# Coursework7

*Fortunat Mutunda*

*March 23, 2016*

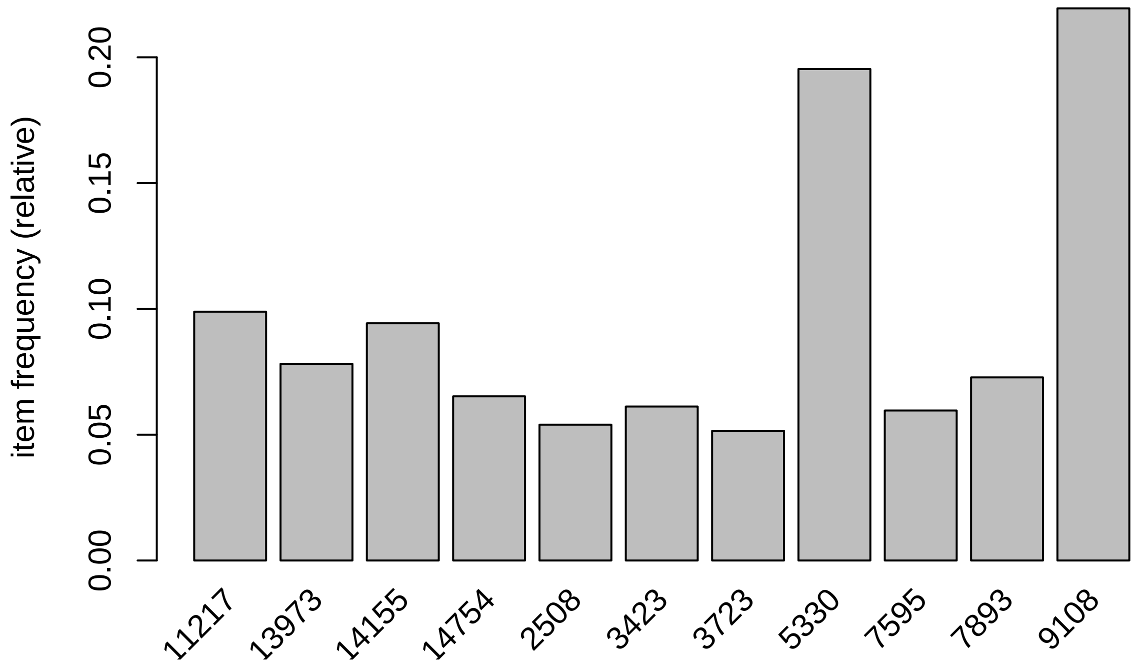
1. Report which tools you decided to use, how you used them, what were the first results. Also report the running times for the tools chosen.

***comments:*** *Mine rules from the supermarket txt* 1. Run whatever tool you run and plot the running time 2. Get support and confidence 3. Which tool did you use , how did you use it how long did it run. 4. Discribe in general… show examples, max, min,… etc. 5. How did you get the result what params etc…

For this first tast I have selected the *arules* library because that's the one I am used to, and it's the one I've been using for the couple past homeworks. It is not only easy to use but it is also very intuitive and pretty straight forward. With its nice inuilt function `read.transaction()` I can get all the transations without an hassle.

```
#df = read.csv("supermarket.txt", header = F, sep = " ")

#Starts from here
ptm <- proc.time()
trans = read.transactions("supermarket.txt", format = "basket" , sep = " ")
itemFrequencyPlot(trans, support = 0.05)
```

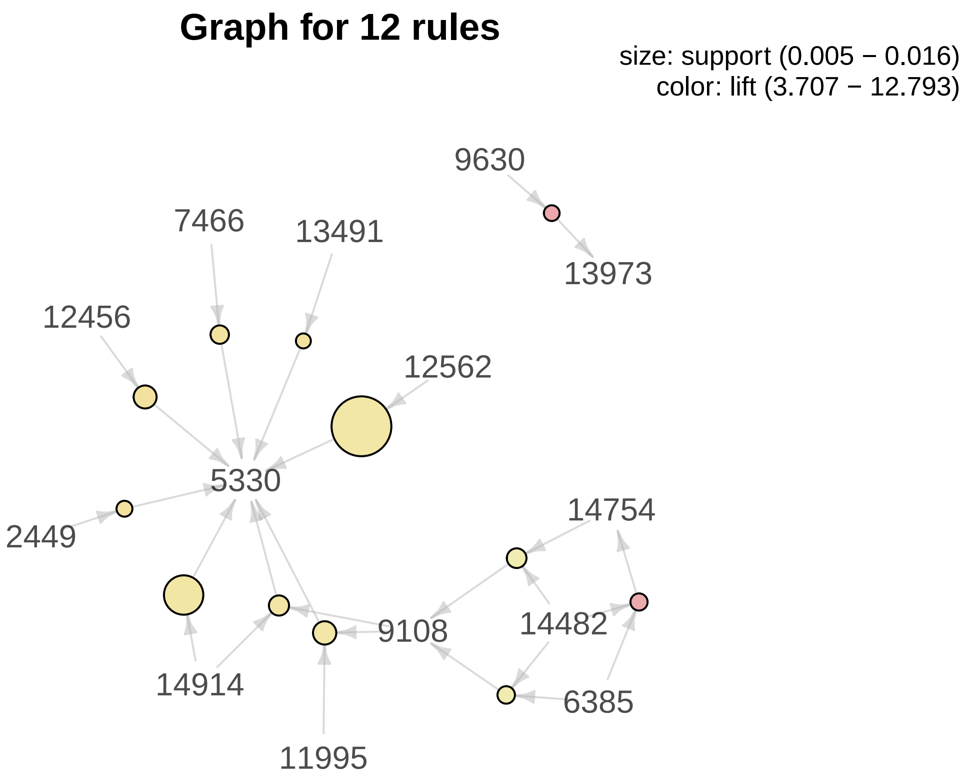


1. First I've read the transaction then I have proceeded by getting which items are the most frequents in the transaction. And it appears that `9108` is the most frequent item in the given itemset, when I use a support of 0.05.

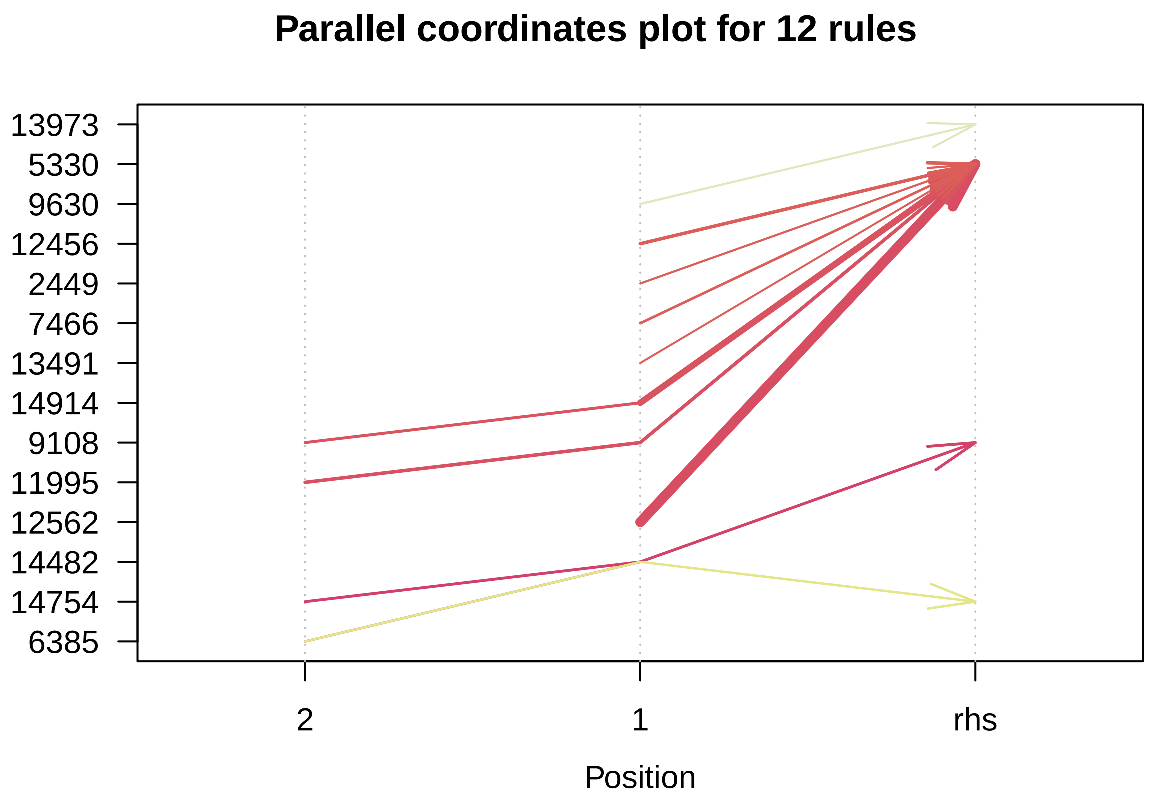
```
#rules
ruless <- apriori(trans, parameter= list(supp=0.005))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.8    0.1    1 none FALSE            TRUE   0.005      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: 126
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[15699 item(s), 25382 transaction(s)] done [0.03s].
## sorting and recoding items ... [186 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [12 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
plot(ruless, method = "graph")
```



```
plot(ruless, method="paracoord", control=list(reorder=TRUE))
```



- Second I have applied the `apriori` alogrithm on the given transaction with a support of 0.05. And the I have plotted two differents plots. *Notice* that whenever a client almost get any kind of item they also end up getting the item `5330` but bigger chances are mostly when `12562` is gotten then `5330` will be gotten too or when `14914` it is still the same case.

```
confidence  0.8   |   minval 0.1  | smax 1  | arem none | aval FALSE   | originalSupport
TRUE | support  0.005 | minlen 1 | maxlen 10
```

- This is the running time of the whole process.

```
proc.time() - ptm
```

```
##    user  system elapsed
##   5.665   0.270   6.162
```

`proc.time` returns five elements for backwards compatibility, but its print method prints a named vector of length 3. The first two entries are the total user and system CPU times of the current R process and any child processes on which it has waited, and the third entry is the real' elapsed time since the process was started.

2. Report overall 5 different high-support, high-confidence, high-lift rules; provide the respective contingency tables and scores. ***contengancy table*** if I can find a tool that get me a contengancy table. show the top 5 rules.

The code here might be a bit redundant but I am going to try to get first the top five 5 support

```r
ruless.sorted = sort(ruless,by = "support")
five_rules = head(ruless.sorted,5)
inspect(five_rules)
```

```
##    lhs              rhs    support     confidence lift
## 7  {12562}      => {5330} 0.016231975 0.8673684  4.440408
## 6  {14914}      => {5330} 0.011110236 0.8924051  4.568581
## 12 {11995,9108} => {5330} 0.007091640 0.8737864  4.473265
## 5  {12456}      => {5330} 0.007012844 1.0000000  5.119403
## 8  {14914,9108} => {5330} 0.006303680 0.9142857  4.680597
```

Then the top 5 hight confidence

```r
ruless.sorted = sort(ruless,by = "confidence")
five_rules = head(ruless.sorted,5)
inspect(five_rules)
```

```
##   lhs        rhs     support     confidence lift
## 1 {9630}  => {13973} 0.005200536 1          12.793347
## 2 {13491} => {5330}  0.005003546 1           5.119403
## 3 {7466}  => {5330}  0.005870302 1           5.119403
## 4 {2449}  => {5330}  0.005239934 1           5.119403
## 5 {12456} => {5330}  0.007012844 1           5.119403
```

Lastly the top 5 lift

```r
ruless.sorted = sort(ruless,by = "lift")
five_rules = head(ruless.sorted,5)
inspect(five_rules)
```

```
##   lhs             rhs     support     confidence lift
## 1 {9630}       => {13973} 0.005200536 1.0000000  12.793347
## 9 {14482,6385} => {14754} 0.005555118 0.8245614  12.638296
## 2 {13491}      => {5330}  0.005003546 1.0000000   5.119403
## 3 {7466}       => {5330}  0.005870302 1.0000000   5.119403
## 4 {2449}       => {5330}  0.005239934 1.0000000   5.119403
```

data("UCBAdmissions") data("PreSex") data(HairEyeColor) hec = structable(Eye ~ Sex + Hair, data = HairEyeColor)

3. Discuss whether some other scores studied last week or in the lecture slides would help identify "more interesting" and different rules? ***Use something from last homewok*** no confidence or lift or anything try other maesures

4. Given the ability to discover frequent itemsets and association rules, propose a strategy to use these tools to study different customer segments, shops, shopping times, or specific products. ***What can you do with the data?*** Just write something, what method toools you have learnt so far to study diff rules… Suggest a plan…etc. There is nothing specific, just be creative… In which part can you use frequent set item or item set mining

5. Select some relatively high-support high-confidence rule (A->B) and based on that example describe the conditional probabilities P(A|B) and P(B|A), as well as the Bayes rule.

*** Just learn what a conditional probalities ***

6. (Bonus 2p) Run Krimp on same data, provide commands and describe your findings and compare to Frequent Itemset Mining + Association rules. (link to Krimp documentation)

***Krimp***