Association Rule Mining in R

Erin Shellman

May 18, 2015

Contents

Reading in the data

The *arules* package reads in data slightly differently than other packages. Specifically, it has its own `read.transactions` function that can be used to read columns stored as transactions.

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

# read in the co-purchased items
bought = read.transactions('people_who_bought.txt',
                           format = 'basket',
                           sep = ',')

# read in the product catalog
catalog = read.delim('product_catalog.tsv',
                     header = TRUE,
                     sep = '\t'
                     quote = '')
```

The file *people_who_bought.txt* consists of rows of co-purchased items, in the form:

0,1,2,3,4,5
1,0,2,4,5,15
2,0,11,12,13,14
3,63,64,65,66,67

Where each row represents a "transaction." In reality each row corresponds to a product and the set of items that are frequently purchased together with the item.

Once we've read in the itemsets and the catalog, we can append the catalog data to the itemsets. This allows us to do convenient things like aggregate at the product category level.

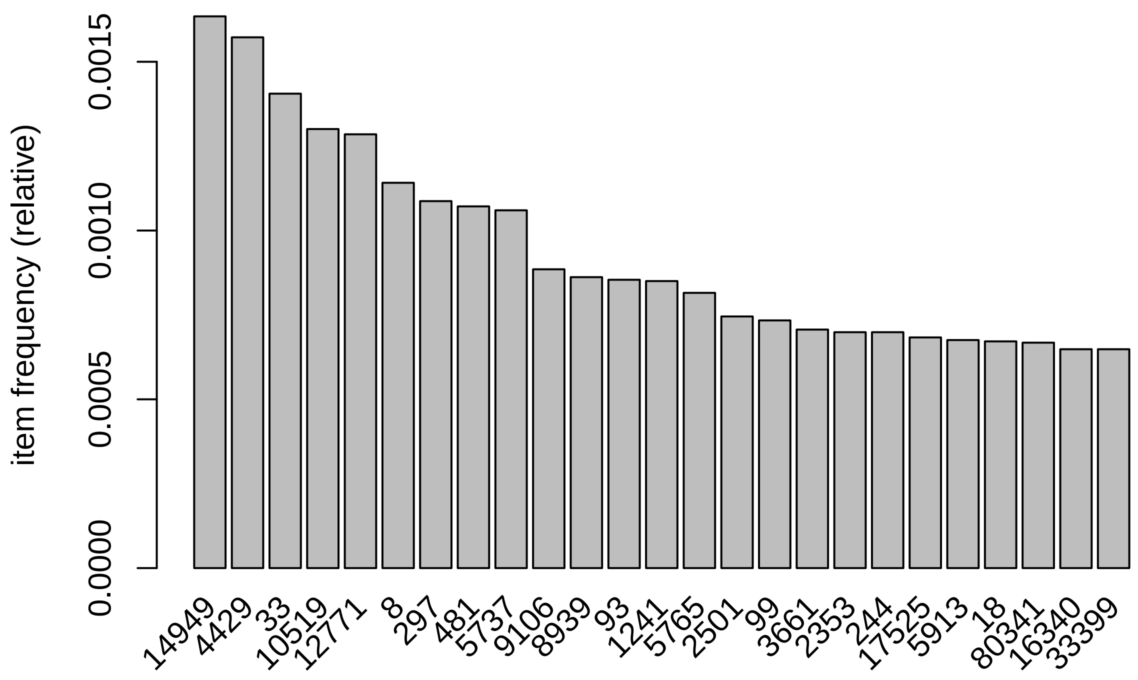
```r
# attach the catalog data onto the item-sets
itemInfo(bought) = catalog

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

```
##   items
## 1 {0,
##    1,
##    2,
##    3,
##    4,
##    5}
## 2 {0,
##    1,
##    15,
##    2,
##    4,
##    5}
## 3 {0,
##    11,
##    12,
##    13,
##    14,
##    2}
## 4 {3,
##    63,
##    64,
##    65,
##    66,
##    67}
## 5 {16,
##    17,
##    18,
##    19,
##    4,
##    7}
```

```r
summary(bought)
```

```
## transactions as itemMatrix in sparse format with
##  257570 rows (elements/itemsets/transactions) and
##  262111 columns (items) and a density of 2.210642e-05
##
## most frequent items:
##   14949    4429      33   10519   12771 (Other)
##     421     405     362     335     331 1490593
##
## element (itemset/transaction) length distribution:
## sizes
##      2      3      4      5      6
##   3803   5654   6557   7685 233871
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   6.000   6.000   5.794   6.000   6.000
##
## includes extended item information - examples:
##   labels avg_rating downloaded group reviews_count salesrank
## 1      0         NA         NA                   NA        NA
## 2      1          5          2  Book              2    396585
## 3     10          4          6  Book             6    220379
##                     title
## 1
## 2  Patterns of Preaching
## 3 The Edward Said Reader
```

```
# plot the most frequent items
itemFrequencyPlot(bought, topN = 25)
```

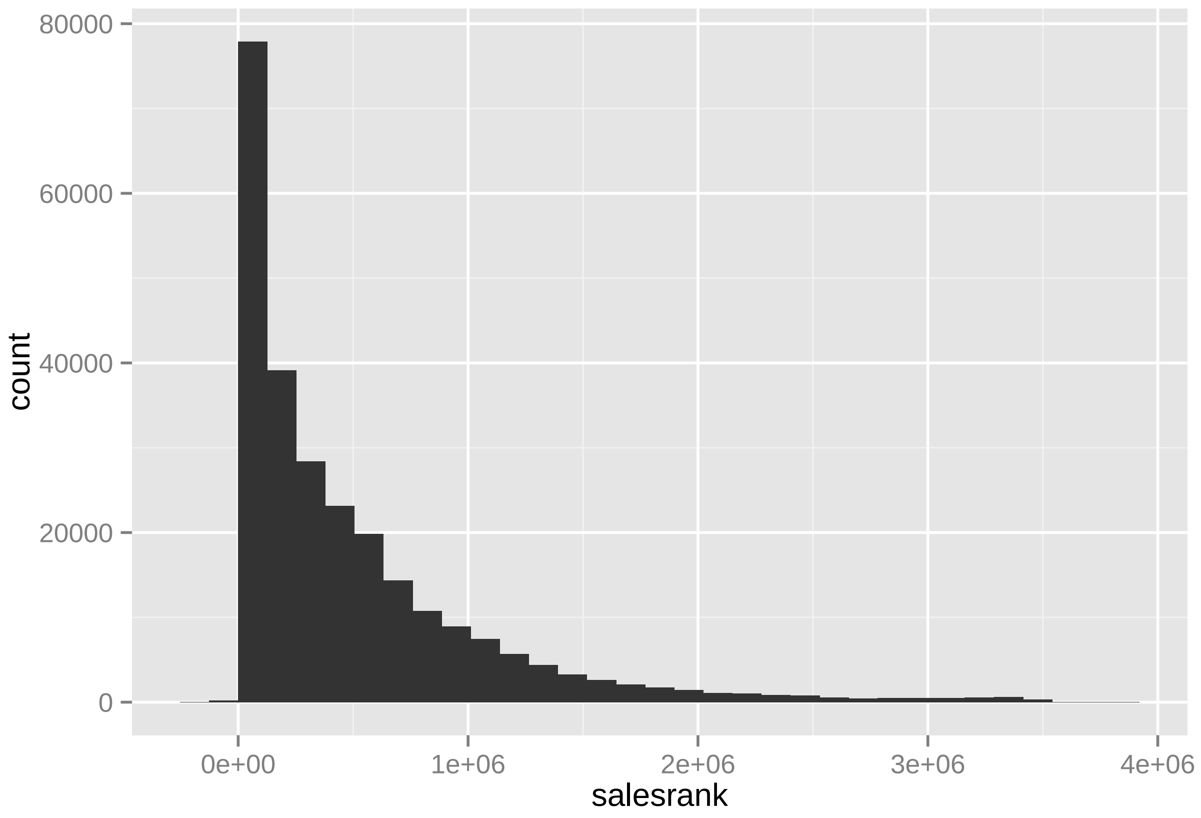


Exploratory data analysis

We have a new dataset again, so let's explore a bit.

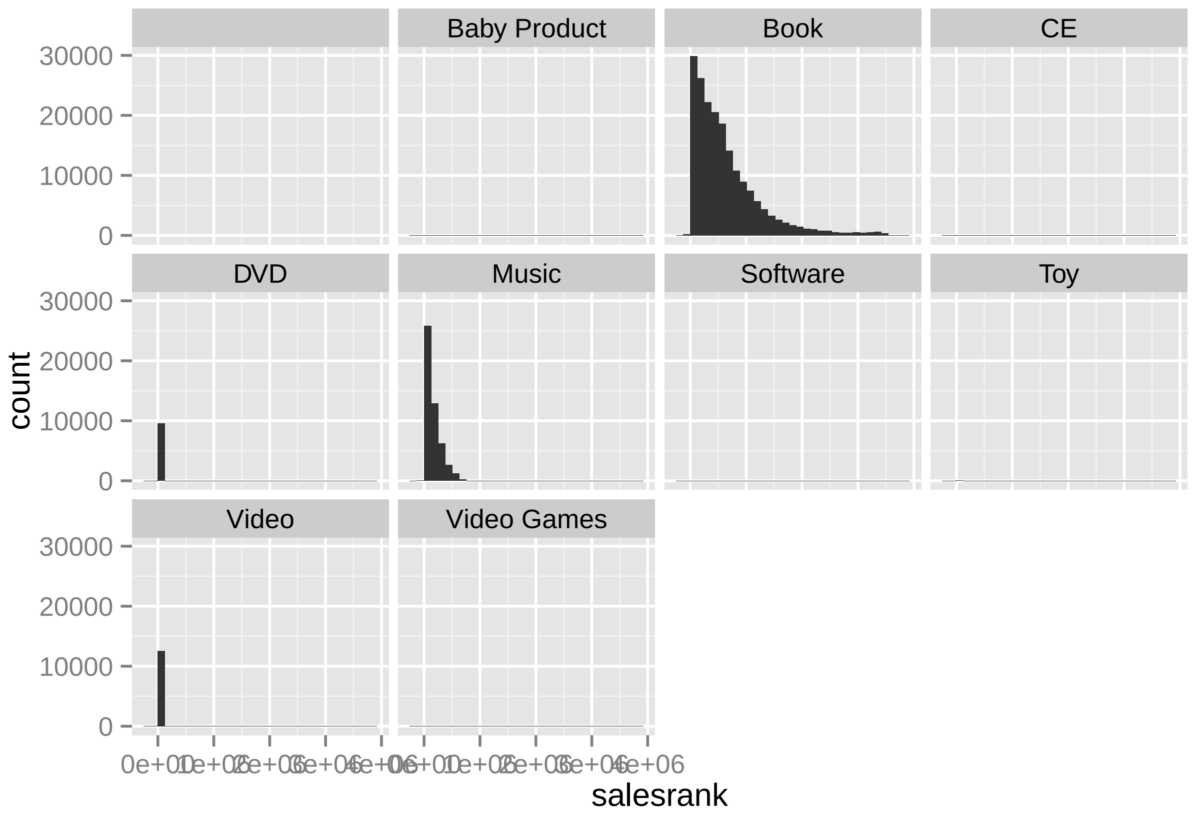
```
# what does the distribution of salesrank look like?
ggplot(catalog, aes(x = salesrank)) +
  geom_histogram()
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

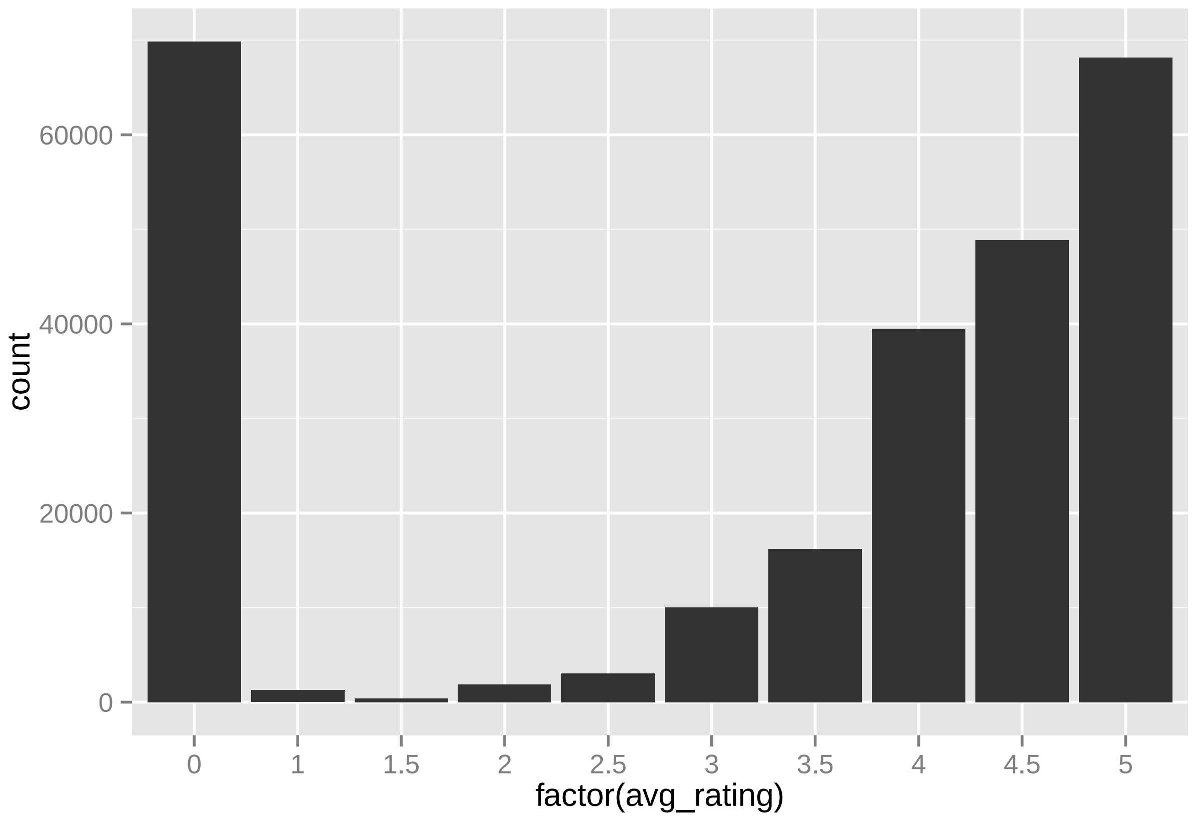


```r
# how about by product group?
ggplot(catalog, aes(x = salesrank)) +
  geom_histogram() +
  facet_wrap(~ group)
```

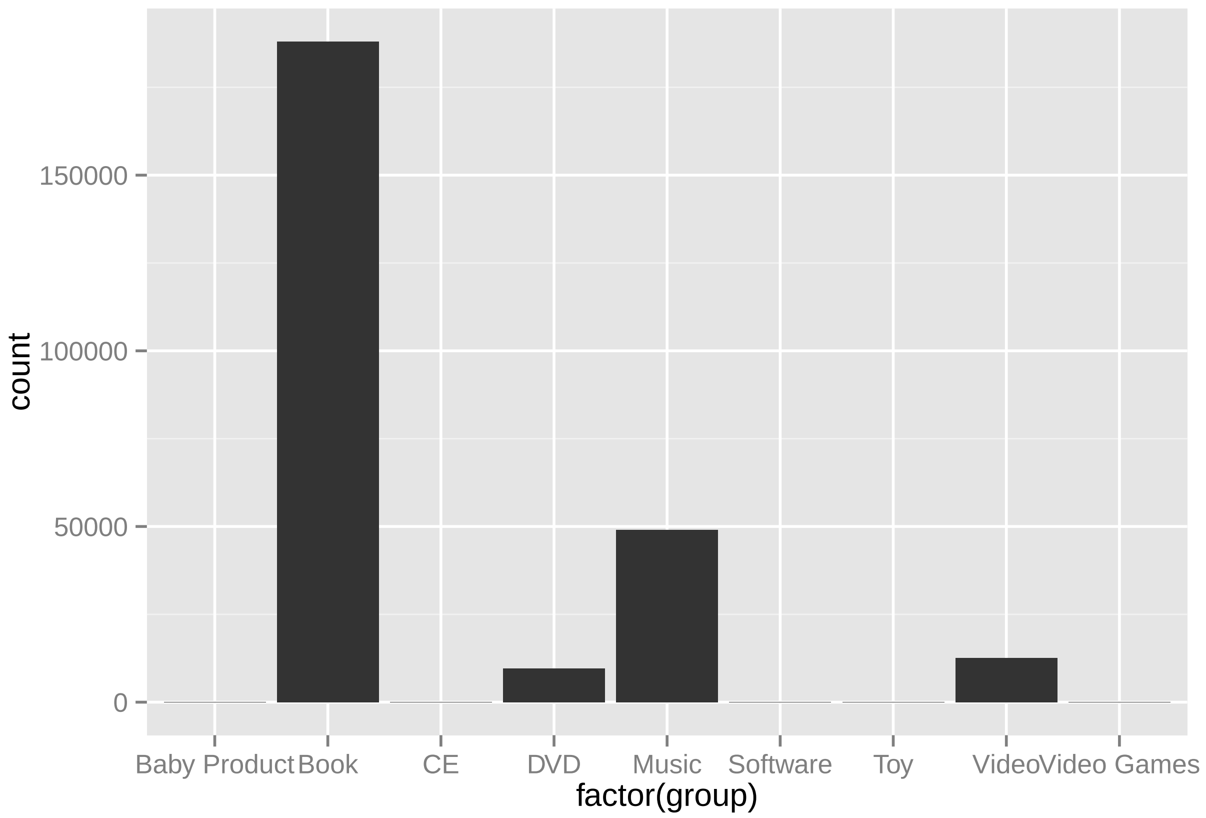
```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



```r
# how about average rating?
ggplot(na.omit(catalog), aes(x = factor(avg_rating))) +
  geom_bar()
```



```r
# how many items in each product group?
ggplot(na.omit(catalog), aes(x = factor(group))) +
  geom_bar()
```



Generating rules

In this data set we have a large catalog and relatively few transactions, which means for any given item pairing the support will be relatively small (in this case support = 0.0001).

```
# run the apriori algorithm
rules = apriori(bought,
                parameter = list(sup = 0.0001, conf = 0.0001, target = 'rules'))
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##       1e-04    0.1    1 none FALSE            TRUE   1e-04      1     10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## apriori - find association rules with the apriori algorithm
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[262111 item(s), 257570 transaction(s)] done [0.56s].
## sorting and recoding items ... [2355 item(s)] done [0.03s].
## creating transaction tree ... done [0.08s].
## checking subsets of size 1 2 3 done [0.04s].
## writing ... [2601 rule(s)] done [0.01s].
## creating S4 object  ... done [0.11s].
```

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

```
##   lhs    rhs          support   confidence lift
## 1 {}  => {26078}  0.0001048259 0.0001048259    1
## 2 {}  => {201419} 0.0001048259 0.0001048259    1
## 3 {}  => {50524}  0.0001009434 0.0001009434    1
## 4 {}  => {50451}  0.0001009434 0.0001009434    1
## 5 {}  => {165128} 0.0001009434 0.0001009434    1
## 6 {}  => {20169}  0.0001009434 0.0001009434    1
```

```
# sort the rules by lift
inspect(head(sort(rules, by = 'lift'), 10))
```

```
##    lhs         rhs          support confidence     lift
## 1  {127682} => {134413} 0.0001048259  0.7941176 4870.021
## 2  {134413} => {127682} 0.0001048259  0.6428571 4870.021
## 3  {35412}  => {35413}  0.0001087083  0.6086957 4611.228
## 4  {35413}  => {35412}  0.0001087083  0.8235294 4611.228
## 5  {55573}  => {55574}  0.0001009434  1.0000000 4518.772
## 6  {55574}  => {55573}  0.0001009434  0.4561404 4518.772
## 7  {44090}  => {33913}  0.0001475327  0.8636364 4277.823
## 8  {33913}  => {44090}  0.0001475327  0.7307692 4277.823
## 9  {66943}  => {37292}  0.0001397678  0.7826087 4113.807
## 10 {37292}  => {66943}  0.0001397678  0.7346939 4113.807
```

```
# filter by LHS to pull out specific product ids
rules_subset = subset(rules, (lhs %in% c('241')))
inspect(rules_subset)
```

```
##   lhs      rhs        support confidence     lift
## 1 {241} => {251} 0.0001087083  0.2800000 1567.817
## 2 {241} => {252} 0.0001281205  0.3300000 1214.259
## 3 {241} => {244} 0.0002989479  0.7700000 1101.827
## 4 {241,
##    251} => {244} 0.0001009434  0.9285714 1328.734
## 5 {241,
##    244} => {251} 0.0001009434  0.3376623 1890.689
## 6 {241,
##    252} => {244} 0.0001164732  0.9090909 1300.859
## 7 {241,
##    244} => {252} 0.0001164732  0.3896104 1433.599
```

```
# aggregate the rules over the product type
group_rules = aggregate(rules, itemInfo(bought)$group)
inspect(group_rules)
```

```
##    lhs        rhs
## 1  {}      => {Book}
## 2  {}      => {Music}
## 3  {}      => {DVD}
## 4  {}      => {Video}
## 5  {}      => {}
## 6  {Music} => {Book}
## 7  {Book}  => {Music}
## 8  {Book}  => {Video}
## 9  {Video} => {Book}
## 10 {Video} => {Music}
## 11 {Music} => {Video}
## 12 {}      => {Video}
## 13 {Video} => {}
## 14 {DVD}   => {Book}
## 15 {Book}  => {DVD}
## 16 {Video} => {DVD}
## 17 {DVD}   => {Video}
## 18 {Music} => {DVD}
## 19 {DVD}   => {Music}
## 20 {}      => {DVD}
## 21 {DVD}   => {}
```

```r
# quality
quality(rules) = cbind(quality(rules), coverage = coverage(rules))
```

Cool, now we have 2,601 rules to work with, but inspecting all of them manually sounds like a nightmare. Lucky for us there's a second package called *arulesViz* that gives us lots of great visualization support.

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

where:

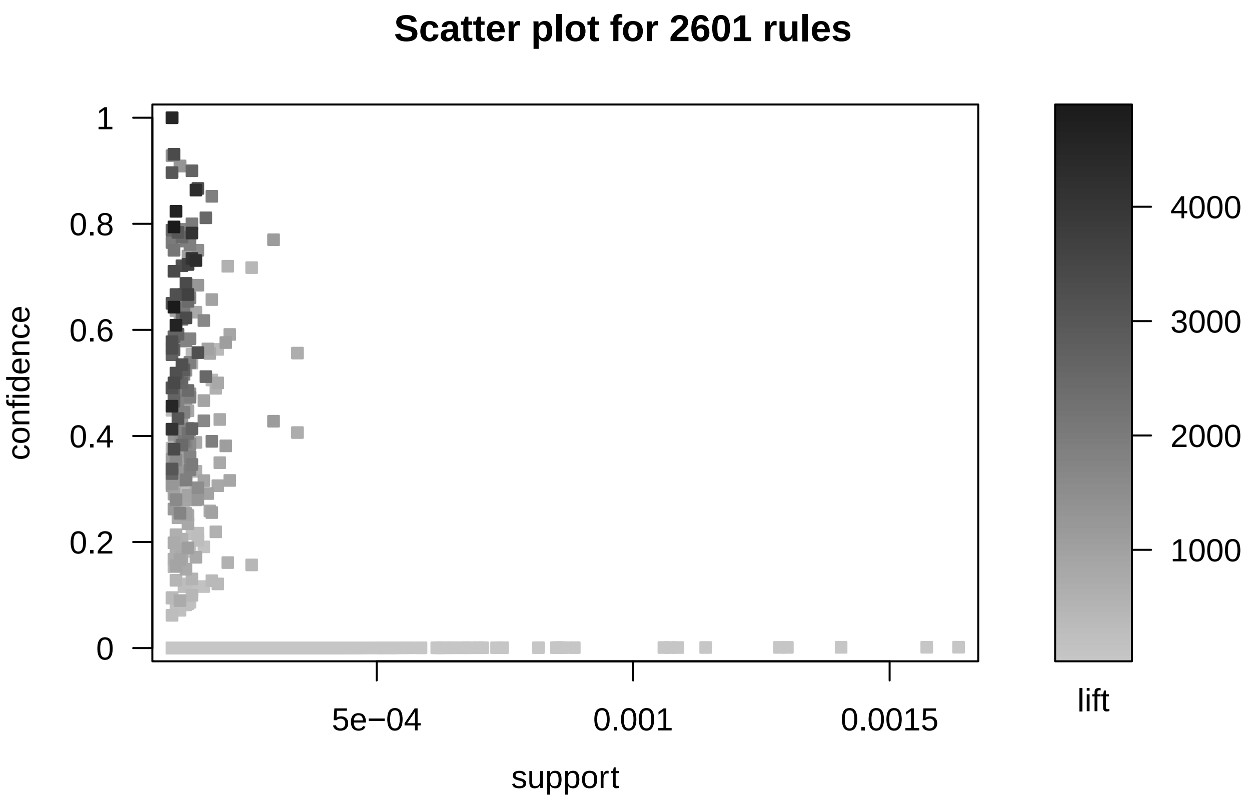
- *x*: is the set of rules to be visualized
- *method*: the visualization method
- *measure*: and shading contain the interest measures used by the plot
- *interactive*: indicates whether you want to interactively explore
- *data*: can contain the transaction data set used to mine the rules
- *control*: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

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

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

```r
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.

```r
plot(rules)
```



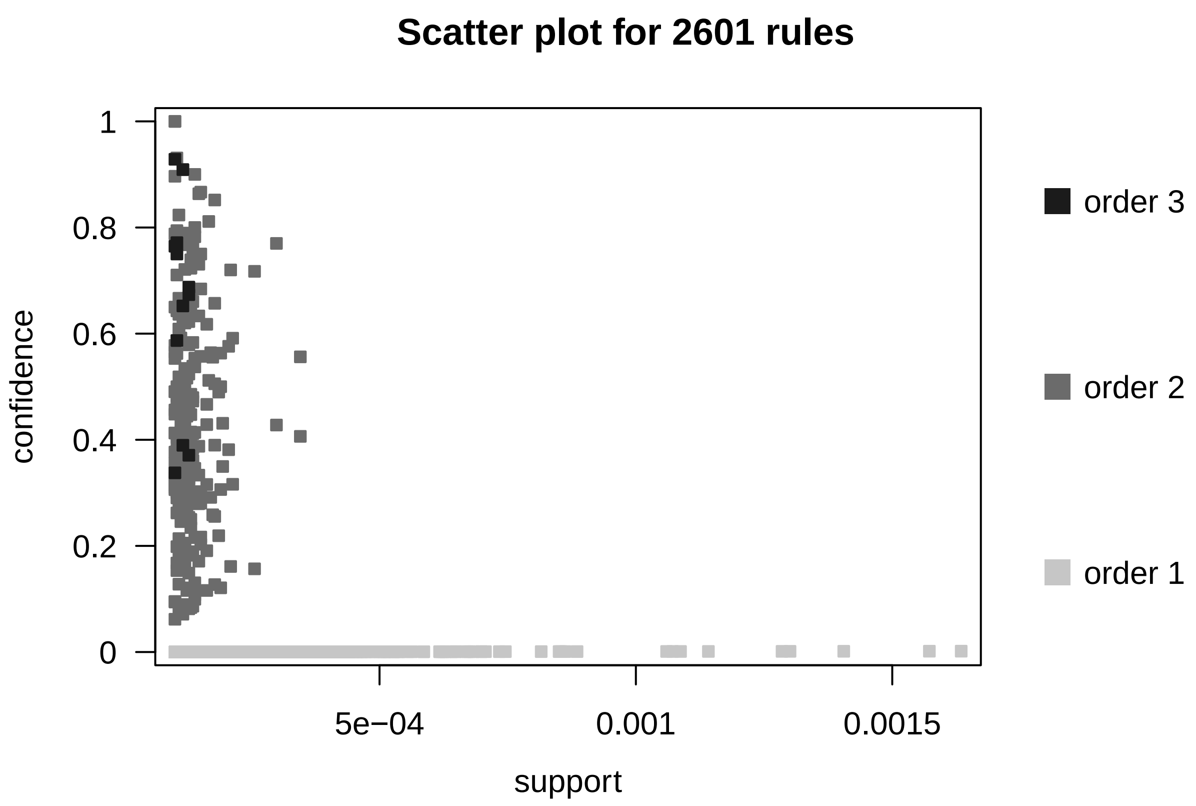
```r
head(quality(rules))
```

```
##        support   confidence lift coverage
## 1 0.0001048259 0.0001048259    1        1
## 2 0.0001048259 0.0001048259    1        1
## 3 0.0001009434 0.0001009434    1        1
## 4 0.0001009434 0.0001009434    1        1
## 5 0.0001009434 0.0001009434    1        1
## 6 0.0001009434 0.0001009434    1        1
```

```r
# there's also an interactive mode
# plot(rules, interactive = TRUE)
```

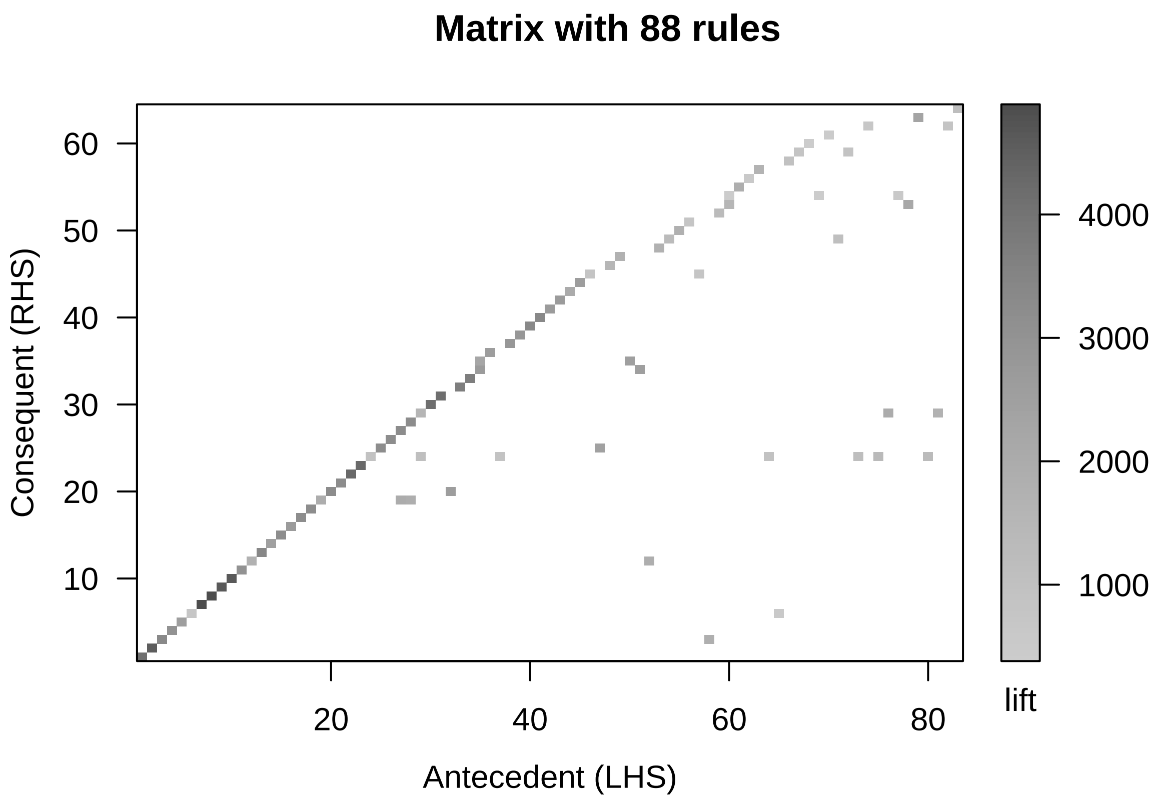
Another version of the scatter plot called two-key plot. Here support and confidence are used for the x and y-axes and the color of the points indicates the 'order,' *i.e.*, the number of items contained in the rule:

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



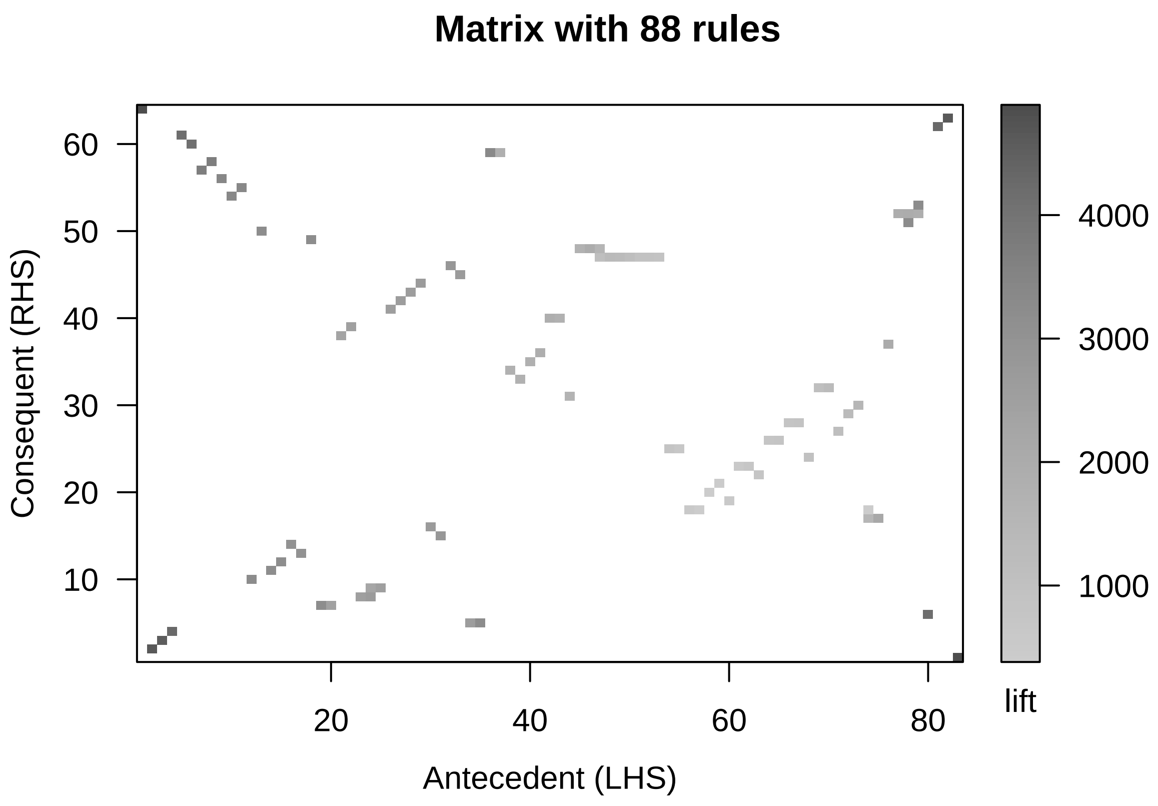
```
subrules = rules[quality(rules)$confidence > 0.5]
plot(subrules,
     method = 'matrix',
     measure = 'lift')
```

```
## Itemsets in Antecedent (LHS)
##  [1] "{83518}"     "{55573}"     "{58186}"     "{11039}"     "{12185}"
##  [6] "{16337}"     "{127682}"    "{134413}"    "{35413}"     "{35412}"
## [11] "{62848}"     "{82910}"     "{60540}"     "{7509}"      "{102492}"
## [16] "{43544}"     "{367}"       "{368}"       "{10709}"     "{12863}"
## [21] "{12862}"     "{44090}"     "{33913}"     "{250}"       "{85650}"
## [26] "{78197}"     "{9256}"      "{9334}"      "{251}"       "{66943}"
## [31] "{37292}"     "{7142}"      "{8551}"      "{8517}"      "{60311}"
## [36] "{91418}"     "{15944}"     "{63772}"     "{56317}"     "{8057}"
## [41] "{17160}"     "{47171}"     "{47170}"     "{95278}"     "{1498}"
## [46] "{13998}"     "{85648}"     "{6488}"      "{10897}"     "{60310}"
## [51] "{60312}"     "{95149}"     "{23146}"     "{9405}"      "{9864}"
## [56] "{4573}"      "{13997}"     "{43659}"     "{10419}"     "{94}"
## [61] "{6428}"      "{1673}"      "{4257}"      "{252}"       "{16338}"
## [66] "{20899}"     "{5915}"      "{4697}"      "{95}"        "{22073}"
## [71] "{9404}"      "{1439}"      "{241}"       "{1964}"      "{241,251}"
## [76] "{244,251}"   "{94,95}"     "{33,94}"     "{33,95}"     "{241,252}"
## [81] "{244,252}"   "{1964,2563}" "{1241,2563}"
## Itemsets in Consequent (RHS)
##  [1] "{83519}"  "{55574}"  "{58184}"  "{7447}"   "{7459}"   "{10519}"
##  [7] "{134413}" "{127682}" "{35412}"  "{35413}"  "{60468}"  "{82909}"
## [13] "{60561}"  "{7604}"   "{77634}"  "{43543}"  "{368}"    "{367}"
## [19] "{9332}"   "{12862}"  "{12863}"  "{33913}"  "{44090}"  "{244}"
## [25] "{78197}"  "{85650}"  "{9334}"   "{9256}"   "{241}"    "{37292}"
## [31] "{66943}"  "{8517}"   "{8551}"   "{60310}"  "{60312}"  "{83718}"
## [37] "{56317}"  "{63772}"  "{17160}"  "{8057}"   "{47170}"  "{47171}"
## [43] "{69483}"  "{1499}"   "{8939}"   "{1825}"   "{5936}"   "{36122}"
## [49] "{7153}"   "{5966}"   "{3661}"   "{3247}"   "{95}"     "{33}"
## [55] "{6427}"   "{481}"    "{4259}"   "{15934}"  "{5913}"   "{4429}"
## [61] "{14949}"  "{1241}"   "{94}"     "{1964}"
```



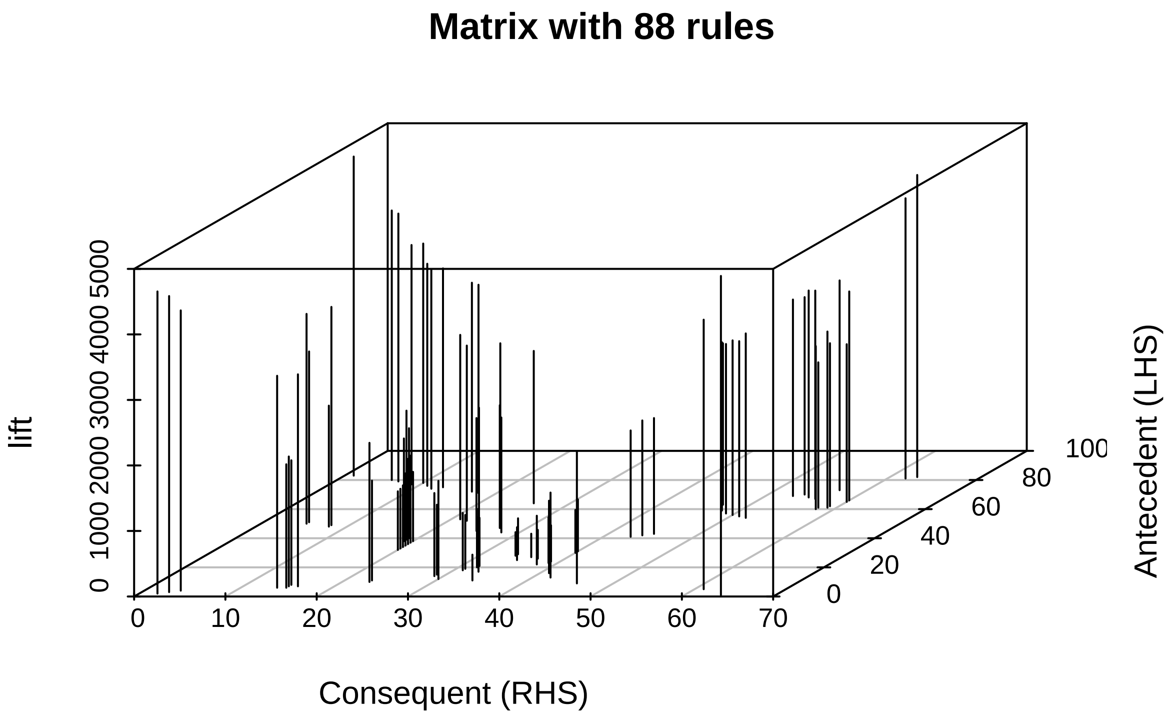
```
# reorder based on
plot(subrules,
     method = 'matrix',
     measure = 'lift',
     control = list(reorder = TRUE))
```

```
## Itemsets in Antecedent (LHS)
##  [1] "{127682}"    "{35413}"     "{55573}"     "{44090}"     "{37292}"
##  [6] "{83518}"     "{8551}"      "{8517}"      "{60540}"     "{8057}"
## [11] "{17160}"     "{12862}"     "{78197}"     "{368}"       "{102492}"
## [16] "{11039}"     "{62848}"     "{367}"       "{85650}"     "{85648}"
## [21] "{33,95}"     "{7509}"      "{60312}"     "{60311}"     "{60310}"
## [26] "{12185}"     "{1498}"      "{91418}"     "{43544}"     "{47170}"
## [31] "{63772}"     "{56317}"     "{47171}"     "{7142}"      "{12863}"
## [36] "{58186}"     "{43659}"     "{23146}"     "{10897}"     "{9864}"
## [41] "{6428}"      "{95149}"     "{82910}"     "{4257}"      "{244,252}"
## [46] "{244,251}"   "{251}"       "{241,251}"   "{241,252}"   "{241}"
## [51] "{252}"       "{250}"       "{15944}"     "{1964,2563}" "{1964}"
## [56] "{94,95}"     "{95}"        "{4697}"      "{22073}"     "{1673}"
## [61] "{16338}"     "{16337}"     "{4573}"      "{13997}"     "{13998}"
## [66] "{5915}"      "{1439}"      "{20899}"     "{9404}"      "{9405}"
## [71] "{1241,2563}" "{10419}"     "{6488}"      "{94}"        "{33,94}"
## [76] "{95278}"     "{10709}"     "{9256}"      "{9334}"      "{66943}"
## [81] "{33913}"     "{35412}"     "{134413}"
## Itemsets in Consequent (RHS)
##  [1] "{127682}" "{35412}"  "{55574}"  "{33913}"  "{12862}"  "{37292}"
##  [7] "{78197}"  "{60310}"  "{60312}"  "{12863}"  "{367}"    "{77634}"
## [13] "{60468}"  "{7447}"   "{56317}"  "{47171}"  "{95}"     "{33}"
## [19] "{481}"    "{4429}"   "{14949}"  "{3661}"   "{10519}"  "{15934}"
## [25] "{1241}"   "{8939}"   "{1964}"   "{5913}"   "{3247}"   "{1825}"
## [31] "{4259}"   "{7153}"   "{5936}"   "{36122}"  "{5966}"   "{6427}"
## [37] "{69483}"  "{94}"     "{7604}"   "{82909}"  "{7459}"   "{1499}"
## [43] "{83718}"  "{43543}"  "{47170}"  "{63772}"  "{244}"    "{241}"
## [49] "{368}"    "{85650}"  "{9334}"   "{9332}"   "{9256}"   "{17160}"
## [55] "{8057}"   "{60561}"  "{8517}"   "{8551}"   "{58184}"  "{83519}"
## [61] "{66943}"  "{44090}"  "{35413}"  "{134413}"
```

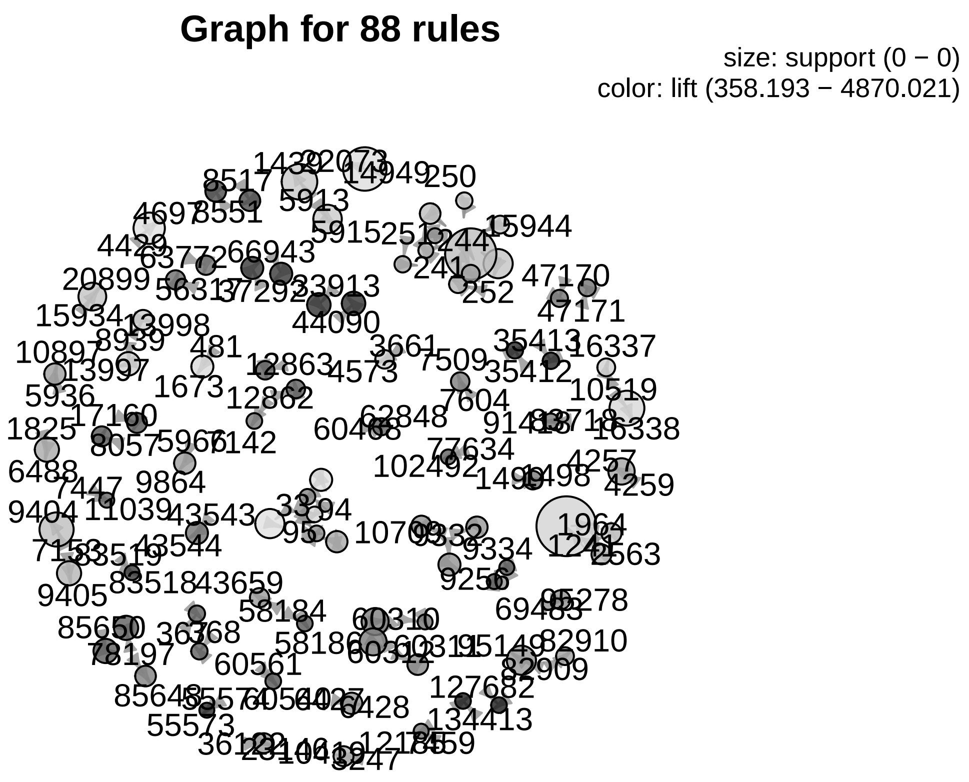


```
plot(subrules,
     method = 'matrix3D',
     measure = 'lift',
     control = list(reorder = TRUE))
```

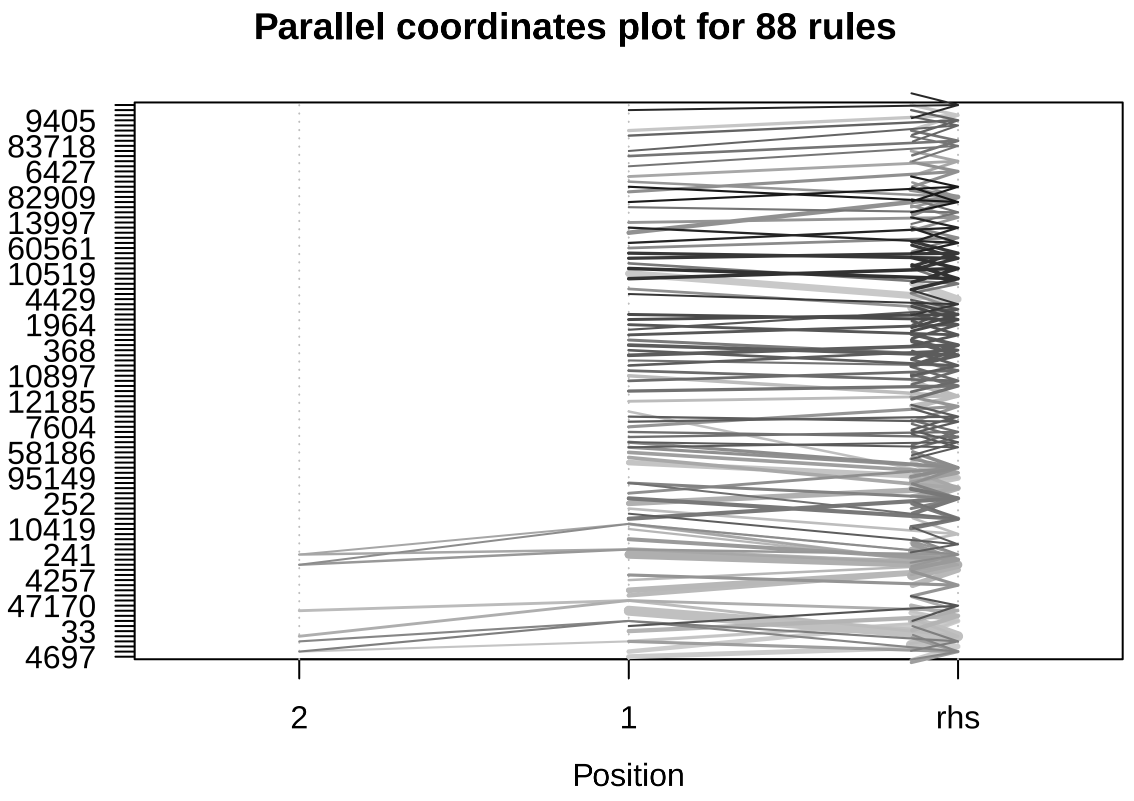
```
## Itemsets in Antecedent (LHS)
##  [1] "{127682}"    "{35412}"     "{55573}"     "{44090}"     "{66943}"
##  [6] "{9334}"      "{9256}"      "{10709}"     "{95278}"     "{33,94}"
## [11] "{94}"         "{6488}"       "{10419}"      "{9405}"       "{9404}"
## [16] "{1241,2563}" "{20899}"      "{1439}"       "{5915}"       "{13998}"
## [21] "{13997}"     "{4573}"       "{16337}"      "{16338}"      "{1673}"
## [26] "{22073}"     "{4697}"       "{95}"         "{94,95}"      "{1964}"
## [31] "{1964,2563}" "{15944}"      "{250}"        "{252}"        "{241}"
## [36] "{241,252}"   "{241,251}"    "{251}"        "{244,251}"    "{244,252}"
## [41] "{4257}"      "{10897}"      "{9864}"       "{23146}"      "{82910}"
## [46] "{95149}"     "{6428}"       "{43659}"      "{58186}"      "{12863}"
## [51] "{7142}"      "{47171}"      "{63772}"      "{56317}"      "{47170}"
## [56] "{43544}"     "{91418}"      "{1498}"       "{12185}"      "{60310}"
## [61] "{60311}"     "{60312}"      "{7509}"       "{33,95}"      "{85648}"
## [66] "{85650}"     "{367}"        "{102492}"     "{11039}"      "{62848}"
## [71] "{368}"       "{78197}"      "{12862}"      "{17160}"      "{8057}"
## [76] "{60540}"     "{8517}"       "{8551}"       "{83518}"      "{37292}"
## [81] "{33913}"     "{35413}"      "{134413}"
## Itemsets in Consequent (RHS)
##  [1] "{127682}" "{35413}"  "{55574}"  "{33913}"  "{12862}"  "{66943}"
##  [7] "{83519}"  "{58184}"  "{8551}"   "{8517}"   "{60561}"  "{8057}"
## [13] "{17160}"  "{9256}"   "{9332}"   "{9334}"   "{85650}"  "{367}"
## [19] "{241}"    "{244}"    "{56317}"  "{47170}"  "{95}"     "{1499}"
## [25] "{82909}"  "{94}"     "{6427}"   "{36122}"  "{7153}"   "{1825}"
## [31] "{5913}"   "{8939}"   "{15934}"  "{33}"     "{481}"    "{4429}"
## [37] "{14949}"  "{3661}"   "{10519}"  "{1241}"   "{1964}"   "{3247}"
## [43] "{4259}"   "{5936}"   "{5966}"   "{69483}"  "{7604}"   "{7459}"
## [49] "{83718}"  "{43543}"  "{47171}"  "{63772}"  "{7447}"   "{60468}"
## [55] "{77634}"  "{368}"    "{12863}"  "{60312}"  "{60310}"  "{78197}"
## [61] "{37292}"  "{44090}"  "{35412}"  "{134413}"
```



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



```r
# parallel coordinates plot
plot(subrules, method = 'paracoord', control = list(reorder = TRUE))
```


Parallel coordinates plot for 88 rules
rhs
Position