Transaction Mining with arules and arulesViz

Chris Dunhill, Mountain Warehouse

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/

What method was used?

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 and this RMarkdown document were created with the open-source RStudio IDE.

R Packages employed

All R packages need to be downloaded before loading with the library command.

```
library(arules)
library(arulesViz)
library(datasets)   # Only required if following the tutorial (link above)
library(visNetwork) # Alternative to arulesViz. Interactive graphs, better arranged
library(knitr)      # Used for creating this Markdown document
library(readr)      # Importing CSV
```

Creating and loading the Data Set

For this analysis to work, it is required to arrange data as one row per basket, with each column being a separate item in the basket. Here, I opted to aggregate products up to TYPE level as I expect it might produce better results compared to product (style / description) level. However, I've not tested this theory so this would be an obvious next step to take. The code would need to be adjusted.

The other limitation to this is that only the last 5 weeks' data have been queried. Web transactions also excluded here, however it would be fairly straightforward to adjust this code to include WWW. In this case, you would need to change the transaction ID TXID as the format is different for web orders compared to retail branches.

As mentioned, the merchandise is aggregated to TYPE level and represented in the MEN-ACC-SOC format - ie. CAT-DEPT-TYPE.

The SQL makes use of Common Table Expressions (CTEs) - especially useful as the PIVOT command is applied to the data which could otherwise become a bit complicated.

The SQL is run in SQL Server Manager and a CSV file is generated. There is an R package called RODBC which enables SQL querying directly from R. This could be useful for later versions of this analysis, though I couldn't get it to work this time for some reason.

A little tidying was also done using Excel due to my familiarity of that tool. For example, I excluded baskets which included ski package deals as these were distorting the results. However, in future these should be

included - with the components of the package perhaps aggregated into one 'product' (thanks to Andrew Steavenson for suggesting this!)

[nb. SQL syntax highlighting possible using *knit_engines$set?* command then commencing the following code box with '{sql eval=FALSE}' (hidden here). *eval=FALSE* ensures that the code is not 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 an easier way of eliminating the NULLs... perhaps 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);
```

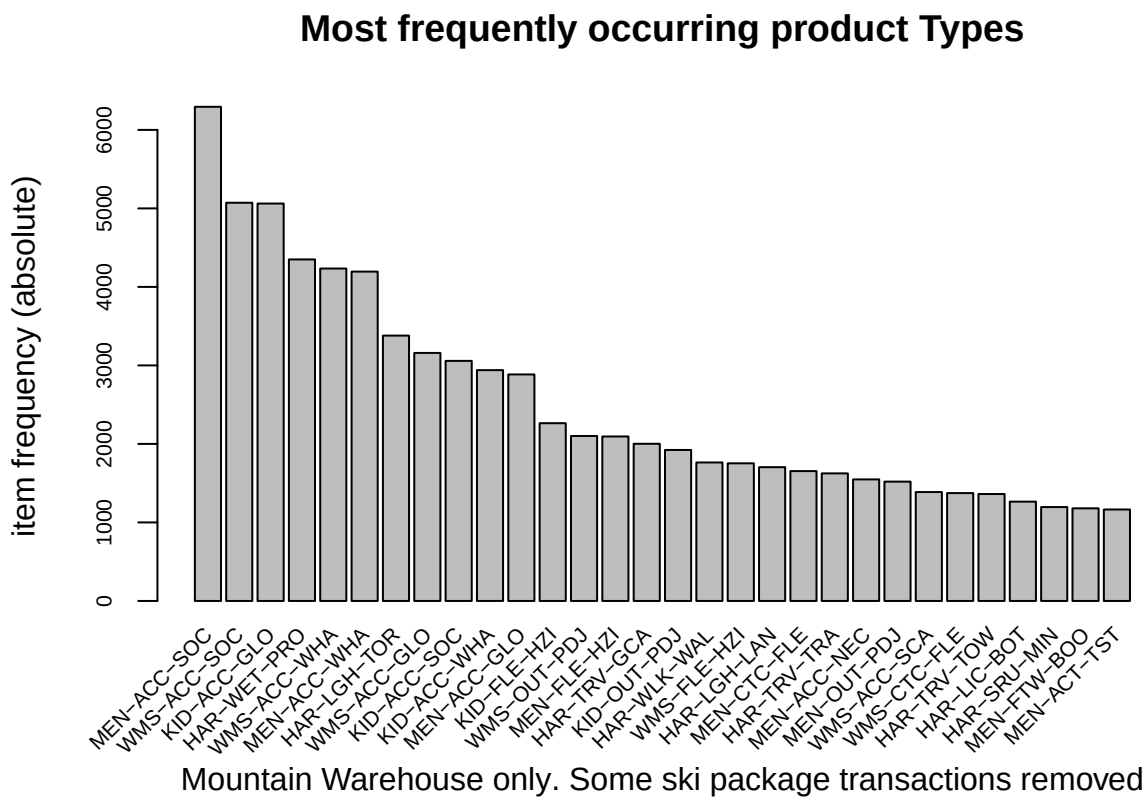
This is how we load the data set into R and view it:

```
mwlw = read.transactions("LW_TX_data excl SKI packages etc.csv", sep = ",")
```

Basic frequency plot

The data has been aggregated at Cat-Dept-Type level for this exercise. Let's take a look at a simple 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 this analysis works is to first establish the 'rules' before sorting them in order of frequency of occurrence.

Some terminology

- An example of an association RULE is *if a customer buys men's shoes, he is 70% likely to also purchase socks*.
- 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

Apriori

The *apriori* function from the arules package is used to generate the rules for the data set. The *support* has been set to 0.002 (0.2%). There are 51,282 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 [nb. need to check this is the case!]

```
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 line above sorts the *rules* object in order of highest *confidence*, however it may be useful at times to sort by *support* or *lift*.

The object cannot be viewed directly but we can inspect the top results as shown:

```
# Show the top 5 rules, but only to 2 digits
options(digits=2) # Base package: number of significant digits when printing numbers
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
```

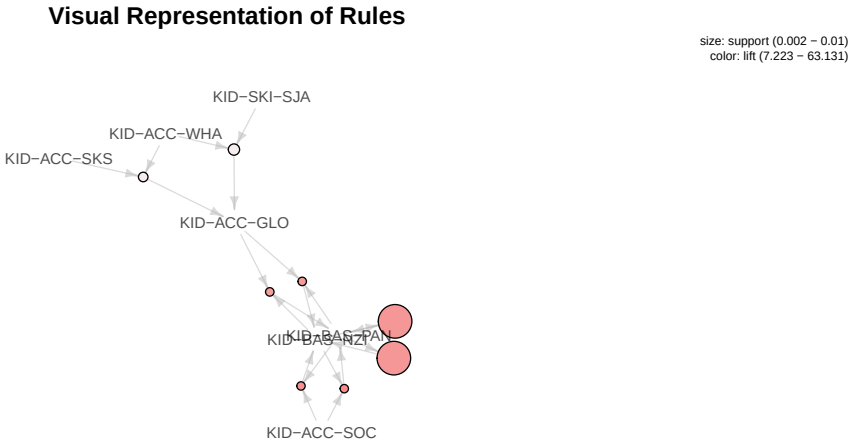
My interpretation of this is - taking the first line as an example - if a customer buys both kids socks (KID-ACC-SOC) and base-layer pants (KID-BAS-PAN) then they are 85% likely (*confidence*) to also buy no-zip baselayer tops (KID-BAS-NZI). However, the evidence for this comes from only 102 transactions: the *support* of 0.002 X 51,282 (total baskets).

I still need to better understand the *lift* parameter but think that this takes into account the 'chance' occurence of the two items being in the same basket.

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 Rules")) # cex to reduce font size
```




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

The graph above is admittedly not optimally laid out. The *visNetwork* package does a better job of it, however I struggled to get it to work with with the RMarkdown required to produce this document.

As I understand it, the red circles represent each of the top *rules* with the size showing the *support* and the 'brightness' of the red indicating the *lift*.