# Data structures for association rules– implementation in the R package **arules**

Bettina Grün

August 16, 2004

## 1 Introduction

Mining of frequent itemsets or association rules is a popular method for discovering interesting relations between variables in large databases. Apriori and Eclat (Agrawal, Imielinski, and Swami, 1993; Agrawal and Srikant, 1994) are two algorithms for fast evaluation of frequent itemsets and association rules given certain threshold values.

The R package **arules** provides an interface to the C programs by Christian Borgelt which implement the two algorithms and basic infrastructure for creating and suitably transforming the input and analyzing the results.

## 2 Transaction data

The main application of association rules is for market basket analysis. In every transaction there are the items contained which are purchased at one visit. In one transaction there is usually only a small part of the items contained and the main interest is if the item is purchased or not. There are also other kinds of transaction data possible, as e.g., data containing the web pages visited during one session.

It can be assumed that transaction data consists of entries of the form

- transaction ID
- item ID
- user ID (optional)

Further information on transactions (as e.g., time, location), items (as e.g., category, price) or users (as e.g., socio-demographic variables: age, gender) might be available.

This kind of data can be transformed into a matrix with columns equal to the number of different transactions and rows equal to the number of different items. The matrix entries are the items present in the transaction. This matrix will in general be sparse.

Another application of mining association rules has been proposed for discovering interesting relationships between the values of different categorical variables. An example can be found in Hastie, Tibshirani, and Friedman (2001), where questionnaires data is used. Their data consists of ordinal, nominal and metric variables. A transaction for this kind of data consists of one customer and his characteristics and the items are the different values of all variables. This data can be transformed into a binary matrix, by first dichotomizing the metric variables and then coding each variable with $k$ categories by $k$ dummy variables.

The algorithms used for mining frequent itemsets or association rules need as input the transaction data transformed into a binary incidence matrix. A natural representation is a sparse matrix format. For our implementation we choose the format `cscMatrix` which is defined in the R package **Matrix**. "csc" stands for compressed, sparse, column-oriented. It contains the indices of the rows unequal to zero, the pointers to the initial indices of elements in each column and the non-zero elements of the matrix. For a binary matrix the non-zero elements are all equal to 1.

Another possibility of storing binary matrices would be to interpret every column as a binary number with can be coerced into an integer in the decimal system. We could then represent each column by one number. The drawback is that the decimal number grows exponentially with $2^n$ where $n$ is the number of columns. In R this transformation from a binary vector to a decimal number is only possible for $n \leq 30$, because then the integer number limit is reached. As the number of items will in general be much larger we have decided not to use this sparse format to represent the data.

The natural format for the questionnaires data in R is a `data.frame`. In order to mine frequent itemsets/association rules with `apriori` or `eclat` this data needs to be transformed to an incidence matrix. In addition it has to be taken care that the information on the different variables and their levels is not lost. Transaction data is hence represented in **arules** as a S4 class named `assMatrix` containing the following S4 classes:

- `cscMatrix`: sparse matrix representation of the incidence matrix
- `attributes`: contains the factor names, their levels, and labels constructed by collapsing factor name and level name separated by a dot

The attributes are necessary in order to keep track of the factors and levels of a data frame which is transformed into the sparse matrix format. This allows the retransformation of a `assMatrix` into the original data frame.

Methods implemented are

- `show`
- coercion: from (binary) matrices and data frames (and the corresponding retransformation)
- extract ([])
- `%in%`
- `as.character`
- `as.list`

# 3 Mining frequent itemsets/association rules

Free reference implementations of the algorithms apriori and eclat in C are available by Christian Borgelt (Borgelt and Kruse, 2002; Borgelt, 2003). The code is called directly from R by the functions `rapriori` and `reclat` and no writing to external files is necessary for the data exchange.

The input format of the data for the R functions is `assMatrix` (or a data format which can be coerced to `assMatrix`). Furthermore, information on the parameters needs to be given. It is distinguished between parameters which change the characteristics of the itemsets/rules/hyperedges mined, as e.g. the minimum support or the target, and parameters which influence the performance of the algorithm, as e.g., an initial sorting of the items with respect to their frequency.

The parameters which influence the output are contained in the argument `parameter`, the other parameters are summarized in `control`. These arguments have to be either instances of the classes `APparameter` and `APcontrol` for `rapriori` and `ECparameter` and `ECcontrol` for `reclat` or data which can be coerced to these classes, as e.g., `NULL` which will give the default values or a named list (names equal to slot names to change the default values). In these classes each slot specifies a different parameter and the values are already validated (to a certain extend). Parameters which have an integer specification are coerced before validity checking. The default values are equal to the defaults of the stand-alone `C` programs (cp. Borgelt, 2004) except that by default the original support definition is used for the specified minimum support required.

For `rapriori` there can be also the argument `appearance` specified, which determines which itemsets/rules/hyperedges are mined with respect to the items contained in the body or head.

It will in general be specified by a named list containing

- default: `character`, one of "head", "body", "none", "both" (or one of the other names described in the apriori manual; Borgelt, 2004)
- head: `character` or `integer`, specifying the items which are allowed to appear in the head of the rules
- body: `character` or `integer`, specifying the items which are allowed to appear in the body of the itemsets/rules/hyperedges
- none: `character` or `integer`, specifying the items which are not allowed to be contained in the itemsets/rules/hyperedges
- both: `character` or `integer`, specifying the items which are allowed to be contained in the itemsets/rules/hyperedges

This list is coerced to an `ASappearance` object within `rapriori` where the attributes information in the `assMatrix` data is used for determining the column number if `character` values are given.

The output of the functions `rapriori` and `reclat` is an object of class `arules` which contains the call, the (suitably modified) parameter specifications and the sets mined.

# 4 Sets: itemsets, rules, hyperedges

The output of `rapriori` or `reclat` are either itemsets (frequent itemsets, closed itemsets, maximal itemsets), rules or hyperedges. All three kinds contain the items involved and quality measures, as e.g., support, confidence, ....

In **arules** there is a virtual class `sets` defined which contains `itemsets`, `rules` and `hyperedges` as subclasses. It contains the following slots/classes:

- body: `cscMatrix` containing in each column the items which are in the body of the corresponding itemset/rule/hyperedge
- quality: `data.frame` with a row for each itemset/rule/hyperedge and a column for each quality measure (support, body.support, confidence, ...) returned
- rnb: `integer` number of rules
- `attributes`: containing the factor names, levels and labels from the input data

In addition `rules` contain

- head: `cscMatrix` containing in each column the items contained in the head of each rule

and `itemsets` contain

- trans: `cscMatrix` containing in each column the transactions which support the corresponding itemset (only returned by `eclat` if `trans` = `TRUE`)

Methods implemented are

- `show`

- `summary`
- extract ([)
- `subset`
- `as.data.frame`
- `as.character`

# 5 Example

**arules** provides the census data set from the UCI machine learning repository (Blake and Merz, 1998).

```
> library(arules)

Loading required package: stats4
Loading required package: Matrix

> data(census)
> dim(census)

[1] 48842    14

> census[1:2, 1:4]

          age        workclass education education-num
1 middle-aged        State-gov Bachelors            13
2      senior Self-emp-not-inc Bachelors            13

> census.assMatrix <- as(census, "assMatrix")
> census.assMatrix

Transaction matrix in sparse format with dimension 132 48842
```

The census data set contains 48842 observations on 14 categorical variables. If the data frame is transformed into a binary incidence matrix using dummy coding the number of different items is 132.

We can use this data for calling `rapriori` and finding all rules with a minimum support of 0.05 and where all other parameters have the default values.

```
> rules <- rapriori(census.assMatrix, parameter = list(support = 0.05))

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[132 item(s), 48842 transaction(s)] done [0.21s].
sorting and recoding items ... [41 item(s)] done [0.03s].
creating transaction tree ... done [0.25s].
checking subsets of size 1 2 3 4 5 done [0.69s].
writing ... [19038 rule(s)] done [0.02s].
creating S4 object  ... done [0.09s].

> rules
```

```
rules containing 19038 sets
derived using a minimum original support of 0.05

Call:
rapriori(data = census.assMatrix, parameter = list(support = 0.05))

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target
        0.8      0    1 none FALSE            TRUE    0.05      1      5  rules
  ext
 TRUE

Algorithmic control specification:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE
```

The specified parameter values are validated and a support > 1 gives

```
> error <- try(rapriori(census.assMatrix, parameter = list(support = 1.3)))
> error

[1] "Error in validObject(.Object) : Invalid \"APparameter\" object: support = 1.3 > 1\n"
attr(,"class")
[1] "try-error"
```

The function `subset` can be used if we are only interested in the subset of rules which certain variables or values of variables in the head/body or where a given quality measure fulfills a certain criterion.

```
> sets <- rules@sets
> sets.sub <- subset(sets, subset = head %in% "sex" & lift > 1.4)
```

For an overview on the rules mined the function `summary` can be used which gives a description on the number of rules, their lengths, the items contained in body and head and the quality measures.

```
> summary(sets.sub)

Object contains 612 rules.

body lengths:

  1   2   3   4
  2  33 168 409

head:
Female.sex   Male.sex
         1        611

body:
             Husband.relationship Married-civ-spouse.marital-status
                              470                               172
                none.capital-loss      United-States.native-country
                              159                               158
                       White.race                 none.capital-gain
                              157                               155
```

```
                            (Other)
                                937

quality:
    support           confidence          lift          body.support
 Min.   :0.05004   Min.   :0.8005   Min.   :1.401   Min.   :0.05004
 1st Qu.:0.06447   1st Qu.:0.9997   1st Qu.:1.495   1st Qu.:0.06585
 Median :0.08276   Median :0.9999   Median :1.496   Median :0.08357
 Mean   :0.10863   Mean   :0.9886   Mean   :1.481   Mean   :0.10956
 3rd Qu.:0.12598   3rd Qu.:1.0000   3rd Qu.:1.496   3rd Qu.:0.12599
 Max.   :0.40365   Max.   :1.0000   Max.   :2.415   Max.   :0.40367
```

The sets object can be coerced to a data frame which allows the printing of the rules similar to the output format of the C programs.

```
> as.data.frame(sets.sub)[1:3, ]

        head                                                 body    support
54  Male.sex                            Craft-repair.occupation 0.11852504
145 Male.sex                               Husband.relationship 0.40364850
436 Male.sex Exec-managerial.occupation Husband.relationship 0.06615208
    confidence     lift body.support
54   0.9471531 1.416871   0.12513820
145  0.9999493 1.495851   0.40366897
436  1.0000000 1.495926   0.06615208
```

# References


Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216. ACM Press, 1993. bg0003.

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 September 1994. bg0004.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. URL http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Christian Borgelt. Efficient implementations of apriori and eclat. In *Workshop Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL, USA)*, 2003.

Christian Borgelt. *Apriori—Finding Association Rules/Hyperedges with the Apriori Algorithm*, 2004. URL http://fuzzy.cs.uni-magdeburg.de/ borgelt/apriori.html.

Christian Borgelt and Rudolf Kruse. Induction of association rules: Apriori implementation. In *Proc. 15th Conf. on Computational Statistics (Compstat 2002, Berlin, Germany)*, Heidelberg, Germany, 2002. Physika Verlag. bg0001.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning (Data Mining, Inference and Prediction)*. Springer Verlag, 2001.