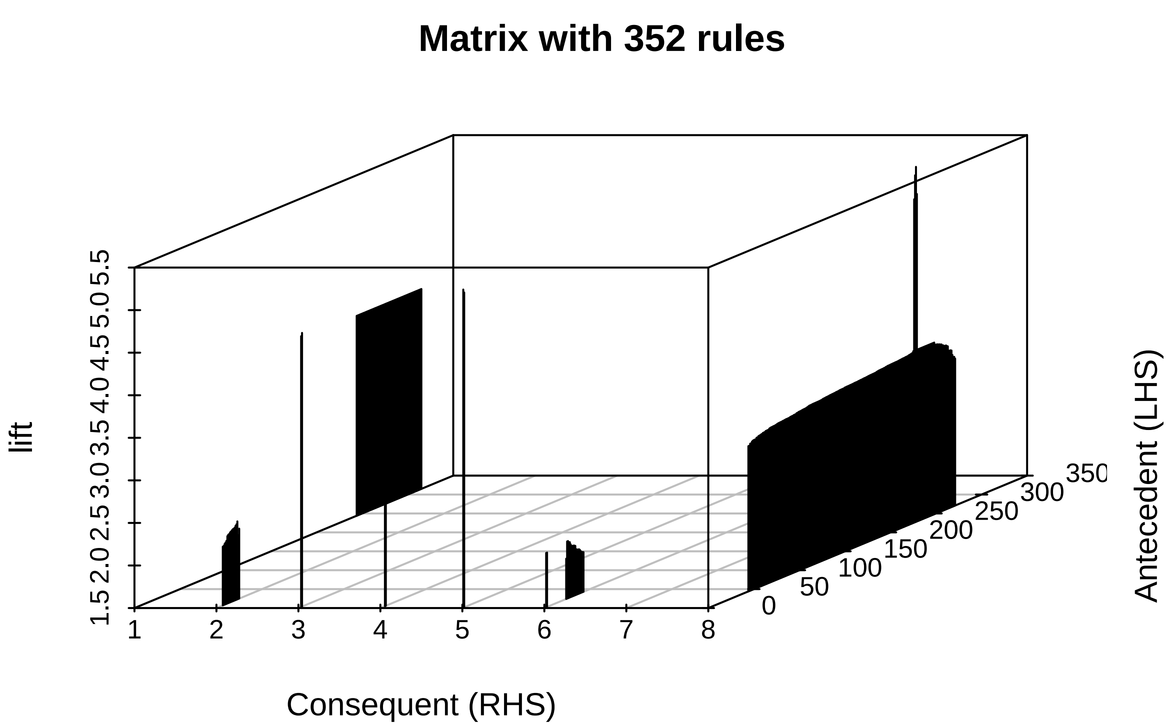

```r
plot(subrules,
     method = 'matrix',
     measure = 'lift')
```



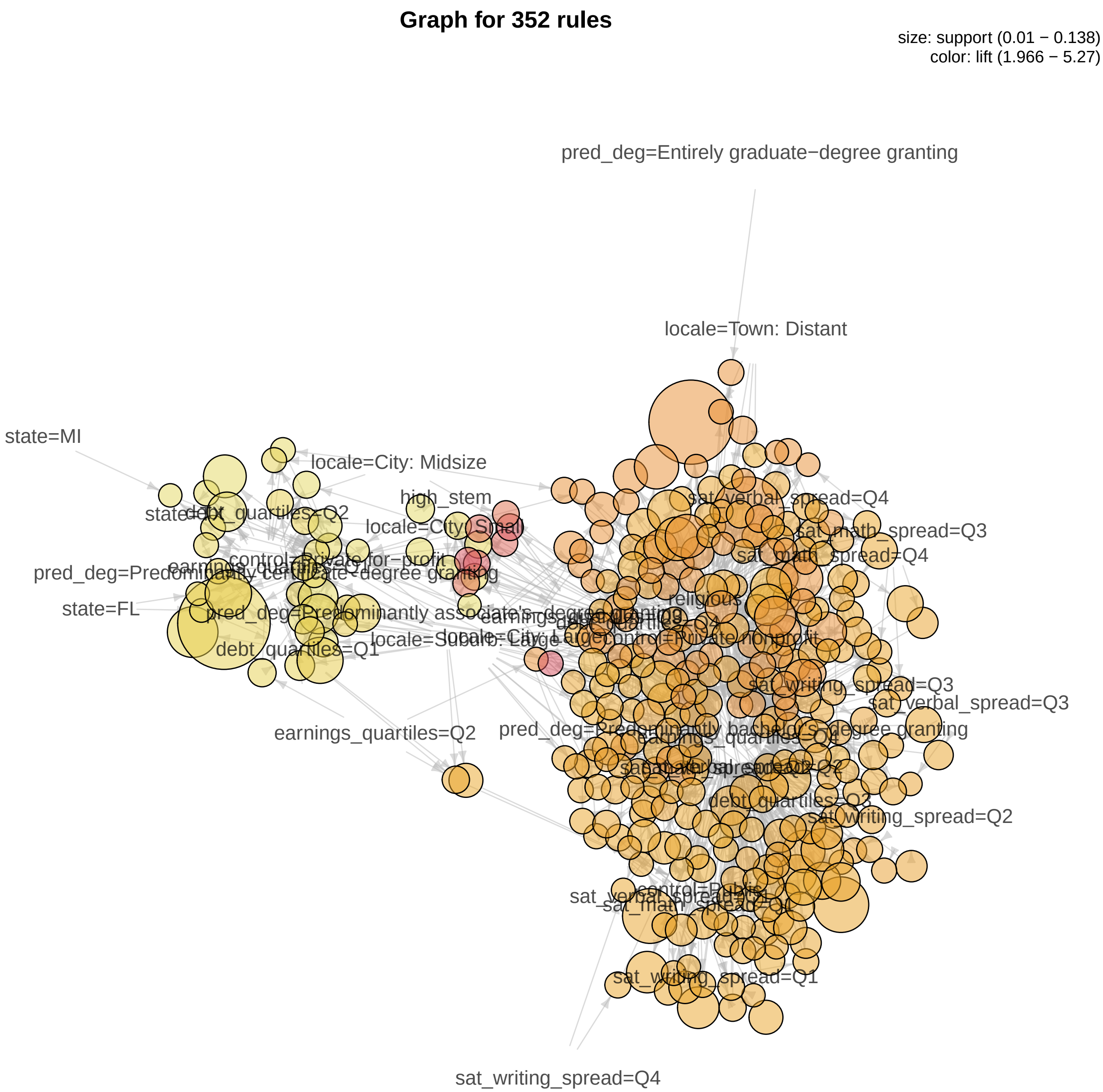

```r
# reorder based on
plot(subrules,
     method = 'matrix',
     measure = 'lift',
     control = list(reorder = TRUE))
```




```r
plot(subrules,
     method = 'matrix3D',
     measure = 'lift',
     control = list(reorder = TRUE))
```



```r
# plot a graph
plot(subrules, method = 'graph')
```




# Project 3, Part 1

## Feature engineering

1. Use feature engineering to construct 3 additional variables to include in your association mining.
2. Plot your new features to view their distributions.

## `arules`

3. Mine for rules with data that includes your new features from 1. Filter the rules for your features and describe the patterns you find. **Note:** you might need to try different values of support and confidence to generate rules with your newly created features.