Arules

Clear the working environment and set the working directory

1. install.packages("arules")

2. Read 'Transactions.csv' such that the arules package treats the input file as "transaction" data.

```
require(arules)

## Loading required package: arules

## Loading required package: Matrix

##
## Attaching package: 'arules'

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

trans = read.transactions(file="Transactions.csv",
                          rm.duplicates= FALSE, format="single",
                          sep=",",cols =c(1,2))
```

3. Explore and understand the data and items of transaction data

```
inspect(trans)

##    items                      transactionID
## 1  {Choclates,Marker,Pencil}  1001
## 2  {Choclates,Pencil}         1002
## 3  {Coke,Eraser,Pencil}       1003
## 4  {Choclates,Cookies,Pencil} 1004
## 5  {Marker}                   1005
## 6  {Marker,Pencil}            1006
## 7  {Choclates,Pencil}         1007
## 8  {Choclates,Cookies,Pencil} 1008
## 9  {Marker,Pencil}            1009
## 10 {Coke,Marker}              1010
```

Look at the transactions

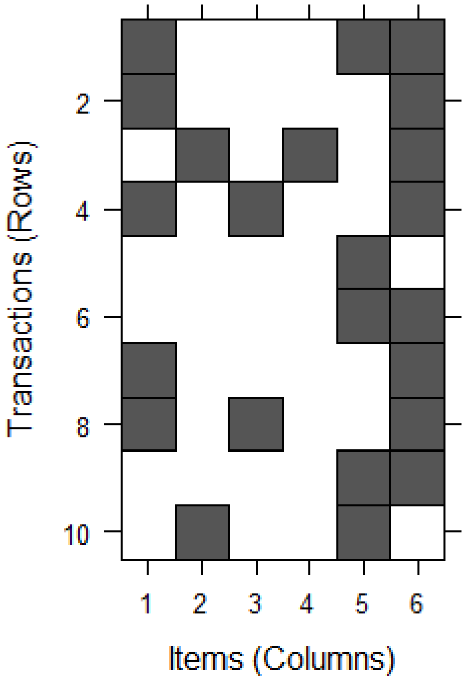
```
trans

## transactions in sparse format with
##  10 transactions (rows) and
##  6 items (columns)
```

Plotting & visualizing helps a lot in manual analysis and getting a basic idea of the data

Plot the transaction and view it. This works only for smaller datasets

```
image(trans)#details of items (1st item is Choclates)
```



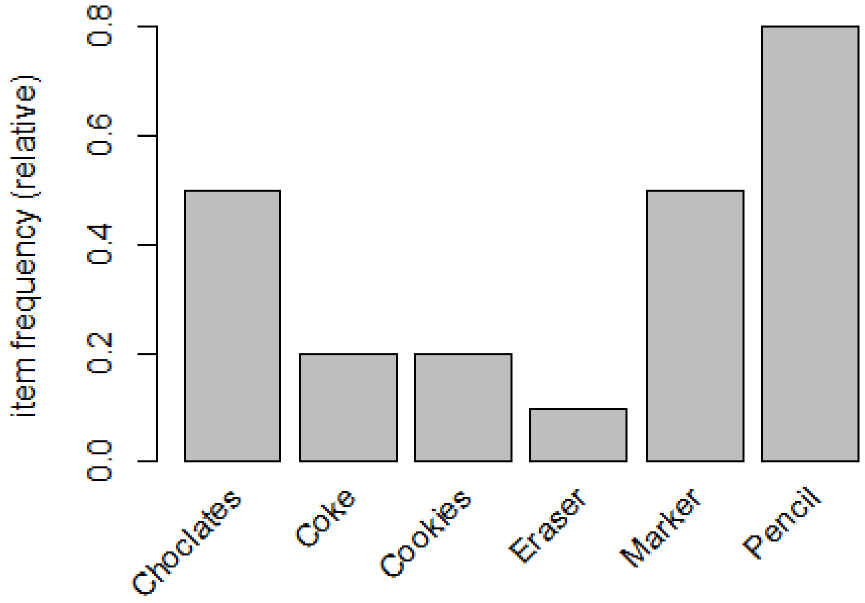
# Record numbers

```
itemFrequency(trans)

## Choclates      Coke    Cookies     Eraser     Marker     Pencil
##       0.5       0.2        0.2        0.1        0.5        0.8
```

Plot item frequency

```
itemFrequencyPlot(trans)
```



# 5. Implementing association mining using 'Apriori' algorithm to extract rules

```
rules <- apriori(trans,parameter = list(sup = 0.2, conf =
0.6,target="rules"))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.6    0.1    1 none FALSE            TRUE     0.2      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: 2
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[6 item(s), 10 transaction(s)] done [0.00s].
## sorting and recoding items ... [5 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [8 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

6. Understand the rules

```
summary(rules)

## set of 8 rules
##
## rule length distribution (lhs + rhs):sizes
## 1 2 3
## 1 5 2
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.125   2.250   3.000
##
## summary of quality measures:
##     support         confidence          lift
##  Min.   :0.2000   Min.   :0.6000   Min.   :0.750
##  1st Qu.:0.2000   1st Qu.:0.7562   1st Qu.:1.188
##  Median :0.2500   Median :1.0000   Median :1.250
##  Mean   :0.3625   Mean   :0.8781   Mean   :1.344
##  3rd Qu.:0.5000   3rd Qu.:1.0000   3rd Qu.:1.438
##  Max.   :0.8000   Max.   :1.0000   Max.   :2.000
##
## mining info:
##   data ntransactions support confidence
##  trans            10     0.2        0.6
```

Inspect them

```
inspect(rules)

##   lhs                    rhs         support confidence lift
## 1 {}                  => {Pencil}    0.8     0.800      1.00
## 2 {Cookies}           => {Choclates} 0.2     1.000      2.00
## 3 {Cookies}           => {Pencil}    0.2     1.000      1.25
## 4 {Marker}            => {Pencil}    0.3     0.600      0.75
## 5 {Choclates}         => {Pencil}    0.5     1.000      1.25
## 6 {Pencil}            => {Choclates} 0.5     0.625      1.25
## 7 {Choclates,Cookies} => {Pencil}    0.2     1.000      1.25
## 8 {Cookies,Pencil}    => {Choclates} 0.2     1.000      2.00
```

Print top 5 rules sorted by confidence and then support as a data.frame.

```
top_rules = sort(rules, by = c("confidence", "support"))
head(as(top_rules, "data.frame"), n=5)
```

```
##                              rules support confidence lift
## 5         {Choclates} => {Pencil}     0.5          1 1.25
## 2        {Cookies} => {Choclates}     0.2          1 2.00
## 3           {Cookies} => {Pencil}     0.2          1 1.25
## 7 {Choclates,Cookies} => {Pencil}     0.2          1 1.25
## 8 {Cookies,Pencil} => {Choclates}     0.2          1 2.00
```

Order the rules by decreasing confidence

```
rules_by_conf = rules[sort(rules, by = "confidence", order = TRUE)]
as(rules_by_conf,"data.frame")
```

```
##                              rules support confidence lift
## 2        {Cookies} => {Choclates}     0.2      1.000 2.00
## 3           {Cookies} => {Pencil}     0.2      1.000 1.25
## 5         {Choclates} => {Pencil}     0.5      1.000 1.25
## 7 {Choclates,Cookies} => {Pencil}     0.2      1.000 1.25
## 8 {Cookies,Pencil} => {Choclates}     0.2      1.000 2.00
## 1                  {} => {Pencil}     0.8      0.800 1.00
## 6         {Pencil} => {Choclates}     0.5      0.625 1.25
## 4            {Marker} => {Pencil}     0.3      0.600 0.75
```

Summary

Get familiar with 'Arules' library on how to generate association rules.

Read transaction data, bring it in a format and generate rule using apriory algorithm.

Inspect transactions/rules of interest with a certain support & confidence.