PDXDataSciRecommender

Charles Howard

November 14, 2017

Overview

The goal is to build a recommendation engine for games. The R package arules is used to mine associations between lists of items. The arulesViz package has plot methods to visualize relationships between items.

I started with the original set of 834415 rows and 3 columns. The arules package requires nominal variables be converted to factors and continuous variables to be discretized. I followed examples given in the following webpage: http://michael.hahsler.net/research/arules_RUG_2015/demo/

R code follows:

```r
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
library(arulesViz)
```

```
## Loading required package: grid
```

```r
#library(Matrix)  if needed
datdir<-"C:/Users/Charles/Documents/PDXDataSciRecommender/"
setwd(datdir)
dat<-read.csv(paste(datdir,"boardgame-ratings.csv",sep=""))
# sorting data by 1.) UserId, then 2.) gameID
dat<-dat[order(dat$UserID,dat$gameID),]
# determining groupings by UserID
usergrping<-grouping(dat$UserID)
userid.ends<-attr(usergrping,"ends")
userid.starts<-c(1,userid.ends[1:(length(userid.ends)-1)]+1)
userid.counts<-diff(userid.starts)
# convert to factors
dat[,"UserID"]<-factor(dat[,"UserID"])
dat[,"gameID"]<-factor(dat[,"gameID"])
# discretize ratings
dat[,"rating"]<-discretize(dat$rating,method="interval",categories=5)
# for first attempt, I create a list of gameID's by UserID
translist<-lapply(1:length(userid.ends),function(n){
  rws<-userid.starts[n]:userid.ends[n]
  x<-dat$gameID[rws]
})
# the transaction class is the primary one used for arules
datrans<-as(translist,"transactions")
```

Each list in translist is a “transaction”. For instance, the gameID’s for the first two UserID’s are given below.

```
## [1] "UserID 1 gameID's 13"     "UserID 1 gameID's 3076"
## [3] "UserID 1 gameID's 31260"  "UserID 1 gameID's 36218"
## [5] "UserID 1 gameID's 40692"  "UserID 1 gameID's 68448"
## [7] "UserID 1 gameID's 129622" "UserID 1 gameID's 148228"
```

```
##  [1] "UserID 2 gameID's 11"     "UserID 2 gameID's 13"
##  [3] "UserID 2 gameID's 2651"   "UserID 2 gameID's 14996"
##  [5] "UserID 2 gameID's 30549"  "UserID 2 gameID's 34635"
##  [7] "UserID 2 gameID's 40692"  "UserID 2 gameID's 68448"
##  [9] "UserID 2 gameID's 70323"  "UserID 2 gameID's 110327"
## [11] "UserID 2 gameID's 148228" "UserID 2 gameID's 178900"
```

Summary of the datrans transactions object.

```r
summary(datrans)
```

```
## transactions as itemMatrix in sparse format with
##  154655 rows (elements/itemsets/transactions) and
##  27 columns (items) and a density of 0.1998271
##
## most frequent items:
##      13     822   30549   36218   68448 (Other)
##   57284   57092   54279   47936   45617  572207
##
## element (itemset/transaction) length distribution:
## sizes
##     1     2     3     4     5     6     7     8     9    10    11    12
## 44648 14747 12740 10951  9544  8222 13791  5547  4890  4515  4059  3550
##    13    14    15    16    17    18    19    20    21    22    23
##  6081  2624  2136  1848  1601  1276  1539   235    92    17     2
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   4.000   5.395   8.000  23.000
##
## includes extended item information - examples:
##   labels
## 1     11
## 2     13
## 3    103
```

Some standard measures for item lists are support and confidence. Support is the proportion of a given item list in the data. Confidence is a conditional probability type measure. The confidence of item set A => item set B is: support(item set A) U support(item set B)/support(item set A)
I arbitrarily chose a target of 1000 to arrive at a support value.

```
## [1] "For a value of 1000 support is 0.00647. Computed as 1000/nrow(datrans)"
```

The apriori function takes the transaction object and creates itemlists based on parameters such as support, confidence et al. Below I have chosen frequent itemsets with a support as calculated above and a minimum length of three.

```r
itemsets <- apriori(datrans, parameter = list(target = "frequent",
                                              supp=sup, minlen = 3))
```

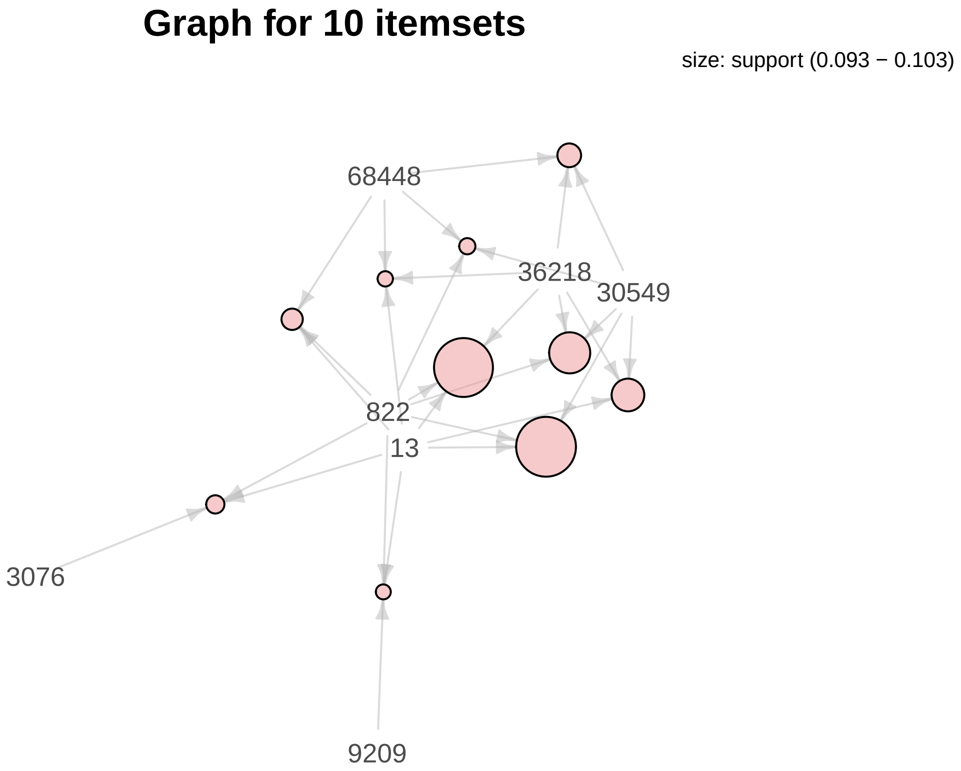
```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime     support
##          NA    0.1    1 none FALSE            TRUE       5 0.006466005
##  minlen maxlen            target   ext
##       3     10 frequent itemsets FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 1000
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[27 item(s), 154655 transaction(s)] done [0.05s].
## sorting and recoding items ... [26 item(s)] done [0.01s].
## creating transaction tree ... done [0.09s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 done [4.16s].
## writing ... [175304 set(s)] done [0.02s].
## creating S4 object  ... done [0.08s].
```

```
inspect(head(sort(itemsets), n=10))
```

```
##      items               support    count
## [1]  {13,822,30549}      0.10280948 15900
## [2]  {13,822,36218}      0.10259610 15867
## [3]  {822,30549,36218}   0.09851605 15236
## [4]  {13,30549,36218}    0.09659565 14939
## [5]  {30549,36218,68448} 0.09453946 14621
## [6]  {13,822,68448}      0.09398985 14536
## [7]  {13,822,3076}       0.09328505 14427
## [8]  {822,30549,68448}   0.09281304 14354
## [9]  {13,36218,68448}    0.09260612 14322
## [10] {13,822,9209}       0.09252853 14310
```

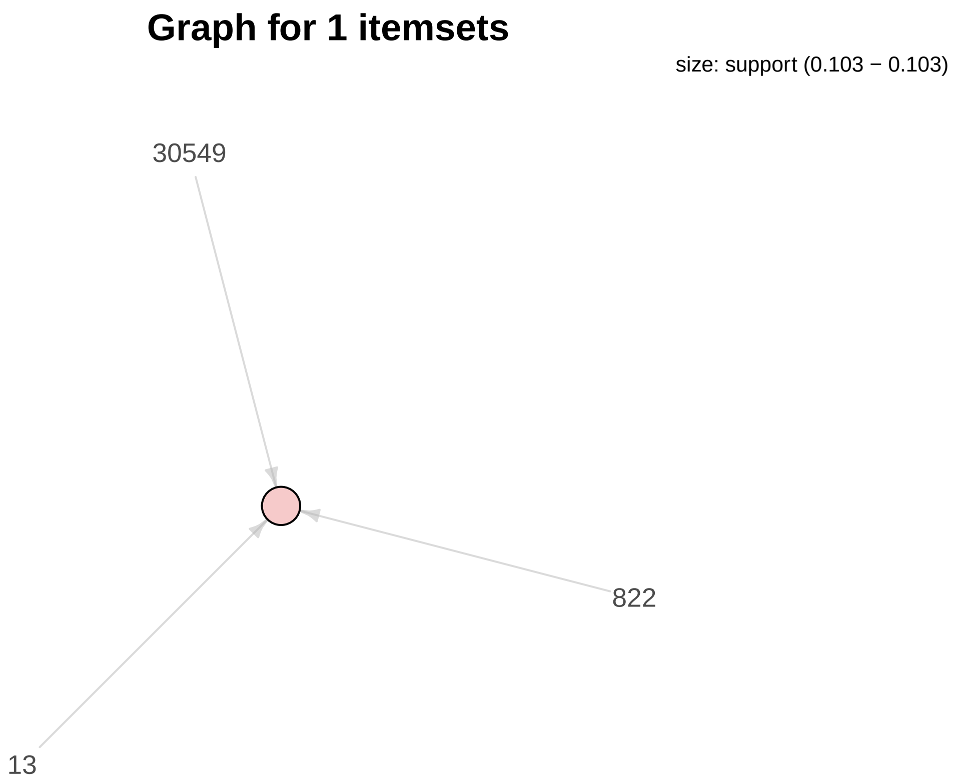
There is a really nice graph plot method in arulesViz. Following is the graph plot for the top 10 itemsets displayed above.

```
plot(head(sort(itemsets, by = "support"), n=10), method = "graph", control=list(cex=.8))
```



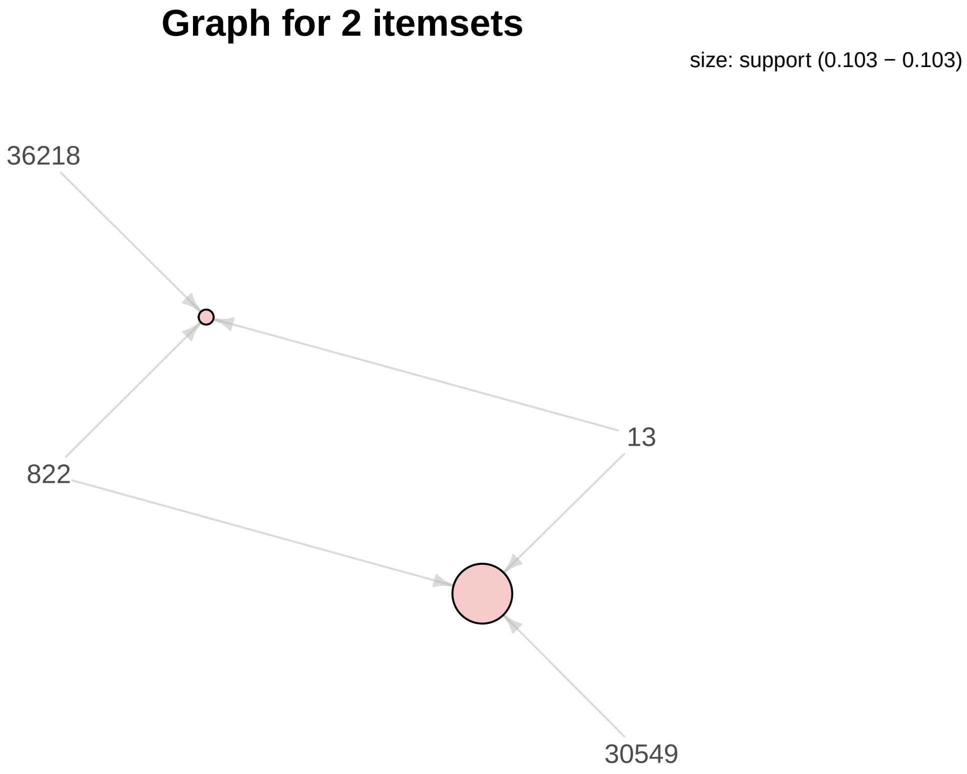
For extra clarity, some smaller plots. The first grouping in the table above.

```
plot(head(sort(itemsets, by = "support"), n=1), method = "graph", control=list(cex=.8))
```



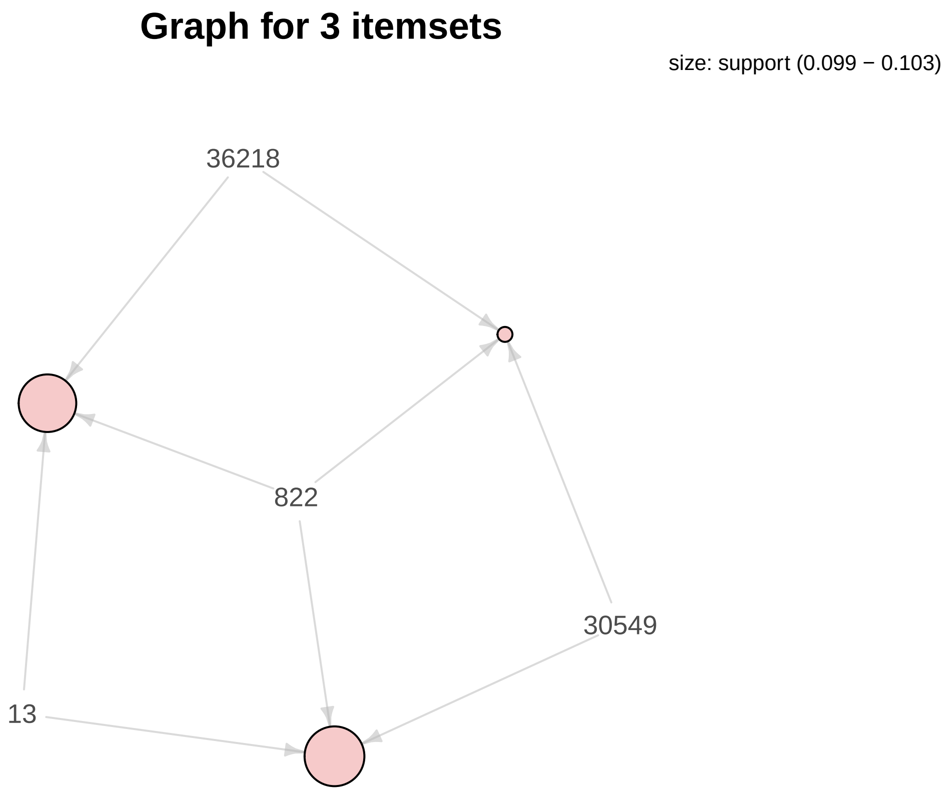
The first and second groupings in the table above.

```
plot(head(sort(itemsets, by = "support"), n=2), method = "graph", control=list(cex=.8))
```



. . . and so on. . .

```
plot(head(sort(itemsets, by = "support"), n=3), method = "graph", control=list(cex=.8))
```


Graph for 3 itemsets
size: support (0.099 − 0.103)
36218
822
30549
13