A Computational Environment for Mining Association Rules and Frequent Item Sets

Michael Hahsler and Bettina Grün and Kurt Hornik

April 15, 2005

Abstract

Mining frequent itemsets and association rules is a popular and well researched approach to discovering interesting relationships between variables in large databases. The R package **arules** presented in this paper provides a basic infrastructure for creating and manipulating input data sets and for analyzing the resulting itemsets and rules. The package also includes interfaces to two fast mining algorithms, the popular C implementations of Apriori and Eclat by Christian Borgelt. These algorithms can be used to mine frequent itemsets, maximal frequent itemsets, closed frequent itemsets and association rules.

1 Introduction

Mining frequent itemsets and association rules is a popular and well researched method for discovering interesting relations between variables in large databases. Piatetsky-Shapiro (1991) describes analyzing and presenting strong rules discovered in databases using different measures of interest. Based on the concept of strong rules, Agrawal, Imielinski, and Swami (1993) introduced the problem of mining association rules from transaction data as follows.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called *items*. Let $\mathcal{D} = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the *database*. Each transaction in $\mathcal{D}$ contains a subset of the items in I. A *rule* is defined as an implication of the from $X \Rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$. The sets of items (for short *itemsets*) X and Y are called *antecedent* (left-hand-side or LHS) and *consequent* (right-hand-side or RHS) of the rule.

To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best-known constraints are minimum thresholds on support and confidence. *Support* is defined on an itemset as the proportion of transactions in the data set which contain the itemset. All itemsets which have a support above a set minimum support threshold are called *frequent itemsets*. Finding frequent itemsets can be seen as a simplification of the unsupervised learning problem called "mode finding" or "bump hunting" (Hastie, Tibshirani, and Friedman, 2001). For these problems each item is seen as a variable. The goal is to find prototype values so that the probability density evaluated at these values is sufficiently large. However, for practical applications with a large number of variables, probability estimation will be unreliable and computationally too expensive. This is why in practice frequent itemsets are used instead of probability estimation.

Confidence is defined on rules as $\mathrm{conf}(X \Rightarrow Y) = \mathrm{supp}(X \cup Y)/\mathrm{supp}(X)$. This can be interpreted as an estimate of the probability $P(Y|X)$, the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS (see e.g., Hipp, Güntzer, and Nakhaeizadeh, 2000). Association rules are typically required to satisfy both constraints, minimum support and minimum confidence, at the same time.

At medium to low support values, often a great number of frequent itemsets are found in a database. However, since the definition of support enforces that all subsets of a frequent itemset have to be also frequent, it is sufficient to only mine all *maximal frequent itemsets*, defined as frequent itemsets which are not proper subsets of any other frequent itemset (Zaki, Parthasarathy, Ogihara, and Li, 1997). Another approach to reduce the number of mined itemsets is to only mine *frequent closed itemsets.* An itemset is closed if no proper superset of the itemset is contained in each transaction in which the itemset is contained (Pasquier, Bastide, Taouil, and Lakhal, 1999; Zaki, 2004). Frequent closed itemsets are a superset of the maximal frequent itemsets. Their advantage over maximal frequent itemsets is that in addition to be able to infer all frequent itemsets, they also preserve the support information for all frequent itemsets which can be important for computing additional interest measures after the mining process is finished (e.g., *lift* (Brin, Motwani, Ullman, and Tsur, 1997), or *all-confidence* (Omiecinski, 2003)).

In the last decade research on algorithms to solve the frequent itemset problem has been abundant. Goethals and Zaki (2004) compare the currently fastest algorithms. Among these algorithms are the implementations of the Apriori and Eclat algorithms by Borgelt (2003) interfaced in the package **arules**. The two algorithms use very different mining strategies. Apriori, developed by Agrawal and Srikant (1994), is a level-wise, breadth-first algorithm which counts transactions. In contrast, Eclat (Zaki et al., 1997) employs equivalence classes, depth-first search and set intersection instead of counting. The algorithms can be used to mine frequent itemsets, maximal frequent itemsets and closed frequent itemsets. The implementation of Apriori can additionally be used to generate association rules.

The R package **arules** presented in this paper provides the infrastructure needed to create and manipulate input data sets for the mining algorithms and for analyzing the resulting itemsets and rules. Since it is common to work with large sets of rules and itemsets, the package uses sparse matrix representation to minimize memory usage. The infrastructure provided by the package was also created to explicitly facilitate easy extensions, both for interfacing new algorithms and for adding new types of interest measures and associations.

The rest of the paper is organized as follows: In the next section we give an overview of the data structure implemented in the package **arules**. In sections 3 and 4 we introduce the functionality of the classes to handle transaction data and associations. In section 5 we describe the way mining algorithms are interfaced in **arules** using the already implemented interfaces for Apriori and Eclat as examples. We provide several examples in sections 6 to 8. The first two examples show typical R sessions for analyzing and manipulating a transaction data set, and for mining association rules. The third example demonstrates how **arules** can be extended to integrate a new interest measure. We conclude with a summary of the features and advantage of the package **arules** as a computational environment for mining association rules and frequent itemsets.

2 Data structure overview

To enable the user to represent and work with input and output data of association rule mining algorithms in R, a well thought out structure is necessary which can deal in an efficient way with large amounts of sparse binary data. The S4 class structure implemented in the package **arules** is presented in figure 1.

For input data the class transactions is provided. The output of the mining algorithms comprises the classes itemsets and rules representing a set of itemsets or a set of rules, respectively. Both classes directly extend a common virtual class called associations which provides a common interface. In this structure it is easy to add a new type of associations by adding a new class that extends associations.

Items in associations and transactions are implemented by the itemMatrix class which provides a facade for the sparse Matrix implementation dgCMatrix from package **Matrix** (Bates and Maechler, 2005). Objects of the itemMatrix class are not intended to be directly accessed by the end user of **arules**. The interfaces of associations and transactions can be used without knowledge of how the internal representation of the data works. However, the data structure in itemMatrix or even the dgCMatrix can be directly accessed if necessary (e.g., to efficiently compute a distance matrix

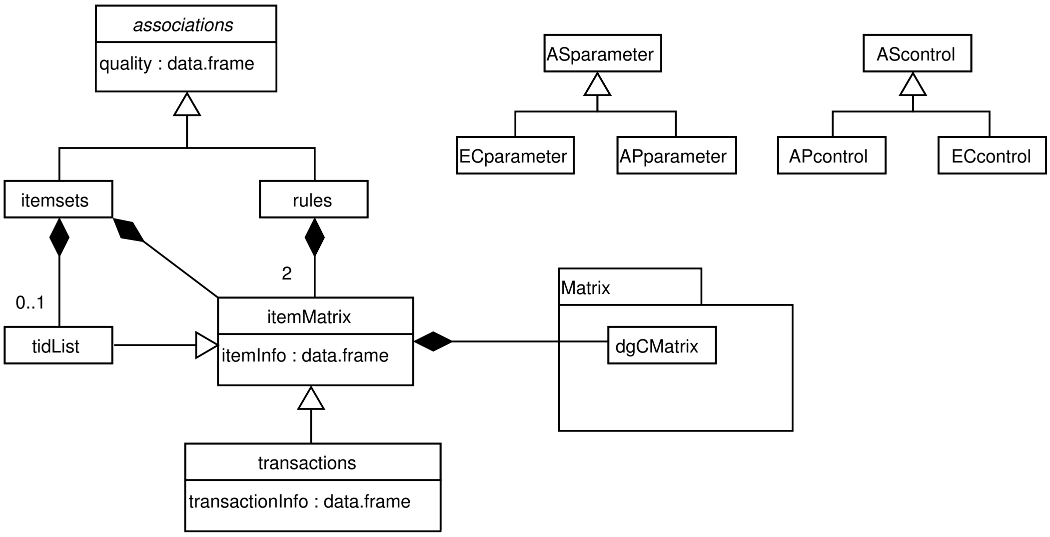


Figure 1: UML class diagram of the **arules** package.

between itemsets for clustering).

To control the behavior of the mining algorithms, the two classes ASparameter and AScontrol are used. Since each algorithm can use additional algorithm-specific parameters, we implemented for each interfaced algorithm its own set of control classes. We used the prefix 'AP' for Apriori and 'EC' for Eclat. In this way, it is easy to extend the control classes when interfacing a new algorithm.

3 Transaction data

The main application of association rules is for market basket analysis where large transaction data sets are mined. In this setting each transaction contains the items which were purchased at one visit to a retail store ((see e.g., Berry and Linoff, 1997)). Transaction data are normally recorded by point-of-sale scanners and consists of tuples of the form:

$$< \textit{transaction ID}, \textit{item ID}, \ldots >$$

All tuples with the same transaction ID form a single transaction which contains all the items given by the item IDs in the tuples. Additional information denoted by the dots might be available. For example, the customer ID might be available via a loyalty program in a supermarket. Further information on transactions (e.g., time, location), on the items (e.g., category, price) or on the customer (socio-demographic variables as age, gender, etc.) might be available.

For mining, the transaction data is first transformed into a binary purchase incidence matrix with columns equal to the number of different items and rows equal to the number of different transactions. The matrix entries represent the presence (1) or absence (0) of an item in a particular transaction. An example of a binary incidence matrix is depicted in Figure 2. This format is often called the *horizontal* database layout (Zaki, 2000). Alternatively, transaction data can be represented in a *vertical* database layout in the form of a *transaction ID list* (Zaki, 2000). In this format for each item a list of IDs of the transactions the item is contained in is stored. Depending of the algorithm, one of the layouts is used for mining. In **arules** both layouts are implemented as the classes transactions and tidList and the data can be directly transformed from one format to the other.

Since a typical supermarket transaction only contains a small number of items compared to the total number of available items, the binary incidence matrix will in general be very sparse with items

transactions	i_1	i_2	i_3	...	i_n
t_1	0	1	0	...	1
t_2	0	1	0	...	1
t_3	0	1	0	...	0
t_4	0	0	0	...	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
t_{m-1}	1	0	0	...	1
t_m	0	0	1	...	1

Figure 2: Example of a transaction data set represented as a binary incidence matrix.

many items and a very large number of transactions. A natural representation for such data is a sparse matrix format. For our implementation we chose the dgCMatrix which is defined in the R package **Matrix** implemented by Bates and Maechler (2005). The dgCMatrix is a compressed, sparse, column-oriented matrix which 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. Since the package **Matrix** does not provide subset selection functionality for dgCMatrix, we implemented a suitable function in C and interfaced is as the subset selection method (`[`). Despite the column orientation of the dgCMatrix, it is more convenient to work with incidence matrices which are row-oriented. This makes the most important manipulation, selecting a set of transactions from a data set for mining, more comfortable and efficient. Therefore, we implemented the class itemMatrix providing a row-oriented facade to the dgCMatrix which stores a transposed incidence matrix. At this level also the constraint that the incidence matrix is binary (and not real valued as the dgCMatrix) is enforced. Additionally, itemMatrix stores item labels (e.g., name of the items) and handles the necessary mapping between the item label and the corresponding column number in the incidence matrix. Optionally, itemMatrix can also store additional information on items. For example, the category hierarchy in a supermarket setting can be stored which enables the analyst to select only transactions (or as we late see also rules and itemsets) which contain items from a certain category (e.g., all dairy products).

For itemMatrix, basic methods including `dim`, subset selection (`[`) and coercion from and to matrix and list primitives are provided. Additionally, methods specific to the needs for **arules** are implemented. Since itemMatrix is used to store a set of transactions or, more general, a set of itemsets, we implemented a `length` method which returns the number of elements in the set (i.e., the number of transactions or the number of itemsets in the set). Technically, `length` returns the number of rows of the sparse matrix. The `size` method returns a vector with the sizes of each element in the set (row in the matrix). For example, for a purchase incidence matrix we will get a vector of length of the number of transactions in the matrix and each element of the vector contains the size (number of items) of the corresponding transaction. This information can be used to select or filter unusually long or short transactions. Finally, an `image` method can be used to produce a level plot of the binary matrix useful for quick visual inspection. For transaction data sets (e.g., point-of-sale data) a plot can be very helpful for checking if the data set contains structural changes (e.g., items were not offered or out-of-stock during part of the observation period) or to find abnormal transactions (e.g., transactions which contain almost all items may point to recording problems). Spotting such problems can be very helpful for data preparation.

The class transactions directly extends itemMatrix and inherits its basic matrix functionality (e.g., subset selection). In addition, transactions has a slot to store additional information for each transaction in form of a data.frame. The slot can hold arbitrary named vectors with length equal to the number of stored transactions. In **arules** the slot is currently used to store transaction IDs, however, it can easily be used to store user IDs, revenue or profit, or other information on each transaction. With this information subsets of transactions (e.g., only transactions of a certain user or exceeding a set profit) can be selected. Objects of class transactions can be easily created by coercion from matrix or list. If names are available in this data structures, they are used as item labels or transaction IDs accordingly. To import data from a file, the `read.transactions` function is provided. This function reads files structured as shown above and also the very common format with one line per transaction and the items separated by a predefined character. Finally, the method `inspect` can be used to inspect transactions (e.g., an interesting transaction selected with subset selection).

Another important 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 et al. (2001), where questionnaire data is used. The natural format for questionnaire data in R is a data.frame with the answers coded as ordinal, nominal and metric variables. In order to mine associations with Apriori or Eclat this data needs to be transformed into a binary incidence matrix with each row representing one questionnaire. To create the binary matrix, first the metric variables are transformed into ordinal variables by building categories (e.g., the variable salary is transformed into an ordinal variable with the three values: low, medium and high). Then, each variable with k categories is represented by k binary dummy variables. Since it is crucial to carefully choose a value range for each categories, the first transformation has to be done manually by the analyst. The result is a data.frame with all ordinal or nominal variables coded as factors. The second step, the generation of the needed number of dummy variables, is then done automatically by the `coerce` method from data.frame to transactions. In this process, the original variable names and categories from the questionnaire are preserved as additional item information and can later be used to select itemsets or rules which contain items referring to a certain original variable. The resulting transactions object can be mined and analyzed the same way as market basket data, see the example in Section 6.

4 Sets of itemsets and sets of rules

The result of mining transaction data in **arules** are associations. Associations are conceptually sets of objects. Each object describes the relationship between some items (e.g., an itemset or a rule). and has values for different measures of quality assigned. Such quality measures can be measures of significance (e.g., support) or measures of interest (e.g., confidence, lift) or other measures (e.g., revenue covered by the association).

All association types have a common interface suitable for set operations. Methods for subset extraction (`[` and the `subset` method), getting the number of elements in the set with `length`, and sorting the set using the values of different quality measures (method `SORT`) are available. A `summary` method produces a short overview of the set and with `inspect` individual associations can be inspected.

In **arules** currently sets of itemsets (e.g., used for frequent itemsets of their closed or maximal subset) and sets of rules (e.g., association rules) are implemented as associations. Both itemsets and rules directly extend the virtual class associations. Class itemsets contains one itemMatrix object to store the items as a binary matrix where each row in the matrix represents an itemset. In addition, it contain a transaction ID list of class tidList which is implemented as a sparse matrix (reusing itemMatrix). A transaction ID list stores for each itemset a list of transaction ID in which the itemset appears. Such lists are only returned by `eclat`. Class rules consists of two itemMatrix objects representing the left-hand-side (LHS) and the right-hand-side (RHS) of the rules, respectively.

The items in the associations and the quality measures can be accessed and manipulated in a safe way using accessor and replace methods for `quality`, `items`, `lhs` and `rhs`. In addition the association classes have built-in validity checking which ensures that all elements have a matching dimension.

It is simple to add new quality measures to existing associations. Since the `quality` slot holds a data.frame, additional columns with new quality measures can be added. These new measures can then be used to sort or select associations using the SORT or the subset methods. Adding a new type of associations to **arules** is easy as well. One has only to implement a new class extending the virtual associations class.

5 Mining algorithm interfaces

In package **arules** we interface free reference implementations of Apriori and Eclat by Christian Borgelt (Borgelt and Kruse, 2002; Borgelt, 2003). The code is called directly from R by the functions apriori and eclat and the data objects are directly passed from R to the C code and back without writing to external files.

The input format of the data for the apriori and eclat functions is transactions or a data format which can be coerced to transactions (e.g., matrix or list). The algorithm parameters are divided into two groups represented by the arguments parameter and control. The mining parameters (parameter) change the characteristics of the mined itemsets or rules (e.g., the minimum support) and the control parameters (control) influence the performance of the algorithm (e.g., an initial sorting of the items with respect to their frequency). These arguments have to be instances of the classes APparameter and APcontrol for the function apriori or ECparameter and ECcontrol for the function eclat, respectively. Alternatively, data which can be coerced to these classes (e.g., NULL which will give the default values or a named list with names equal to slot names to change the default values) can be passed. In these classes each slot specifies a different parameter and the values. The default values are equal to the defaults of the stand-alone C programs (Borgelt, 2004) except that by default the more common original support definition (instead of the support of only the antecedent) is used for the specified minimum support required.

For apriori the appearance feature implemented by Christian Borgelt can also be used. With argument appearance of function apriori one can specify which items have to or cannot appear in itemsets or rules. For more information on this feature we refer to the Apriori manual Borgelt (2004).

The output of the functions apriori and eclat is an object of a class extending associations which contains the sets mined associations and can be further analyzed using the methods provided for these classes.

It is straightforward to interface additional algorithms which use a incidence matrix or transaction ID list representation as input. The necessary steps are:

1. Adding interface code to the algorithm, preferably by directly calling into the native implementation language (rather than using files for communication), and an R function calling this interface.

2. Implementing extensions for parameter and control.

Implementations of algorithms as kDCI, LCM, FP-Growth or Patricia are discussed in Goethals and Zaki (2003), and the source code is available on the internet.

6 Example 1: Analyzing the Epub data set

In this example we look at the Epub transaction data contained in package **arules**. This data set contains downloads of documents from the Electronic Publication platform of the Vienna University of Economics and Business Administration available via http://epub.wu-wien.ac.at.

First, we load **arules** and the data set.

```
> library("arules")
```

```
Loading required package: Matrix
Loading required package: stats4
```

```
> data("Epub")
> Epub
```

```
transactions in sparse format with
 2771 transactions (rows) and
 419 items (columns)
```

We see that the data set consists of 2771 transactions and is represented as a sparse matrix with 2771 rows and 419 columns which represent the items. Next, we use the summary method to get more information about the data set.

```
> summary(Epub)
```

```
transactions as itemMatrix in sparse format with
 2771 rows (elements/itemsets/transactions) and
 419 columns (items)

most frequent items:
epub-wu-01_11d epub-wu-01_4c6 epub-wu-01_2cd  epub-wu-01_71 epub-wu-01_364
          177            100             90             90             89
       (Other)
          4436

element (itemset/transaction) length distribution:
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
1976  411  164   78   43   20   13   12   10    5    7    4    4    2    1    3
  17   18   19   20   22   24   25   28   34   38   74
   2    2    5    2    1    1    1    1    1    1    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   1.000   1.798   2.000  74.000

includes extended transaction information - examples:
                     transactionIDs                TimeStamp
1 epub-wu-01_session4795-1041447099 Wed Jan  1 19:59:00 2003
2 epub-wu-01_session4797-1041486295 Thu Jan  2 06:46:01 2003
3 epub-wu-01_session479a-1041497371 Thu Jan  2 09:50:38 2003
```

The summary method displays the most frequent items in the data set, information about the transaction length distribution and that the data set contains some extended transaction information. We see that the data set contains transaction IDs and in addition time stamps for the transactions. The additional information can be used for analyzing the data set.

```
> year <- substr(as(transactionInfo(Epub)[["TimeStamp"]], "character"),
+     21, 24)
> table(year)
```

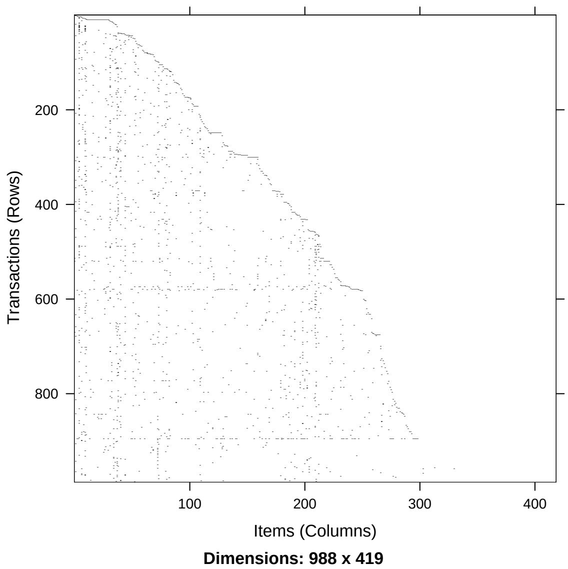
```
year
2003 2004 2005
 988 1375  408
```

We selected only the year part of the time stamps. For 2003, the first year in the dataset we have 988 transactions. We can select the corresponding transactions and inspect the structure using a level-plot.

```
> Epub_2003 <- Epub[year == "2003"]
> length(Epub_2003)

[1] 988

> image(Epub_2003)
```



The plot is a direct visualization of the binary incidence matrix where the the dark dots represent the ones in the matrix. From the plot we see that the items in the data set are not evenly distributed. In fact, the white area to the top right site suggests, that in the beginning of 2003 only very few items were available (less than 50) and then during the year more items were added till it reached a number of around 300 items. Also, we can see that there are two transactions in the data set which contain a very high number of items (denser horizontal lines). These transactions need further investigation since they could originate from data collection problems (e.g., a web robot downloading many documents from the publication site). To find the very long transactions we can use the size method and select very long transactions (containing more than 20 items).

```
> transactionInfo(Epub_2003[size(Epub_2003) > 20])

                       transactionIDs                TimeStamp
301 epub-wu-01_session56e2-1051611211 Tue Apr 29 12:30:38 2003
580 epub-wu-01_session6308-1061133365 Sun Aug 17 17:16:12 2003
896 epub-wu-01_session72dc-1072722731 Mon Dec 29 19:35:35 2003
```

We found three long transactions and printed the corresponding transaction information. Of course, size can also be used in a similar fashion to remove long or short transactions.

Transactions can be inspected using the inspect method. Since the long transactions identified above would result in a very long printout, we will inspect the first 5 transactions in the subset for 2003.

```
> inspect(Epub_2003[1:5])

  items                                       transactionIDs                TimeStamp
1 {epub-wu-01_3d6}  epub-wu-01_session4795-1041447099 Wed Jan  1 19:59:00 2003
2 {epub-wu-01_16f}  epub-wu-01_session4797-1041486295 Thu Jan  2 06:46:01 2003
3 {epub-wu-01_f4}   epub-wu-01_session479a-1041497371 Thu Jan  2 09:50:38 2003
4 {epub-wu-01_11d,
   epub-wu-01_1a7,
   epub-wu-01_83}   epub-wu-01_session47b7-1041526514 Thu Jan  2 17:55:50 2003
5 {epub-wu-01_154}  epub-wu-01_session47bb-1041535625 Thu Jan  2 20:27:44 2003
```

Most transactions contain one item. Only transaction 4 contains three items. Alternatively, transactions can be converted into a list with:

```
> as(Epub_2003[1:5], "list")

$"epub-wu-01_session4795-1041447099"
[1] "epub-wu-01_154"

$"epub-wu-01_session4797-1041486295"
[1] "epub-wu-01_3d6"

$"epub-wu-01_session479a-1041497371"
[1] "epub-wu-01_16f"

$"epub-wu-01_session47b7-1041526514"
[1] "epub-wu-01_f4"  "epub-wu-01_11d" "epub-wu-01_1a7"

$"epub-wu-01_session47bb-1041535625"
[1] "epub-wu-01_83"
```

Finally, transaction data in horizontal layout can be converted to transaction ID list in vertical layout using coercion.

```
> Epub_tidList <- as(Epub, "tidList")
> Epub_tidList

tidList in sparse format for
 419 items/itemsets (rows) and
 2771 transactions (columns)
```

For performance reasons the transaction ID list is also stored in a sparse matrix. To get a list, coercion to list can be used.

```
> as(Epub_tidList[1:3], "list")

$"epub-wu-01_154"
[1] "epub-wu-01_session4795-1041447099" "epub-wu-01_session6082-1058883924"
[3] "epub-wu-01_session60dd-1059130239" "epub-wu-01_session67db-1065044430"
[5] "epub-wu-01_session769c-1075191357" "epub-wu-01_session7ee3-1079450030"
```

```
$"epub-wu-01_3d6"
 [1] "epub-wu-01_session4797-1041486295" "epub-wu-01_session4893-1042136277"
 [3] "epub-wu-01_session48f4-1042453749" "epub-wu-01_session4ca3-1044889013"
 [5] "epub-wu-01_session52c6-1049273642" "epub-wu-01_session5712-1051701668"
 [7] "epub-wu-01_session58e3-1052992410" "epub-wu-01_session5984-1053467491"
 [9] "epub-wu-01_session5b20-1054675502" "epub-wu-01_session5c20-1055421043"
[11] "epub-wu-01_session5dc0-1056639134" "epub-wu-01_session5eac-1057261298"
[13] "epub-wu-01_session6599-1063473887" "epub-wu-01_session673d-1064583856"
[15] "epub-wu-01_session683e-1065381126" "epub-wu-01_session6f2f-1069854482"
[17] "epub-wu-01_session708a-1070754608" "epub-wu-01_session7a0c-1076882429"
[19] "epub-wu-01_session7de5-1078926808" "epub-wu-01_session89db-1084827080"
[21] "epub-wu-01_session9227-1089148583" "epub-wu-01_session9941-1094031566"
[23] "epub-wu-01_sessiona4d7-1100508833" "epub-wu-01_sessiona8c0-1102612273"
[25] "wu01_session4450a-1045050224"      "wu01_session4a129-1057762457"
[27] "wu01_session4d25a-1066490150"

$"epub-wu-01_16f"
[1] "epub-wu-01_session479a-1041497371" "epub-wu-01_session56e2-1051611211"
[3] "epub-wu-01_session630c-1061175093" "epub-wu-01_session72dc-1072722731"
[5] "epub-wu-01_session8b3e-1085510896" "epub-wu-01_session91ab-1088878266"
[7] "epub-wu-01_sessiona202-1098976943" "epub-wu-01_sessiona7b9-1101827029"
```

In this representation each item has an entry in with a list of all transactions it occurs in.

7 Example 2: Mining the Adult data set

As a second example, we use the Adult data set from the UCI machine learning repository (Blake and Merz, 1998) provided by package **arules**. This data set is similar to the data used by Hastie et al. (2001). The data originates from the U.S. census bureau database and contains 48842 instances with 14 variable like age, work class, education, salary, etc.

```
> library("arules")
> data("Adult")
> dim(Adult)

[1] 48842    14

> Adult[1:2, 1:4]

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

The metric variables in the `Adult` data frame have already been transformed into suitable categories and the values of all variables have been encoded as factors. The data can be automatically recoded as a binary incidence matrix by coercing the data set to transactions.

```
> Adult_transactions <- as(Adult, "transactions")

Recoded 14 variables to 132 binary items
```

The census data set contains 14 categorical variables which are automatically recoded into 132 binary items. During encoding the item labels were generated in the form of *<variable name>* = *<category label>*.

```
> summary(Adult_transactions)

transactions as itemMatrix in sparse format with
 48842 rows (elements/itemsets/transactions) and
 132 columns (items)

most frequent items:
           capital-loss = none            capital-gain = none
                         46560                          44807
native-country = United-States                   race = White
                         43832                          41762
                salary = small                        (Other)
                         37155                         463207

element (itemset/transaction) length distribution:
   11    12    13    14
   46  2753   821 45222

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  11.00   14.00   14.00   13.87   14.00   14.00

includes extended item information - examples:
             labels variables      levels
1 age = middle-aged       age middle-aged
2         age = old       age         old
3      age = senior       age      senior
```

The summary of the transaction data set gives a rough overview showing the most frequent items, the length distribution of the transactions and the extended item information which shows which variable and which value were used to create each binary item. In the first example we see that the item with label `age=middle-aged` was generated by variable `age` and value `middle-aged`.

Next, we call the function `apriori` to find all rules (the default association type for `apriori`) with a minimum support of 0.01 and a confidence of 0.8 in the first 40,000 transactions.

```
> rules <- apriori(Adult_transactions[1:40000], parameter = list(support = 0.01,
+     confidence = 0.8))

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target
        0.8    0.1    1 none FALSE            TRUE    0.01      1      5  rules
   ext
 FALSE

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

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), 40000 transaction(s)] done [0.11s].
sorting and recoding items ... [81 item(s)] done [0.02s].
creating transaction tree ... done [0.11s].
checking subsets of size 1 2 3 4 5 done [1.10s].
writing ... [104952 rule(s)] done [0.07s].
creating S4 object  ... done [0.91s].

Result: set of 104952 rules
```

First, the function prints the used parameters. Apart from the specified minimum support and minimum confidence, all parameters have the default values. It is important to note that with parameter `maxlen`, the maximum size of mined frequent itemsets, is by default restricted to 5. Longer association rules are only mined if `maxlen` is set to a higher value. After the parameter settings, the output of the C implementation of the algorithm with timing information is displayed.

The result of the mining algorithm is a set of 104952 rules. For an overview of the mined rules the function `summary` can be used. It shows the number of rules, the most frequent items contained in the left-hand-side and the right-hand-side and their respective length distributions and summary statistics for the quality measures returned by the mining algorithm.

```
> summary(rules)

set of 104952 rules

rule length distribution (lhs + rhs):
    1     2     3     4     5 
    4   374  5013 26129 73432 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   4.000   5.000   4.645   5.000   5.000 

summary of quality measures:
    support          confidence          lift        
 Min.   :0.01002   Min.   :0.8000   Min.   : 0.8676  
 1st Qu.:0.01370   1st Qu.:0.8937   1st Qu.: 1.0089  
 Median :0.02106   Median :0.9369   Median : 1.0468  
 Mean   :0.03729   Mean   :0.9294   Mean   : 2.1943  
 3rd Qu.:0.03898   3rd Qu.:0.9714   3rd Qu.: 1.3087  
 Max.   :0.95335   Max.   :1.0000   Max.   :97.7995  
```

As typical for association rule mining, the number of found rules is huge. To analyze these rules, for example, the function `subset` can be used to produce a subset of rules which contain items which resulted form the variable `salary` in the right-hand-side of the rule and the `lift` measure exceeds 1.4.

```
> rules.sub <- subset(rules, subset = rhs %in% "salary" & lift >
+     1.4)
```

We can then inspect the three rules with the highest lift value (using the SORT method).

```
> inspect(SORT(rules.sub, by = "lift")[1:3])
```

```
  lhs                                          rhs              support confidence     lift
1 {capital-gain = high,
    occupation = Exec-managerial,
    marital-status = Married-civ-spouse} => {salary = large} 0.010050  0.9686747 4.042883
2 {capital-gain = high,
    occupation = Exec-managerial,
    marital-status = Married-civ-spouse,
    capital-loss = none}                 => {salary = large} 0.010050  0.9686747 4.042883
3 {capital-gain = high,
    occupation = Exec-managerial,
    sex = Male,
    native-country = United-States}      => {salary = large} 0.010125  0.9665871 4.034170
```

Using such subset selection and sorting a set of associations can be analyzed even if it is huge.

8 Example 3: Extending arules for all-confidence

In this example we show how easy it is to add a new interest measure. As the interest measure we chose *all-confidence* introduced by Omiecinski (2003). All-confidence is defined on itemsets X as:

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

This measure has the property $\text{conf}(I \Rightarrow Z \setminus I) \geq \text{all-confidence}(X)$ 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 $\max(\text{supp}(i \in X))$.

First, we use Eclat to mine frequent itemsets from the previously used Adult data set.

```
> fsets <- eclat(Adult_transactions, parameter = list(support = 0.05),
+     control = list(verbose = FALSE))


Parameter specification:
 tidList support minlen maxlen            target   ext
   FALSE    0.05      1      5 frequent itemsets FALSE


Result: set of 9371 itemsets
```

For the denominator of all-confidence we need to find all mined single items and their corresponding support values.

```
> single_item_fsets <- fsets[size(items(fsets)) == 1]
> single_items <- data.frame(item = unlist(LIST(items(single_item_fsets),
+     decode = FALSE)), support = quality(single_item_fsets))
> single_items[1:3, ]

     item   support
9331   84 0.9532779
9332   81 0.9173867
9333  128 0.8974243
```

Next, we can calculate the all-confidence for all itemsets and add it to the set's quality data frame.

```
> itemset_list <- LIST(items(fsets), decode = FALSE)
> all_conf <- sapply(1:length(itemset_list), function(x) {
+     quality(fsets)$support[x]/max(single_items$support[match(itemset_list[[x]],
+         single_items$item)])
+ })
> quality(fsets) <- cbind(quality(fsets), all_conf)
```

The new quality measure can now be used to manipulate the set. For example the set can be sorted by all-confidence.

```
> inspect(SORT(fsets, by = "all_conf")[1:3])

  items                       support all_conf
1 {education = Masters,
    education = Bachelors}   0.0543999        1
2 {education-num = 13,
    education-num = 10}      0.1643053        1
3 {education = Some-college,
    education-num = 14}      0.2227182        1
```

9 Summary and outlook

Previously, there was no functionality for mining and handling associations available for R. With package **arules** we provide the basic infrastructure which enables us to easily combine association mining with clustering and visualization techniques already available in R. The main features are as follows.

- Efficient implementation using sparse matrices.
- Simple and intuitive interface to manipulate and analyze transaction data, sets of itemsets and rules with subset selection and sorting.
- Interface to two fast mining algorithms.
- Flexibility in terms of adding new quality measures, and additional item and transaction descriptions which can be used for selecting transactions and analyzing resulting associations.
- Extensible data structure to allow for easy implementation of new types of associations and interfacing new algorithms.

There are several interesting possibilities to extend **arules**. For example, it would be very useful to interface algorithms which use statistical measures to find "interesting" itemsets (which are not necessarily frequent itemsets as used in an association rule context). Such algorithms include implementations of the χ^2-test based algorithm by Silverstein, Brin, and Motwani (1998) or the baseline frequency approach by DuMouchel and Pregibon (2001).

Another interesting extension would be to interface synthetic data generators for fast evaluation and comparison of different mining algorithms. The best known generator for transaction data for mining association rules was developed by Agrawal and Srikant (1994). Alternatively data can be generated by simple probabilistic models as done by Hahsler, Hornik, and Reutterer (2005).

Finally, similarity measuers between itemsets and rules can be implemented in **arules**. With such measures distance based clustering and visualization of associations is possible (see e.g., Strehl and Ghosh, 2003)).

References

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.

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. URL `http://doi.acm.org/10.1145/170035.170072`.

Douglas Bates and Martin Maechler. *Matrix: A Matrix package for R*, 2005. R package version 0.95-5.

Michael J. A. Berry and Gordon S. Linoff. *Data Mining Techniques for Marketing, Sales and Customer Support*. Wiley Computer Publishing, 1997.

Catherine L. Blake and Christopher J. Merz. *UCI Repository of machine learning databases*. University of California, Irvine, Dept. of Information and Computer Sciences, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Christian Borgelt. Efficient implementations of Apriori and Eclat. In *FIMI'03: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, November 2003.

Christian Borgelt. *Apriori—Finding Association Rules/Hyperedges with the Apriori Algorithm*. Working Group Neural Networks and Fuzzy Systems, Otto-von-Guericke-University of Magdeburg, Universitätsplatz 2, D-39106 Magdeburg, Germany, 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.

Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 255–264, Tucson, Arizona, USA, May 1997.

William DuMouchel and Daryl Pregibon. Empirical bayes screening for multi-item associations. In F. Provost and R. Srikant, editors, *Proceedings of the ACM SIGKDD Intentional Conference on Knowledge Discovery in Databases & Data Mining (KDD01)*, pages 67–76. ACM Press, 2001.

Bart Goethals and Mohammed J. Zaki, editors. *FIMI'03: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, November 2003. Sun SITE Central Europe (CEUR).

Bart Goethals and Mohammed J. Zaki. Advances in frequent itemset mining implementations: Report on FIMI'03. *SIGKDD Explorations*, 6(1):109–117, 2004.

Michael Hahsler, Kurt Hornik, and Thomas Reutterer. Implications of probabilistic data modeling for rule mining. Technical Report 14, Department of Statistics and Mathematics, Wirschaftsuniversität Wien, Augasse 2-6, 1090 Wien, March 2005. URL `http://epub.wu-wien.ac.at/dyn/openURL?id=oai:epub.wu-wien.ac.at:epub-wu-01_766`.

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

Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explorations*, 2(2):1–58, 2000.

Edward R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):57–69, Jan/Feb 2003.

Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proceeding of the 7th International Conference on Database Theory, Lecture Notes In Computer Science (LNCS 1540)*, pages 398–416. Springer, 1999.

Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*. AAAI/MIT Press, Cambridge, MA, 1991.

Craig Silverstein, Sergey Brin, and Rajeev Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2:39–68, 1998.

Alexander Strehl and Joydeep Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230, 2003.

Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, May/June 2000.

Mohammed J. Zaki. Mining non-redundant association rules. *Data Mining and Knowledge Discovery*, 9:223–248, 2004.

Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. Technical Report 651, Computer Science Department, University of Rochester, Rochester, NY 14627, July 1997.