Data Mining
Règles d'association

L'objectif de ce TP est de se familiariser avec l'utilisation des règles d'association dans `R`. Un compte-rendu de TP vous est demandé. Ce compte rendu doit contenir le code `R` utilisé et d'abondants commentaires.

1. Vous allez commencer par installer deux packages supplémentaires pour `R` : `arules` et `arulesViz`. Vous pouvez, par exemple, procéder comme dans le TP sur les arbres de décision. N'oubliez pas de charger les packages en utilisant la fonction `library(...)`.
2. Chargez ensuite les données `Groceries` en tapant `data(Groceries)`. De quelle classe est l'object `Groceries` ? Il s'agit de la classe utilisée pour représenter des données de transactions.
3. Pour inspecter les deux premières transactions, vous pouvez taper :
   `inspect(head(Groceries, 2))`.
4. Pour générer les règles avec un support relatif minimal de 1% et une confiance minimale de 0.5, vous pouvez procéder comme suit :
   ```
   rules <- apriori(Groceries,
                    parameter = list(support = 0.01, confidence = 0.5))
   ```
5. En lisant l'aide de la fonction `apriori(...)`, donnez un sens à la sortie obtenue. En particulier, que signifient `minval, smax, arem, aval, originalSupport, maxtime, minlen` et `maxlen` ?
6. Pour obtenir les trois règles ayant la plus grande confiance, vous pouvez procéder comme suit :
   ```
   inspect(head(sort(rules, by = "confidence"), 3))
   ```
   Listez les 5 règles ayant le plus grand *lift*.
7. Que se passe-t-il si le support minimal requis passe à 2% ? Que faut-il faire si l'on souhaite obtenir des règles avec plus d'*items* ou si l'on souhaite éliminer les règles les plus courtes ?
8. **Remarque :** pour voir les *items* les plus fréquents, on peut par exemple procéder comme suit :
   ```
   frequentItems <- eclat(Groceries,
                          parameter = list(supp = 0.07, maxlen = 15))
   inspect(frequentItems)
   ```
9. Parfois, il est pertinent de supprimer les règles qui sont des "sous-ensembles" de règles plus longues. Expliquez ce que fait le code suivant :
   ```
   rules <- apriori(Groceries, parameter = list(supp = 0.001,
                                                conf = 0.5, maxlen=3))
   subsetRules <- which(colSums(is.subset(rules, rules)) > 1)
   length(subsetRules)
   rules <- rules[-subsetRules]
   ```

10. Parfois, on souhaite se concentrer sur un item particulier dans la partie “conséquent” des règles. Expliquez ce que fait le code suivant :

```
rules <- apriori (data = Groceries, parameter = list(supp = 0.001,conf = 0.08),
                  appearance = list(default = "lhs", rhs = "whole milk"),
                  control = list(verbose = FALSE))
rules.conf <- sort (rules, by = "confidence", decreasing = TRUE)
inspect(head(rules.conf))
```

11. Il est également possible de trouver les “produits achetés en même temps qu’un autre produit”. Expliquez ce que fait le code suivant :

```
rules <- apriori (data = Groceries, parameter = list(supp = 0.001, conf = 0.15,
                                                     minlen=2),
                  appearance = list(default = "rhs",lhs = "whole milk"),
                  control = list(verbose = FALSE))
rules.conf <- sort (rules, by = "confidence", decreasing = TRUE)
inspect(head(rules.conf))
```

12. Enfin, essayez ligne par ligne le code suivant et expliquez ce qu’il fait :

```
rules <- apriori (data = Groceries, parameter = list(supp = 0.01, conf = 0.5),
                  appearance = list(default = "lhs" , rhs = "whole milk"),
                  control = list(verbose=FALSE))
plot(rules)
plot(rules, method = "graph", control = list(type = "items"))
plot(rules, method = "paracoord", control = list(reorder = TRUE))
```