

Grouping Association Rules Using Lift

Michael Hahsler

Department of Engineering Management, Information, and Systems
Southern Methodist University
mhahsler@lyle.smu.edu

Abstract

Association rule mining is a well established and popular data mining method for finding local dependencies between items in large transaction databases. However, a practical drawback of mining and efficiently using association rules is that the set of rules returned by the mining algorithm is typically too large to be directly used. Clustering association rules into a small number of meaningful groups would be valuable for experts who need to manually inspect the rules, for visualization and as the input for other applications. Interestingly, clustering is not widely used as a standard method to summarize large sets of associations. In fact it performs poorly due to high dimensionality, the inherent extreme data sparseness and the dominant frequent itemset structure reflected in sets of association rules. In this paper, we review association rule clustering and their shortcomings. We then propose a simple approach based on grouping columns in a lift matrix and give an example to illustrate its usefulness.

Keywords: Association rule mining, market basket analysis, clustering, visualization.

1 Introduction

The easy to understand problem statement of association rule mining (the so-called *support-confidence framework*) and the apparent applicability of this type of rules for many applications lead to a large body of research papers (see [11] for an overview). Association rules have the potential to be extremely useful for analysts (especially using visualization) and as input for other applications and models. However, a practical drawback of mining and efficiently using association rules is that the set of rules returned by the mining algorithm is typically too large to be directly used.

To address this problems, much research has been devoted to represent large set of frequent itemsets by smaller sets, e.g., maximal frequent itemsets, closed itemsets and non-derivable sets (see [7] for an overview). However, the size of these sets is typically still very large. Other approaches summarize a large collection of itemsets by profiles to approximately reconstruct the support value of the original itemsets [18, 17]. These approaches optimize reconstruction errors but do not aim at discovering useful group in sets of association rules.

Clustering association rules assigns each found rule to a group (from a small set of groups) such that rules in the same group are more similar than rules from different groups. Summary information about these groups can then be given to experts for manual inspection, they can be used for visualization or as the input for other applications. Since cluster analysis is a very well studied and understood problem in statistics and data mining it should be straight-forward to apply it to association rules. However, only very few papers about clustering association rules were published [13, 8, 16, 1, 3, 4] and hardly any applications include

association rule clustering. Over the years we implemented many of these clustering approaches as part of the R extension package called *arules* [9] freely available on the Comprehensive R Archive Network (<http://CRAN.R-project.org/>). We conducted many experiments, however, the results were typically not very encouraging. Here we discuss a different approach based on grouping columns in a lift matrix.

We start with a short introduction to mining association rules in Section 2. In Section 3 we discuss different approaches to cluster association rules and summarize the issues we have encountered. In Section 4 we discuss an alternative approach based on grouping columns of a lift matrix and present an illustrative application. We conclude with Section 5.

2 Association Rule Mining

Mining association rules was first introduced by Agrawal et al. [2] and can be defined as follows. Let $\mathcal{I} = \{i_1, i_2, \dots, i_n\}$ be a set of n binary attributes called *items*. Let $\mathcal{D} = \{t_1, t_2, \dots, t_l\}$ be a set of transactions called the *database*. Each transaction in \mathcal{D} has a unique transaction ID and contains a subset of the items in \mathcal{I} . A *rule* is defined as an implication of the form $X \rightarrow Y$ where $X, Y \subseteq \mathcal{I}$ and $X \cap Y = \emptyset$. The sets of items (for short *itemsets*) X and Y are called *antecedent* and *consequent* of the rule, respectively. Often rules are restricted to only have a single item in the consequent.

Association rules are rules which surpass a specified minimum support and minimum confidence. The *support* $\text{supp}(X)$ of an itemset X is defined as the proportion of transactions in the data set which contain the itemset and the *confidence* of a rule is defined $\text{conf}(X \rightarrow Y) = \text{supp}(X \cup Y) / \text{supp}(X)$. Therefore, an association rule $X \rightarrow Y$ will satisfy

$$\text{supp}(X \cup Y) \geq \sigma \quad \text{and} \quad \text{conf}(X \rightarrow Y) \geq \delta,$$

where σ and δ are the minimum support and minimum confidence thresholds, respectively.

Association rules are typically generated in a two-step process. First, minimum support is used to generate the set of all *frequent itemsets* for the data set. Frequent itemsets are itemsets which satisfy the minimum support constraint. Then, in a second step, each frequent itemset is used to generate all possible rules from it and only the rules which do satisfy the minimum confidence constraint are kept. Analyzing this process, it is easy to see that in the worst case we will generate $2^n - n - 1$ frequent itemsets with two or more items from a database with n distinct items. Since each frequent itemset can generate multiple rules, we will end up with a set of rules in the order of $O(2^n)$. Typically, increasing minimum support is used to keep the number of association rules found at a manageable size. However, this approach also prevents discovering potentially interesting rules with less support. Therefore, the need to deal with large sets of association rules is unavoidable when applying association rule mining in a real setting. A compression step (e.g., clustering as discussed in this paper) is needed to reduce the number for manual inspection, visualization or as the input for other applications.

3 Approaches to Clustering Association Rules

Lent et al. [13] introduced the idea of clustering association rules for the special case of quantitative attributes and presented an algorithm to join items with adjacent values for the same attribute. However, in this paper we look at the general problem of grouping rules.

The aim of clustering a set of m association rules $\mathcal{R} = \{R_1, R_2, \dots, R_m\}$ is to partition the rule set into k subsets $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ called clusters. Rules in the same cluster should be more similar to each other than to rules in different clusters. In the following we review existing approaches for clustering association rules.

Table 1: Comparison between Euclidean and Jaccard distance between rules in a data base with $n = 100$ items.

Rules	$d_{Euclidean}$	$d_{Jaccard}$
$\{bread\} \rightarrow \{butter\}$ $\{beer\} \rightarrow \{liquor\}$	2.00	1.00
$\{bread, milk, cheese\} \rightarrow \{butter\}$ $\{bread, vegetables, yogurt\} \rightarrow \{butter\}$	2.00	0.67

3.1 Clustering binary vectors

A set of m association rules \mathcal{R} can be seen as a set of n -dimensional vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$ where n is the total number of different items in the database and \mathbf{x}_i is the vector representation for rule R_i for $i = 1, 2, \dots, m$. Each vector \mathbf{x}_i contains a 1 at position j if rule R_i contains item j and a 0 otherwise.

A naive clustering approach would be to state the problem of finding groups of similar vectors as the well-known k -means problem which looks for a cluster assignment $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ which minimizes

$$\sum_{i=1}^k \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|_2,$$

the sum of squares of the differences between each vector and the centroid of the cluster it is assigned to. The cluster centroid $\boldsymbol{\mu}_i$ is defined as the mean of all points assigned to the cluster. Several efficient heuristics exist for the k -means problem. This procedure implies that similarity/dissimilarity between rules is measured by Euclidean distances of the binary vectors. The main problem of this approach is that Euclidean distances does not account for the fact that matching 1s (same items in the rule) are much more important than matching 0s. As an example, the Euclidean distance between two pairs of rules is given in Table 1. Interestingly, although we would instinctively say that the rules in the second pair are more similar since they have matching items, both pairs of rules produce the same Euclidean distance of 2 (the number of non-matching items). This obviously is not desirable and can be fixed by using a different distance measure. A well-known measure for binary data that focuses on matching 1s is the Jaccard index.

The Jaccard index is easily defined using set notation instead of binary vectors. Let X_i and X_j be the set of all items (union of antecedent and consequent) contained in R_i and R_j , respectively with $i, j \in \{1, 2, \dots, n\}$. Then the Jaccard distance is defined as

$$d_{Jaccard}(X_i, X_j) = 1 - \frac{|X_i \cap X_j|}{|X_i \cup X_j|}.$$

The distance simply is the number of items that X_i and X_j have in common divided by the number of unique items in both sets. For a set of m rules we can calculate the $m(m-1)/2$ distances and use them as the input for clustering. Table 1 shows how the Jaccard distances is able to differentiate between the two sets of rules and correctly identifies that the second set is more similar than the first one.

It is not straight-forward to compute centroids with Jaccard distances since it is only defined for sets and not for partial membership (e.g., the set contains item 1 50% of the time). Therefore, instead of k -means, k -medoids (the cluster center is restricted to be a data point) or hierarchical clustering is more appropriate. Next to using the Jaccard distance there are many other distance measures and clustering procedures available for binary data (see [14]). However, using clustering on the binary item data for association rules faces several problems. First of all, data sets typically mined for association rules are high-dimensional, i.e., contain many different items. This high-dimensionality is carried over to the mined rules and leads to a situation

referred is as the “course of dimensionality” where, due to the exponentially increasing volume, distance functions lose their usefulness [5]. The situation is even getting worse since minimum support leads in addition to relatively short rules resulting in extremely sparse data for clustering.

3.2 Common covered transactions

An approach that avoids the problem of calculating distances from high-dimensional, sparse data is to take the rules and look again at the original transactions. The idea is that rules which cover a similar set of transactions (i.e., the set of transactions that contain all items in a given rule) are more similar than rules which have no or only very few transactions in common. Toivonen et al. [16] define the distance between two rules with a common consequent, $X \rightarrow Z$ and $Y \rightarrow Z$, as

$$d_{Toivonen}(X \rightarrow Z, Y \rightarrow Z) = |m(X \cup Z)| + |m(Y \cup Z)| - 2|m(X \cup Y \cup Z)|,$$

where $m(X)$ is the set of transactions in \mathcal{D} which are covered by the rule, i.e., $m(X) = \{t \mid t \in \mathcal{D} \wedge X \subseteq t\}$. This measure computes the number of transactions which are covered only by one of the rules but not by both. A problem with this metric is that it depends on the database size and is expected to increase with the database.

Gupta, Strehl and Ghosh [8] define the distance between a pair of rules based on common covered transactions which avoids the database size dependency and does not require a common consequent. For X_i and X_j , the sets of all items in two rules this distance is defined as

$$d_{Gupta}(X_i, X_j) = 1 - \frac{|m(X_i \cup X_j)|}{|m(X_i)| + |m(X_j)| - |m(X_i \cup X_j)|}$$

Similarity is measured as the proportion of transactions which are covered by both rules in the transactions which are covered by at least one of the rules. A distance measure is created by subtraction the similarity from one.

Berrado and Runger [4] also use common covered transactions but do not calculate a distance measure. After mining a set of rules (with a common consequent) a new database \mathcal{Q} with items replaced by rules is constructed by using 1 to indicate that the rules antecedent is covered by the transaction and 0 otherwise. Now association rules restricted to a single item in antecedent and consequent are mined from this new database. These rules are called *metarules*. We denote here a metarule as $[X \rightarrow Z] \Rightarrow [Y \rightarrow Z]$. The described approach is equivalent to counting common covered transactions in the original database. The authors require

$$\begin{aligned} \text{supp}([X \rightarrow Z] \Rightarrow [Y \rightarrow Z]) &= \frac{|m(X \cup Y)|}{|\mathcal{Q}|} \geq \sigma' \quad \text{and} \\ \text{conf}([X \rightarrow Z] \Rightarrow [Y \rightarrow Z]) &= \frac{|m(X \cup Y)|}{|m(X)|} \geq \delta', \end{aligned}$$

where σ' and δ' are user specified minimum support and minimum confidence thresholds for metarules, respectively. A graph is formed with the original association rules as vertices and the metarules are directed edges. Groups of rules are discovered by looking at disconnected components. Changing σ' and δ' controls how connected the graph is and this allows some control over the number of unconnected components. Since $|m(X)|$ is the support count of X most of the needed counts for calculating the distance measures above were already produced during mining the frequent itemsets and only some values will still have to be counted from the database. Using common covered transactions avoids the problems of clustering sparse, high-dimensional binary vectors. However, it introduces a strong bias towards clustering rules which are

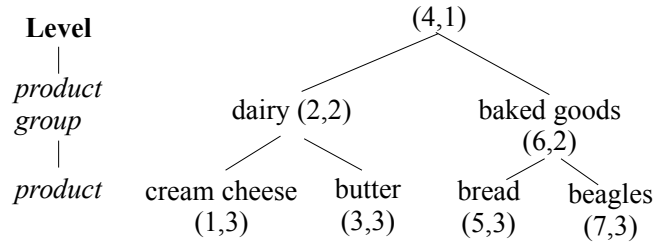


Figure 1: Item hierarchy example with relative semantic position in parentheses (horizontal and vertical position).

generated from the same frequent itemset. By definition of frequent itemsets, two subsets of a frequent itemset will both cover many common transactions. This bias will lead to mostly just rediscovering the already known frequent itemset structure from the set of association rules, which is not very useful.

3.3 Using item hierarchies

Adomavicius and Tuzhilin [1] introduce for their expert-driven validation of rule-based user models a similarity-based rule grouping approach. The authors utilize additional information in the form of an item hierarchy. For many applications such hierarchies exist or can be obtained from domain experts. For example, products in a retail store belong to a product group, category, etc. Rules are grouped by first selecting an appropriate level in the item hierarchy and aggregating the items at that level. That is, replacing each item label by the appropriate label higher up in the item hierarchy. Rules which now contain exactly the same aggregated item labels will be grouped. For example, given the item hierarchy in Figure 1 the two rules $\{butter\} \rightarrow \{bread\}$ and $\{cream\ cheese\} \rightarrow \{beagles\}$ are both grouped at the product group level as $\{dairy\} \rightarrow \{baked\ goods\}$.

This approach successfully reduces problems with high dimensionality and sparseness by aggregating the items into a smaller number of item groups. This approach also enables us to group rules with substitutes (e.g., bread and beagles) if all substitutes are members of the same product group in the hierarchy. However, rules have to match exactly to be grouped which does not address the situation where one has long rules containing items from different product groups. Also, very similar results can be obtained without grouping rules by simply aggregating the items to product groups already before mining association rules.

An et al. [3] also considers item hierarchies. Each node in the hierarchy is tagged with two integers, one indicating its horizontal position and the other its vertical position. A tagged hierarchy is shown in Figure 1. For each rule all items in the rule are translated to the associated pair of numbers called the *relative semantic position*. Rules are considered to be lines in a Cartesian coordinate system, where the averaged relative semantic positions of the antecedents and the consequents are the endpoints of the lines. Distance between rules is now defined by the angle between the lines and the distance between the center points of the rules. Using this distance a clustering algorithm can be used. A problem with this approach is that the horizontal order of items/groups in the hierarchy is an arbitrary artifact. For example, in the hierarchy in Figure 1 butter is closer to bread than cream cheese. However, the order of butter and cream cheese can be switched without affecting the meaning of the hierarchy. Now cream cheese is closer to bread. This effect becomes more pronounced if we rotate subtrees higher up in a multi-level hierarchy. Also taking the mean of the relative positions of all items in the antecedent seems arbitrary since the average position can easily be a product completely unrelated to the actual antecedent items. For example, assume that the antecedent contains butter and beagles and we use the relative semantic positions shown in Figure 1. Now the average of the semantic positions for the antecedent is exactly the same as a rule with the item bread in the antecedent.

3.4 Issues with clustering association rules

While clustering association rules would be useful, the clustering approaches share the following problems.

High dimensionality and sparseness. Association rules represented as binary vectors are extremely high-dimensional and sparse. The dimensionality is equal to the number of items in the database and association rules are typically used because the number of items is high with often thousands of items. Sparseness is created in association rules even for dense datasets automatically since combinatorial explosion prevents finding rules with many items and requiring a minimum support favors shorter rules. For 1000 items in the database and on average 5 items per rule, the binary vectors representing the rule set has only .5% non-zero entries. These observations make calculating distances directly from the rules questionable. Counting common covered transactions and employing expert knowledge in the form of item hierarchies can help to reduce the dimensionality and sparsity problem.

Substitutes. Grouping rules with substitutes, e.g., bread and beagles, is important since it mirrors the abstraction process used by humans. However this can only be done if it is known that both items are similar because they are both baked goods. Inferring this knowledge from the set of association rules or even the transaction database is extremely difficult since they only very rarely occur together in a transaction. Typically additional expert knowledge in the form of an item hierarchy is needed.

Direction of association. Most approaches do not differentiate between items in the antecedent and consequent of rules. However, two rules with the same items but different consequents have a very different meaning and probably need to be in a different group. Some approaches avoid this problem by only considering rules with the same consequent. Solving the general problem of grouping any set of rule is preferable.

Frequent itemset structure. This is probably the most important problem. Sets of association rules are generated from frequent itemsets which form a lattice that contains all subsets of each frequent itemsets. Also, for each frequent itemset several rules with a different item as the consequent might pass the minimum confidence constraint and form an association rule. Thus sets of association rules already have a very strong structure. Concise representations for frequent itemsets exploit this fact [7], however for grouping rules it is widely unused. Especially when common covered transactions are used, clustering association rules will just rediscover the frequent itemset lattice which is not very interesting since it is already known from the mining process.

Computational Complexity. Since we deal with large sets of association rules and many items, computational complexity makes the applicability of many clustering strategies impractical. For example, for distance-based hierarchical clustering just calculating and storing the distance matrix for a set of m rules requires $O(m^2)$ time and space. This severely limits the size of the set of rules which can be clustered.

4 An alternative clustering approach using lift

Here we pursue a completely different approach which inherently distinguishes between antecedents and consequents, and addresses high dimensionality and computational complexity. To guide clustering, we use the interest measure *lift* [6] defined as

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) supp(Y)}$$

Lift can be interpreted as the deviation of the support of the whole rule from the support expected if the items in both sides of the rule occur independently in transactions. A lift value of 1 indicates independence and larger lift values ($\gg 1$) indicate stronger association.

The idea for clustering using lift is that rules with antecedents that are strongly associated with the same set of consequents (i.e., have a high lift value) are similar and thus should be grouped together. For example, if the two rules

$$\{butter, cheese\} \rightarrow \{bread\} \quad \text{and} \quad \{margarine, cheese\} \rightarrow \{bread\}$$

have a similarly high lift, then the antecedents should be grouped. Even more evidence in favor of grouping would be provided if the two antecedents also occur in rules with similar lift values for other consequents like beagles or rolls. Compared to other clustering approaches for rules, this method enables us to even group antecedents containing substitutes which are rarely purchased together since they will have a similar dependence relationship with the same consequents. In the example above, rules with butter and margarine would be clustered. Clustering based on shared items or common covered transaction cannot uncover this type of relationship.

4.1 Definition

To perform this type of clustering we start again with the set of association rules, but separate antecedent and consequent and add the value of the interest measure lift. That is,

$$\mathcal{R} = \{\langle X_1, Y_1, \theta_1 \rangle, \dots, \langle X_i, Y_i, \theta_i \rangle, \dots, \langle X_n, Y_n, \theta_n \rangle\}$$

where X_i is the antecedent, Y_i is the consequent and θ_i is the lift value for the i -th rule, $i = 1, \dots, n$. In \mathcal{R} we identify the set of A unique antecedents and C unique consequent. We create a $A \times C$ matrix $\mathbf{M} = (m_{ac})$, $a = 1, \dots, A$ and $c = 1, \dots, C$, with one column for each unique antecedent and one row for each unique consequent. We populate the matrix by setting $m_{ac} = \theta_i$ where $i = 1, \dots, n$ is the rule index, and a and c correspond to the position of X_i and Y_i in the matrix. Now grouping rules becomes the problem of grouping columns and/or rows in \mathbf{M} .

Since for most applications the consequents in mined rules are restricted to a single item there is no problem with combinatorial explosion and we can restrict our treatment to only grouping antecedents (i.e., columns in \mathbf{M}). Note that the same grouping method can be used also for consequents independently or to both, antecedents and consequents, together leading to a biclustering problem [12].

\mathbf{M} will contain many empty cells since many potential association rules will not meet the required minimum thresholds on support and confidence. Studies have shown that the effect of support-based pruning is to remove mostly itemsets with negatively or poorly correlated items [15]. Thus we use a lift value of one, representing no relationship, to impute all missing values.

We define now the distance between two antecedents X_i and X_j as the Euclidean distance between their column vectors in \mathbf{M} .

$$d_{Lift}(X_i, X_j) = \|\mathbf{m}_i - \mathbf{m}_j\|_2,$$

where \mathbf{m}_i and \mathbf{m}_j are the column vectors representing all rules with antecedent X_i and X_j , respectively. We can cluster the set of antecedents into a set of k groups $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ using hierarchical clustering, k -medoids or we can directly cluster the columns by minimizing the within-cluster sum of squares

$$\underset{\mathcal{S}}{\operatorname{argmin}} \sum_{i=1}^k \sum_{\mathbf{m}_j \in S_i} \|\mathbf{m}_j - \boldsymbol{\mu}_i\|_2,$$

where μ_i is the center (mean) of the column vectors in S_i . The advantage of solving this k -means problem is that instead of creating a distance matrix of size $O(n^2)$, where n is the number of rules, we can work directly with \mathbf{M} of size $A \times C$ with typically both $A \ll n$ and $C \ll n$. Even more, a sparse representation of \mathbf{M} , where ones are not explicitly stored, can be used, reducing memory and computational requirements even more.

4.2 Illustrative Application

An important application of clustering association rules is to support decision making by enabling an analyst to explore large sets of rules via visualization. We have implemented a visualization based on the presented grouping approach in the R package *arulesViz* [9] and an example application for marketing was recently published in [10]. To visualize a set of rules, we first group the matrix by grouping columns based on lift and then we use a balloon plot with antecedent groups as columns and consequents as rows (see Figure 2). The color of each balloon represents the aggregated (median) lift in the group and the size of the balloon shows the aggregated (median) support. The number of rules and the most important (frequent) items in the group are displayed as the labels for the columns followed by the number of other items in that antecedent group. Furthermore, the columns and rows in the plot are reordered such that the aggregated interest measure is decreasing from top down and from left to right, directing the user to the most interesting group in the top left corner. For example, the first column in Figure 2 shows that rules with instant food products were grouped because of their relationship to hamburger meat. The second column shows a group around the item soda because of its relationship to salty snacks, white bread and bottled beer.

To allow the user to explore the whole set of rules further, we can create a hierarchical structure of subgroups. This is simply achieved by creating for each group $S_i, i = 1, \dots, k$, a submatrix \mathbf{M}_i which only contains the columns corresponding to the elements in S_i . Now we can use the same grouping process again on a submatrix selected by the user to “drill down” into the rule set.

5 Conclusion

In this paper we have reviewed methods of clustering association rules and identified a set of issues. The simple approach of grouping columns in a lift matrix overcomes many of these issues. In particular it is not directly impacted by high dimensionality, sparseness and the frequent itemset structure because it does not use the raw binary information. At the same time it has the ability to group substitutes and inherently distinguishes between antecedents and consequents of the rules. A simple visualization indicates that the approach is useful. The next step is to conduct a detailed experimental comparison.

References

- [1] G. Adomavicius and A. Tuzhilin. Expert-driven validation of rule-based user models in personalization applications. *Data Mining and Knowledge Discovery*, 5(1-2):33–58, 2001.
- [2] R. Agrawal, T. Imielinski, and A. 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.
- [3] A. An, S. Khan, and X. Huang. Objective and subjective algorithms for grouping association rules. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference*, pages 477 – 480, 2003.
- [4] A. Berrado and G. C. Runger. Using metarules to organize and group discovered association rules. *Data Mining and Knowledge Discovery*, 14(3):409–431, 2007.
- [5] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is “nearest neighbor” meaningful? In *ICDT '99: Proceedings of the 7th International Conference on Database Theory*, pages 217–235, London, UK, 1999. Springer-Verlag.

Grouped matrix for 5668 rules

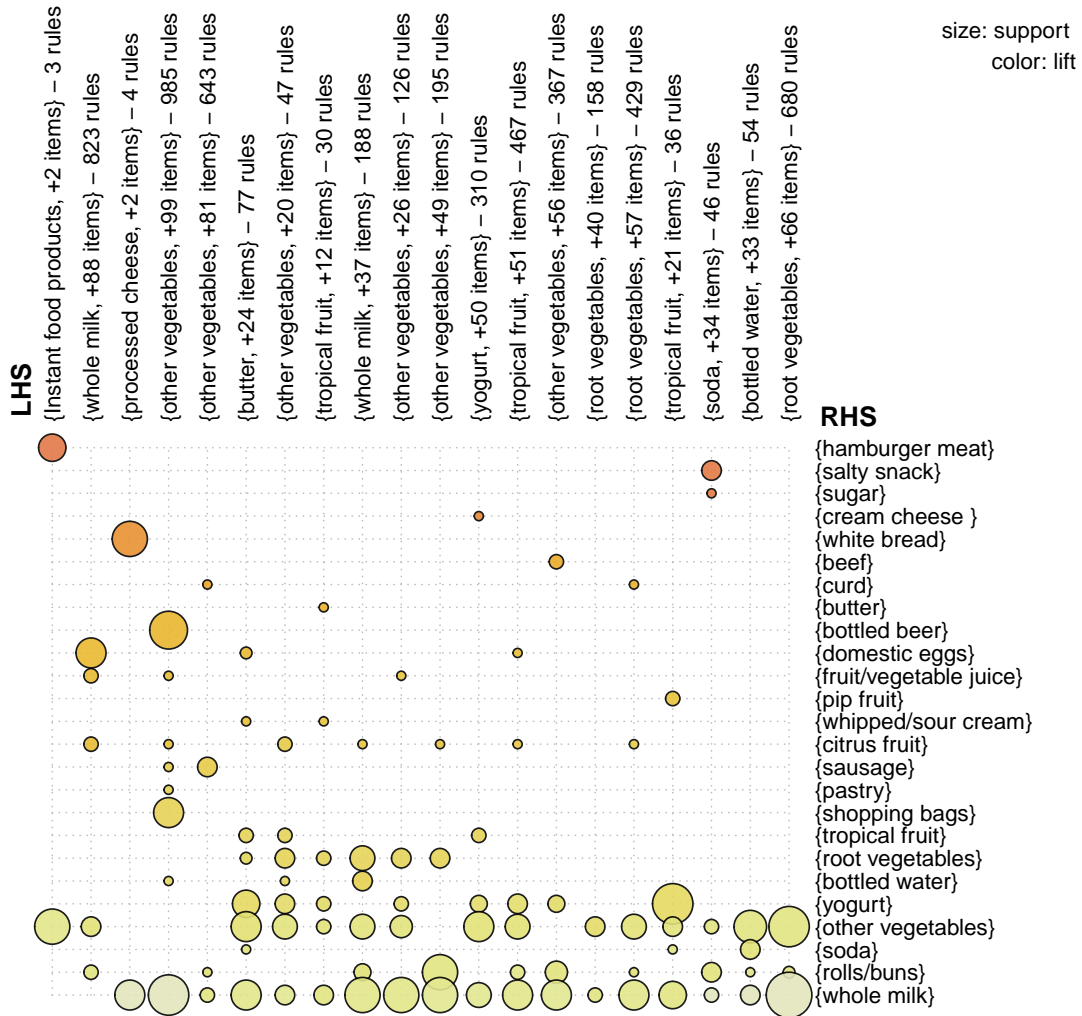


Figure 2: Grouped matrix-based visualization.

- [6] S. Brin, R. Motwani, J. D. Ullman, and S. 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.
- [7] T. Calders, C. Rigotti, and J. François Boulicaut. A survey on condensed representations for frequent sets. In *Constraint Based Mining and Inductive Databases, Springer-Verlag, LNAI*, pages 64–80. Springer, 2005.
- [8] G. Gupta, A. Strehl, and J. Ghosh. Distance based clustering of association rules. In *Intelligent Engineering Systems Through Artificial Neural Networks (Proceedings of ANNIE 1999)*, pages 759–764. ASME Press, 1999.
- [9] M. Hahsler, S. Chelluboina, K. Hornik, and C. Buchta. The arules R-package ecosystem: Analyzing interesting patterns from large transaction datasets. *Journal of Machine Learning Research*, 12:1977–1981, 2011.
- [10] M. Hahsler and R. Karpienko. Visualizing association rules in hierarchical groups. *Journal of Business Economics*, pages 1–19, May 2016.
- [11] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery*, 14(1), 2007.
- [12] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.
- [13] B. Lent, A. Swami, and J. Widom. Clustering association rules. In *Proceedings of ICDE*, 1997.
- [14] T. Li. A general model for clustering binary data. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 188–197. ACM, 2005.
- [15] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right objective measure for association analysis. *Information Systems*, 29(4):293 – 313, 2004. Knowledge Discovery and Data Mining (KDD 2002).
- [16] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. Pruning and grouping discovered association rules. In *Proceedings of KDD'95*, 1995.
- [17] C. Wang and S. Parthasarathy. Summarizing itemset patterns using probabilistic models. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 730–735. ACM, 2006.
- [18] X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: a profile-based approach. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 314–323. ACM, 2005.