Transaction Mining with arules and arulesViz

Author: Chris Dunhill

16 November 2017

## Introduction

This is an experiment with transaction mining (presented using R Markdown). The inspiration for this analysis was the following article <http://www.salemmarafi.com/code/market-basket-analysis-with-r/>

## How did we do it?

The first challenge was to get the data into the correct format. I opted to do this work in SQL as I am slightly more familiar with this language, however it is likely to be possible in R, especially using packages such as *tidyr* and *dplyr*.

All R scripting was conducted using the open source RStudio IDE.

### R Packages Used

library(arules)  
library(arulesViz)  
library(datasets) # not necessary if using own data, I'm guessing?  
library(visNetwork) # a more fancy, animated version of arulesViz  
library(knitr) # used for changing the markdown parameters, I think

### Creating and loading the Data Set

The data was generated with the following SQL query, containing some Common Table Expressions or CTEs. [NB. managed to change to SQL syntax highlighting using *knit\_engines$set?* then commencing the following code box with ‘{sql eval=FALSE}’. eval=FALSE ensures that the code isn’t run].

knitr::knit\_engines$set("sql")

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;  
declare @startDate datetime,@endDate datetime;  
select @startDate=StartDate,@endDate=EndDate from Calendar where PeriodKey = 'L5WKS';  
  
with txqry as (  
 select   
 left (sh.saleshash,25) [TXID],  
 P.Category + '-' + P.Dept + '-' + P.SubDept [Prod],  
 RANK() OVER  
 (PARTITION BY left (sh.saleshash,25) ORDER BY P.Category + '-' + P.Dept + '-' + P.SubDept) [Item]   
 from SalesHis SH with (nolock)  
 left Join Products P with (nolock) on P.Ref = SH.Ref  
 left Join Branches B with (nolock) on B.Code = SH.Branch  
 where 1=1  
 and SH.Date between @startDate and @endDate  
 and sh.Qty > 0  
 and P.Dept not in ('CAR', 'DUM', 'NFS', 'PNM')  
 and P.Brand = 'MW'  
 and B.BranchCategory in ('HS','FOC')  
 group by  
 P.Category + '-' + P.Dept + '-' + P.SubDept, left (sh.saleshash,25)  
),  
numitems as  
(  
select TXID, max([Item]) [NumItems] from txqry group by TXID  
),  
morethan1 as  
(  
select t.TXID, t.Prod, t.Item, n.NumItems from txqry t inner join numitems n on t.txid = n.txid where numitems>1  
)  
  
-------------------------------------------------------------------  
  
-- Maybe there is a tidier way of eliminating those pesky NULLs... maybe could have done it in R  
  
select  
 isnull([1],'') [1],  
 isnull([2],'') [2],  
 isnull([3],'') [3],  
 isnull([4],'') [4],  
 isnull([5],'') [5],  
 isnull([6],'') [6],  
 isnull([7],'') [7],  
 isnull([8],'') [8],  
 isnull([9],'') [9],  
 isnull([10],'') [10]  
from morethan1  
pivot  
(  
 max([Prod]) for [Item] in ([1],[2],[3],[4],[5],[6],[7],[8],[9],[10])  
) piv  
  
option (recompile);

Ultimately it would be good to run this code directly in R using RODBC. However, for now, this has been generated in MSSQLSM and exported to a CSV file. Some cleaning was also done using Excel; it will be great to begin to use R for this purpose given there are countless tools for data cleansing and transformation.

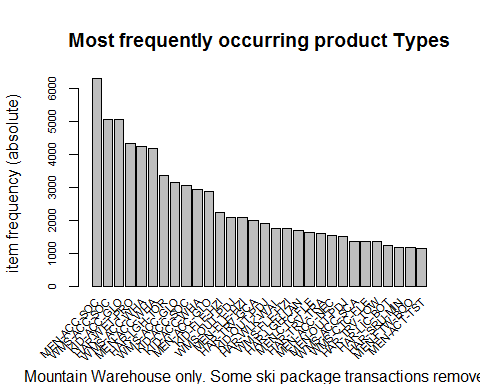
This is how we load the data set into R and view it

library(readr)  
mwlw = read.transactions("LW\_TX\_data excl SKI packages etc.csv", sep = ",")

### A basic plot

The data has been aggregated at Cat-Dept-Type level for this exercise. Let’s take a look at a basic freqency plot to see whether it looks about right, plotting the top 30:

# NB. the cex arguments are to reduce the font size for the X and Y axes, respectively  
itemFrequencyPlot(mwlw,topN=30,type="absolute", main = "Most frequently occurring product Types", sub = "Mountain Warehouse only. Some ski package transactions removed", cex.names = 0.7, cex.axis = 0.7)



### How to mine

For the purpose of this analysis we will use the *arules* R package which uses the methods of *association rules* mining. The way it works is to first establish the ‘rules’ before sorting them in order of frequency of occurrence.

**Some terminology**:

* SUPPORT is how many times the product combination appears in our transaction list. For example, a value of 0.01 would only show where the particular combination (‘LHS’, or antecedent) appears in >=1% transactions.
* CONFIDENCE is how often the rule is shown to be true
* LIFT is the ratio of the observed *support* to that expected if the LHS and RHS were independent

The *apriori* function is used to generate the rules for the data set. The *support* has been set to 0.002 (0.2%). There are about 51,000 transactions in our data so a support of 0.2% means that any rules with fewer than 102 transactions will be ignored. The *confidence* limit has been set to 0.5 (50%) so only rules which are proven to be true in more than half of the supported transactions will be shown. The *maxlen* parameter limits the number of basket items in the calculation [… I think!]

rules <- apriori(mwlw, parameter = list(supp = 0.0020, conf = 0.7,maxlen=3))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.7 0.1 1 none FALSE TRUE 5 0.002 1  
## maxlen target ext  
## 3 rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 102   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[239 item(s), 51282 transaction(s)] done [0.01s].  
## sorting and recoding items ... [152 item(s)] done [0.00s].  
## creating transaction tree ... done [0.02s].  
## checking subsets of size 1 2 3 done [0.00s].  
## writing ... [8 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

rules <- sort(rules, by="confidence", decreasing=TRUE)

The *rules* object cannot be viewed directly but we can look at the top results as shown:

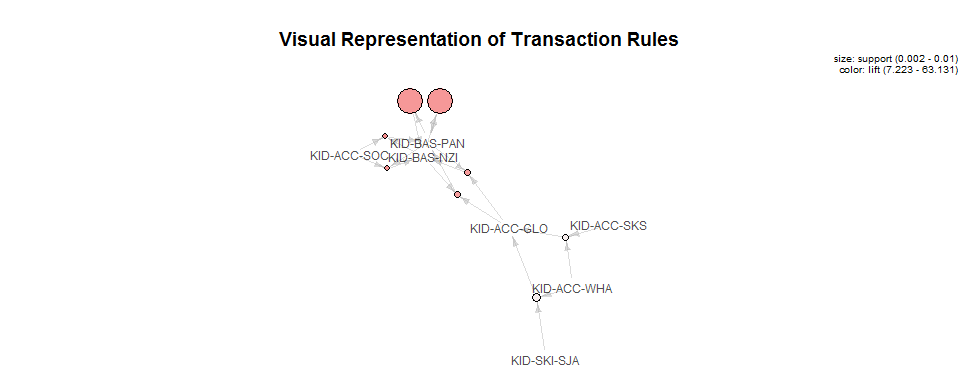
# Show the top 10 rules, but only to 2 digits  
options(digits=2) # Part of base package; digits controls no. sig digits when printing numeric values  
inspect(rules[1:5])

## lhs rhs support confidence lift  
## [1] {KID-ACC-SOC,KID-BAS-PAN} => {KID-BAS-NZI} 0.0020 0.85 61   
## [2] {KID-BAS-PAN} => {KID-BAS-NZI} 0.0105 0.82 59   
## [3] {KID-ACC-SOC,KID-BAS-NZI} => {KID-BAS-PAN} 0.0020 0.81 63   
## [4] {KID-ACC-GLO,KID-BAS-PAN} => {KID-BAS-NZI} 0.0021 0.79 57   
## [5] {KID-BAS-NZI} => {KID-BAS-PAN} 0.0105 0.76 59   
## count  
## [1] 104   
## [2] 538   
## [3] 104   
## [4] 106   
## [5] 538

### Visualisation

Let’s take a look at a visual representation of the rules, using the *arulesViz* package:

invisible(plot(rules, method = "graph", cex=0.75, layout=igraph::with\_fr(), main = "Visual Representation of Transaction Rules")) # cex to reduce font size



# Check other layout options  
# \*invisible\* function is to hide console results and only show the plot itself.