Data Mining 2: Modélisation statistique et apprentissage **G. Cohen**

TP 16 mars 2018

ETheme 6 Règles d'association

1 Première partie

Données Groceries (Epicerie)
Les données Groceries contiennent les données de vente d'une épicerie locale avec 9835 transactions et 169 items. Chaque ligne représente le contenu d'un ticket (passage à la caisse). La commande summary donne quelques statistiques de base du jeu de données. Par exemple, on note que l'ensemble de données est plutôt épart avec une densité juste supérieure à 2.6%, que "whole milk" est l'item le plus populaire et qu'en moyenne une transaction contient moins de 5 items.

1. Installation et chargement des packages. Le package *arulesViz* permet d'obtenir une visualisation graphique des règles (http://cran.r-project.org/web/packages/arulesViz/vignettes/arulesViz.pdf).

   ```
   # installation des packages arules et arulesViz
   install.packages("arules", dependencies = TRUE);
   install.packages("arulesViz", dependencies = TRUE);
   library(arules);
   library(arulesViz);
   ```

2. Exploration des données avant la création de règles

   ```
   data(Groceries);
   summary(Groceries);
   ```

   ce qui donne

   ```
   transactions as itemMatrix in sparse format with
    9835 rows (elements/itemsets/transactions) and
    169 columns (items) and a density of 0.02609146

   most frequent items:
   ```

```
      whole milk other vegetables       rolls/buns             soda
            2513             1903             1809             1715
          yogurt          (Other)
            1372            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
  17   18   19   20   21   22   23   24   26   27   28   29   32
  29   14   14    9   11    4    6    1    1    1    1    3    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   4.409   6.000  32.000

includes extended item information - examples:
       labels  level2           level1
1 frankfurter sausage meet and sausage
2     sausage sausage meet and sausage
3  liver loaf sausage meet and sausage

# Create an item frequency plot for the top 20 items
itemFrequencyPlot(Groceries,topN=20,type="absolute")
```

3. Génération de règles avec un support minimum de 0.001 une confiance de 0.8. On montre le top des 5 premières règles.

```
# Les règles
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))
# Montrer les 5 premières règles, mais seulement 2 digits
options(digits=2)
inspect(rules[1:5])
```

ce qui donne

```
  lhs                            rhs             support      confidence  lift
1 {liquor,red/blush wine} => {bottled beer}  0.0019       0.90        11.2
2 {curd,cereals}          => {whole milk}    0.0010       0.91        3.6
3 {yogurt,cereals}        => {whole milk}    0.0017       0.81        3.2
4 {butter,jam}            => {whole milk}    0.0010       0.83        3.3
5 {soups,bottled beer}    => {whole milk}    0.0011       0.92        3.6
```

Ce que l'on lit comme par exemple, Si quelqu'un achète des yogourts et des céréales, il y a 81% de chance qu'il achète du lait entier également.

4. Le premier problème que l'on peut voir ici c'est que les règles ne sont pas triées. On peut vouloir la règle la plus pertinente en premier. Par exemple celle qui est la plus probable, on peut alors facileemnt trier par confiance.

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

ce qui donne

```
  lhs                                                rhs            support  conf.
1 {rice,sugar}                                    => {whole milk}   0.0012   1
2 {canned fish,hygiene articles}                  => {whole milk}   0.0011   1
3 {root vegetables,butter,rice}                   => {whole milk}   0.0010   1
4 {root vegetables,whipped/sour cream,flour}      => {whole milk}   0.0017   1
5 {butter,soft cheese,domestic eggs}              => {whole milk}   0.0010   1
```

5. On remarque que la règle 4 est un peu trop longue. Si l'on veut seulement des règles plus concise on ajoutera le paramètre "maxlen" parameter à la fonction Apriori:

```
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8,maxlen=3))
```

6. Cibler les items. Maintenant que l'on a vu comment générer des règles, on va voir comment cibler certains items pour générer les règles. Il y a essentiellement deux types de cibles par lesquelle on est intéressé et ce que l'on va illustrer avec comme exemple le lait entier "whole milk":

   - Quels sont les items que les clients achèteront vraisemblablement avant d'acheter du lait entier ?
   - Quel sont les items achetés vraisemblablement par les clients après avoir acheté du beurre ?

   Ce qui veut dire que l'on agira soit sur la parite gauche soit la partie droite de la règle. Pour la première question cela donne :

```
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.08),
               appearance = list(default="lhs",rhs="whole milk"),
```

```
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])
```

ce qui donne

```
  lhs                                                rhs          supp.   conf.  l
1 {rice,sugar}                                    => {whole milk}  0.0012   1     3
2 {canned fish,hygiene articles}                  => {whole milk}  0.0011   1     3
3 {root vegetables,butter,rice}                   => {whole milk}  0.0010   1     3
4 {root vegetables,whipped/sour cream,flour}      => {whole milk}  0.0017   1     3
5 {butter,soft cheese, domestic eggs}             => {whole milk}  0.0010   1     3
```

pour la seconde question

```
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.15,minlen=2),
               appearance = list(default="rhs",lhs="whole milk"),
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])
```

ce qui donne

```
  lhs             rhs                support confidence lift
1 {whole milk} => {other vegetables}   0.075       0.29  1.5
2 {whole milk} => {rolls/buns}         0.057       0.22  1.2
3 {whole milk} => {yogurt}             0.056       0.22  1.6
4 {whole milk} => {root vegetables}    0.049       0.19  1.8
5 {whole milk} => {tropical fruit}     0.042       0.17  1.6
6 {whole milk} => {soda}               0.040       0.16  0.9
```

7. Visualisation. La dernière étape consiste à visualiser les données (http://cran.r-project.org/web/packages/arulesViz/vignettes/arulesViz.pdf).

```
library(arulesViz)
plot(rules,method="graph",interactive=TRUE,shading=NA)
```

2 Seconde partie

Données Titanic
On utilise ici la base d'apprentissage TITANIC (fichier titanic.txt) qui comporte trois attributs prédictifs Class (1ST, 2ND, 3RD, Crew), Age (Adult, Child) et Gender (Male, Female) ainsi qu'une classe à prédire Survivor (Yes, No) pour une population totale de 2201 individus.

1. Lire les données, puis supprimewr la colonne Name et regrouper l'âge en trois catégories : $\{Child, Adult, Unknown\}$ (Enfant,Adulte,Inconnu).

```
# Chargement des données titanic et mise en forme
titanic <- read.delim("G:\\gic_DM2_2014\\Etheme_6\\titanic.txt", dec=",");
str(titanic);
# Retrait de la colonne "Name" et regroupement de la colonne "Age"
titanic_ar <- titanic[,2:5];
titanic_ar$Age = as.character(titanic_ar$Age);
c_idx <- which(as.numeric(titanic_ar$Age) < 20);
a_idx <- which(as.numeric(titanic_ar$Age) >= 20);
na_idx <- which(is.na(titanic_ar$Age));
titanic_ar$Age[c_idx] <- "Child";
titanic_ar$Age[a_idx] <- "Adult";
titanic_ar$Age[na_idx] <- "Unknown";
# Convertir les attributs en facteur
titanic_ar$Age <- as.factor(titanic_ar$Age);
titanic_ar$Survived <- as.factor(titanic_ar$Survived);
```

2. Installation et chargement des packages. Le package *arulesViz* permet d'obtenir une visualisation graphique des règles (http://cran.r-project.org/web/packages/arulesViz/vignettes/arulesViz.pdf).

```
# installation des packages arules et arulesViz
install.packages("arules", dependencies = TRUE);
install.packages("arulesViz", dependencies = TRUE);
library(arules);
library(arulesViz);
```

3. Génération des règles selon l'agorithme A priori avec le paramétrage par défaut (http://cran.r-project.org/web/packages/arules/arules.pdf) et 'inspection'.

```
# Génération des règles
rules <- apriori(titanic_ar);
inspect(rules);
```

4. Génération des règles selon l'agorithme A priori avec longeur minimale 3 (minlen: nombre minimal d'items par item set) , support 0.1 et confiance 0.8 et 'inspection'.

```
# Génération des règles
rules <- apriori(titanic_ar, parameter = list(minlen = 3, support = 0.1, conf =
                 appearance = list(rhs = c("Survived=0", "Survived=1"), default=
inspect(rules);
```

5. Visauliser les règles.

```
# Plot the rules
plot(rules);
plot(rules, method="scatterplot");
plot(rules, method="graph", control=list(type = "items", alpha = 1));
plot(rules, method="paracoord", control=list(reorder=TRUE));
```

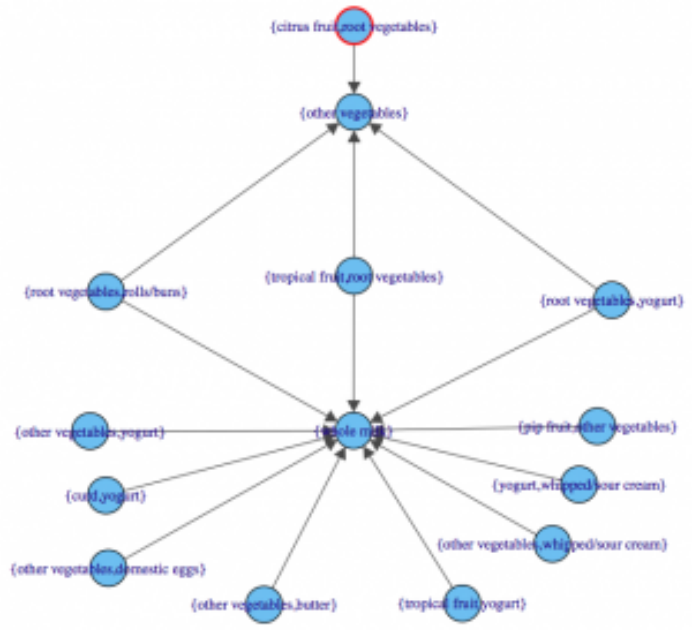

Figure 1: visualisation