Feuille de Travaux Dirigés n° 1
Règles d'association 2

1 Exemple 3

```
> setwd("C:/Données/ESIEA/Module5A/DataMining_2011-2012/TDs_DM")
```

In this example, we show how easy it is to add a new interest measure, using all-confidence as introduced by Omiecinski (2003). The all-confidence of an itemset X is defined as

$$\text{all-confidence}(X) = \frac{\text{supp}(X)}{\max_{I \subset X} \text{supp}(I)} \tag{1}$$

This measure has the property $\text{conf}(I \Rightarrow X \setminus I) \geqslant \text{all-confidence}(X)$ for all $I \subset X$. This means that all possible rules generated from itemset X must at least have a confidence given by the itemset's all-confidence value. Omiecinski (2003) shows that the support in the denominator of Equation 1 must stem from a single item and thus can be simplified to $\max_{i \in X} \text{supp}(\{i\})$. To obtain an itemset to calculate all-confidence for, we mine frequent itemsets from the previously used `Adult` data set using the Eclat algorithm.

```
> library(arules)
> data("Adult")
> fsets <- eclat(Adult, parameter = list(support = 0.05),
+     control = list(verbose = FALSE))
```

For the denominator of all-confidence we need to find all mined single items and their corresponding support values. In the following we create a named vector where the names are the column numbers of the items and the values are their support.

```
> singleItems <- fsets[size(items(fsets)) == 1]
> singleSupport <- quality(singleItems)$support
> names(singleSupport) <- unlist(LIST(items(singleItems),
+     decode = FALSE))
> head(singleSupport, n = 5)

       66        63       111        60         8
0.9532779 0.9173867 0.8974243 0.8550428 0.6941976
```

Next, we can calculate the all-confidence using Equation 1 for all itemsets. The single item support needed for the denomination is looked up from the named vector singleSupport and the resulting measure is added to the set's quality data frame.

```
> itemsetList <- LIST(items(fsets), decode = FALSE)
> allConfidence <- quality(fsets)$support/sapply(itemsetList,
+     function(x) max(singleSupport[as.character(x)]))
> quality(fsets) <- cbind(quality(fsets), allConfidence)
```

The new quality measure is now part of the set of itemsets.

```
> summary(fsets)

set of 8496 itemsets

most frequent items:
           capital-loss=None native-country=United-States
                        4082                          3973
           capital-gain=None                    race=White
                        3962                          3781
           workclass=Private                       (Other)
                        3142                         21931

element (itemset/transaction) length distribution:sizes
   1    2    3    4    5    6    7    8    9   10
  36  303 1078 2103 2388 1689  706  171   21    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   4.000   5.000   4.811   6.000  10.000

summary of quality measures:
    support        allConfidence
 Min.   :0.05002   Min.   :0.05247
 1st Qu.:0.06038   1st Qu.:0.06597
 Median :0.07546   Median :0.08428
 Mean   :0.10124   Mean   :0.11667
 3rd Qu.:0.11279   3rd Qu.:0.12711
 Max.   :0.95328   Max.   :1.00000

includes transaction ID lists: FALSE

mining info:
  data ntransactions support
 Adult         48842    0.05
```

It can be used to manipulate the set. For example, we can look at the itemsets which contain an item related to education (using partial match with %pin%) and sort them by all-confidence (we filter itemsets of length 1 first, since they have per definition an all-confidence of 1).

```
> fsetsEducation <- subset(fsets, subset = items %pin%
+     "education")
> inspect(sort(fsetsEducation[size(fsetsEducation) >
+     1], by = "allConfidence")[1:3])

  items                          support allConfidence
1 {education=HS-grad,
   hours-per-week=Full-time} 0.2090209     0.3572453
2 {education=HS-grad,
   income=small}             0.1807051     0.3570388
3 {workclass=Private,
   education=HS-grad}        0.2391794     0.3445408
```

The resulting itemsets show that the item high school graduate (but no higher education) is highly associated with working full-time, a small income and working in the private sector. All-confidence is along with many other measures of interestingness already implemented in arules as the function `interestMeasure`.

2 Exemple 4

In this example, we show how sampling can be used in `arules`. We use again the `Adult` data set.

```
> data("Adult")
> Adult

transactions in sparse format with
 48842 transactions (rows) and
 115 items (columns)
```

To calculate a reasonable sample size n, we use the formula developed by Zaki et al. (1997a). We choose a minimum support of 5%. As an acceptable error rate for support ε we choose 10% and as the confidence level (1-c) we choose 90%.

```
> supp <- 0.05
> epsilon <- 0.1
> c <- 0.1
> n <- -2 * log(c)/(supp * epsilon^2)
> n

[1] 9210.34
```

The resulting sample size is considerably smaller than the size of the original database. With sample() we produce a sample of size n with replacement from the database.

```
> AdultSample <- sample(Adult, n, replace = TRUE)
```

The sample can be compared with the database (the population) using an item frequency plot. The item frequencies in the sample are displayed as bars and the item frequencies in the original database are represented by the line. For better readability of the labels, we only display frequent items in the plot and reduce the label size with the parameter cex.names. The plot is shown in Figure reffig7.

```
> itemFrequencyPlot(AdultSample, population = Adult,
+     support = supp, cex.names = 0.7)
```

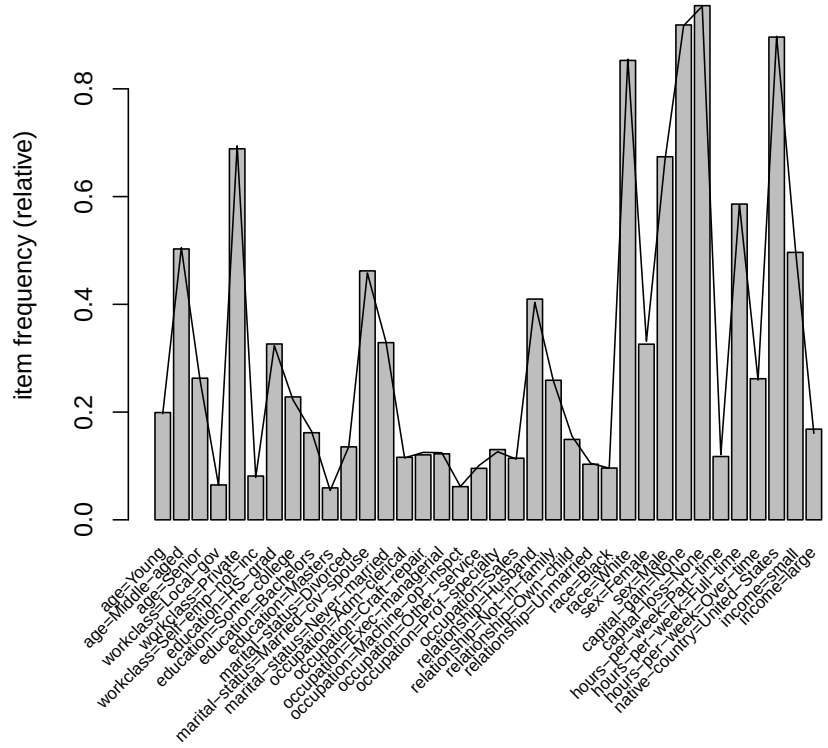

FIGURE 1 – Item frequencies in a sample of the Adult data set (bars) compared to the complete data set (line).

Alternatively, a sample can be compared with the population using the lift ratio (with lift = TRUE). The lift ratio for each item i is $\mathbb{P}(i|sample)/\mathbb{P}(i|population)$ where the probabilities are estimated by the item frequencies. A lift ratio of one indicates that the items occur in the sample in the same proportion as in the population. A lift ratio greater than one indicates that the item is over-represented in the sample and vice versa. With this plot, large relative deviations for less frequent items can be identified visually (see Figure 2).

```
> itemFrequencyPlot(AdultSample, population = Adult,
+     support = supp, lift = TRUE, cex.names = 0.9)
```

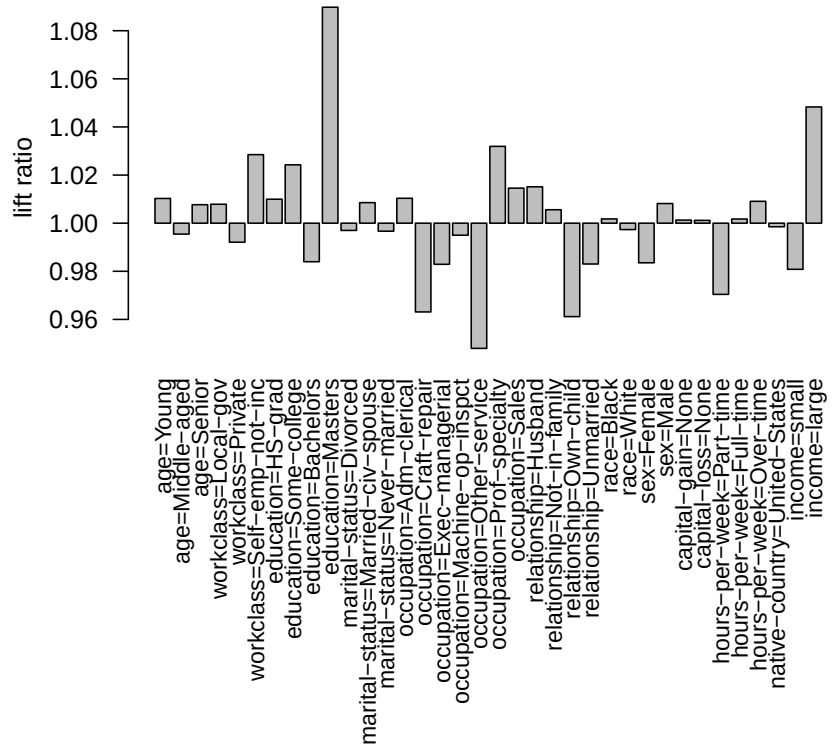

FIGURE 2 – Deviations of the item frequencies in the sample from the complete Adult data set.

To compare the speed-up reached by sampling we use the Eclat algorithm to mine frequent itemsets on both, the database and the sample and compare the system time (in seconds) used for mining.

```
> time <- system.time(itemsets <- eclat(Adult, parameter = list(support = supp),
+     control = list(verbose = FALSE)))
> time

   user  system elapsed
   0.48    0.00    0.48

> timeSample <- system.time(itemsetsSample <- eclat(AdultSample,
+     parameter = list(support = supp), control = list(verbose = FALSE)))
> timeSample
```

```
   user  system elapsed
   0.11    0.00    0.11
```

The first element of the vector returned by system.time() gives the (user) CPU time needed for the execution of the statement in its argument. Therefore, mining the sample instead of the whole data base results in a speed-up factor of :

```
> time[1]/timeSample[1]

user.self
 4.363636
```

To evaluate the accuracy for the itemsets mined from the sample, we analyze the difference between the two sets.

```
> itemsets

set of 8496 itemsets

> itemsetsSample

set of 8539 itemsets
```

The two sets have roughly the same size. To check if the sets contain similar itemsets, we match the sets and see what fraction of frequent itemsets found in the database were also found in the sample.

```
> match <- match(itemsets, itemsetsSample, nomatch = 0)
> sum(match > 0)/length(itemsets)

[1] 0.9724576
```

Almost all frequent itemsets were found using the sample. The summaries of the support of the frequent itemsets which were not found in the sample and the itemsets which were frequent in the sample although they were infrequent in the database give :

```
> summary(quality(itemsets[which(!match)])$support)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.05002 0.05056 0.05108 0.05141 0.05213 0.05534

> summary(quality(itemsetsSample[-match])$support)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.05005 0.05038 0.05114 0.05144 0.05233 0.05505
```

This shows that only itemsets with support very close to the minimum support were falsely missed or found. For the frequent itemsets which were found in the database and in the sample, we can calculate accuracy from the the error rate.

```
> supportItemsets <- quality(itemsets[which(match >
+     0)])$support
> supportSample <- quality(itemsetsSample[match])$support
> accuracy <- 1 - abs(supportSample - supportItemsets)/supportItemsets
> summary(accuracy)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8606  0.9635  0.9790  0.9739  0.9894  1.0000
```

The summary shows that sampling resulted in finding the support of itemsets with high accuracy. This small example illustrates that for extremely large databases or for application where mining time is important, sampling can be a powerful technique.

Table des matières

Références

[1] Omiecinski ER (2003). Alternative Interest Measures for Mining Associations in Databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(1), 57-69.

[2] Zaki MJ, Parthasarathy S, Li W, Ogihara M (1997a). Evaluation of Sampling for Data Mining of Association Rules. *In Proceedings of the 7th InternationalWorkshop on Research Issues in Data Engineering (RIDE '97) High Performance Database Management for Large-Scale Applications*, pp. 42-50. IEEE Computer Society.