ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

VISUALIZATION OF ASSOCIATION RULES

Student

KONSTANTINOS PETRELLIS

Student ID: it134069

Supervisor

PROFESSOR

DIMITRIS A. DERVOS

Date: 18/09/2020

Thesis Title: VISUALIZATION OF ASSOCIATION RULES

Thesis code number: 19058

Student full name: KONSTANTINOS PETRELLIS

Supervisor full name: DIMITRIS A. DERVOS

Thesis assignment date: 26-11-2020

Thesis completion date: 18-9-2020

I hereby declare that I am the original author of this thesis and that every asssistance that I received in the course of the project work reported is acknowledged and (where possible) cited in the "References/Bibliography" section of the present manuscript. All sources of information and/or data, ideas, pictures, graphs, figures and text excerpts used (be it "as is" quotations, or re-phrased), they are properly referenced in footnotes or directly in text. I also declare that, I myself have conducted the project work herewith reported and compiled the present manuscript, along the lines of fulfilling the BSc thesis project requirement of my study with the Department of Information and Electronic Systems Engineering.

The present BSc thesis manuscript and the project work reported comprise the intellectual property of the author, Mr. Konstantinos Petrellis. In the context of an Open Access Mandate1 , the author grants the International Hellenic University the permission to reproduce, lend, display publicly, and make it freely available for educational and/or research purposes. Open access to the full text of the thesis, does not imply, in any way, permission to abandon the author's intellectual property rights on the work reported, nor does it imply permission to reproduce, copy, republish, generate profit from, commercially exploit, re-distibute via platforms not belonging to the International Hellenic University, translate, edit small or extended parts of its content in any form, without the explicit permission from the author and his written consent.

Tha approval of this thesis manuscript and the project work reported should not be taken to imply that the Department of Information and Electronic Engineering, the School of Engineering, and the International Hellenic University share the personal views expressed by the author, nor should it be taken to imply that they are liable for his reported scientific and/or experimental findings.

Preface

During my study at the Department of Information Technology of the Alexander Technology Institution (now: the Department of Information and Electronic Engineering of the International Hellenic University), I had the opportunity to attend the course modules titled “Introduction to Data Analytics”, and “Data Organization and Data Mining”. They both motivated to further investigate the topics involved and acquire exploratory data analysis skills. In this respect, I realized the importance of the adaptive visualization approach in the interpretation, understanding and further exploitation of the information emerging in the output of the analytical processing of data. The present final year thesis project I have seen to comprise an opportunity to further develop my knowledge and skills in the above, plus an opportunity to put into practice my application design and programming skills in order to design and develop a web application that combines ease of use with practical functionality.

Abstract

The aim of this final year thesis project has been to consider the existing R/RStudio methods that make possible the visualization of the Market Basket Analysis (MBA) output, namely the association rules. The latter reflect and quantify causal inter-relations between groups of items (i.e. itemsets) of the (any) real life situation that can be modeled along the lines of the Market Basket Analysis paradigm. For example, items or item categories purchased in super market or e-commerce transactions, web pages visited during the same session by internet users, etc. Having conducted this initial investigation, a number of issues allowing for further improvement have been identified. Cases whereby the user who who wishes to navigate himself in the graphical representation of the association rules or zoom in to selected subsections in search of specific information.

Considering the above, a new proposal (method) for visualizing the association rules output has been designed and developed: ARNet. Operating as a web application, ARNet accepts as input the association rules output of a (any) MBA type analytical processing and makes it possible for the user to explore it interactively (to a certain extend) in order to both interpret and exploit the information presented. The proposed ARNet method is compared to the existing visualization methods, R's ruleExplorer in particular. It is also tested in practice, utilizing data from two real life cases: (a) by considering item categories purchases in super market transactions, and (b) by considering (course, student performance) pairs present in examination grades of the IHU Information and Electronic Engineering Department's post-graduate program titled "MSc in Web Intelligence".

Acknowledgments

I would like to express my deepest appreciation to my supervisor, Professor Dimitris A. Dervos, who has the attitude and the substance of an excellent teacher. He continually guided me to the depths of Market Basket Analysis and Data Mining.

Special thanks to Mr. Leonidas Pispiriggas for his assistance during the development of ARNet. I also wish to thank Mr. Konstantinos Kelesidis who has made available to me excerpts of his postgraduate thesis findings, in order to visualize them via ARNet.

Finally, I would like to thank my family for their support during my studies all these years.

Contents

Contents

Figure Index

Figure Index

Figure Index

Abbreviations

LHS	Left Hand Side
RHS	Right Hand Side
UI	User Interface
MBA	Market Basket Analysis

Abbreviations

Introduction

Nowadays, a large amount of data are available from the Internet, data warehouses or any data source. Data can be retrieved in different forms, and often involves data inter-dependencies, co-relations, and patterns. With the current state of technology, it is possible to collect large amounts of data and process them effectively. The data collected may be analyzed and this reveals relationships and regularities that facilitate the solution of many data-relating tasks/problems.

In order to foster such data-relating solutions, governments, private companies, large organizations and business in general utilize are using data mining implementations. One may simply define data mining as a process that involves searching, collecting, filtering and analyzing the data [1]. Once information is extracted from data, the challenge to be bet is to effectively communicate it to people. That's where visualization comes into play. Different data and different desired results led programmers to develop a variety of packages/libraries of methods and structures, which a programmer can utilize in order to visualize his results in way that makes possible the extraction of useful information.

In this thesis is we consider existing R/Studio methods that make possible the visualization of the association rules mining output. In addition, we utilize existing tools in order to further improve the interactivity of the user interface of the graphical representations in questions.

This thesis is organized as follows. Chapter 1 discusses the data mining process and everything the reader has to know in order to understand the graphs to be considered in the sequel. Chapter 2 presents visualization methods included in the *arulesViz* package[1]. Chapter 3 begins focusing on contribution made by the current work, namely the utilization of the ***shiny R***[2] and ***visNetwork***[3] packages to develop a web application, ARNet, that facilitates the creation of unique interactive graphs. In chapter 4, existing visualization methods are compared and contrasted to the proposed new approach, in relation with the type of information one wishes to extract from the association rules mining output. Chapter 5 discusses two use cases, in which different end users may consult *ARNet* visualization outcome in order to take the right decisions. Finally, chapter 6 summarizes the conclusions made by this thesis and suggests some improvements for *ARNet*.

1https://cran.r-project.org/web/packages/arulesViz/index.html

2https://shiny.rstudio.com

3https://cran.r-project.org/web/packages/visNetwork/index.html

1 Data Mining

1.1 Introduction

Data collection, cleaning, processing and analysis are part of data mining. The results of this process are useful information, such as patterns that can explain the data behavior. Usually, people think that data mining is only relating to marketing, helping companies discover the new trends and achieve better sales. However, one interesting case is of the coordination of banking services in order to reduce their costs by detecting credit card frauds [2].

Although data mining is a process with great results, there are issues that data mining analysts come across frequently. First of all, the raw data can be arbitrary, unstructured or in a format that is not suitable for automated processing. Also, the wide disparity in the problems and data types make it harder for data mining analysts to develop automated algorithms [3]. One last issue is the increment of data volumes. With large streams of data, lots of funds are needed to store them effectively. The stated issues may be tackled by one or more of the following four data mining techniques: Association rules mining, clustering, classification and outlier detection.

This chapter is organized as follows. Section 1.2 considers the main steps of the data mining process. Basic data types are presented in section 1.3 , while in section 1.4 three out of four dominant techniques of data mining are addressed. Finally, section 1.5 introduces the Association rule mining process by addressing their importance, structure and some of the most important interest measures.

1.2 The Data Mining Process

As discussed earlier, data mining involves a number of discrete, sequential, stages such as data cleaning , feature extraction and algorithmic design. But before any of these stages is initiated , data need to be collected.

Data collection is the first stage of the mining process. Although, it is not the analyst's job to decide how the data will be collected, for example using specialized material, user service or a software tools, making a good decision on how to collect data can have a significant impact on the data mining process.

Following the collection of data, an analyst should be able with a little guidance to extract the features that are most relevant to the goal of an application. For example, to detect a credit card fraud, the usual amount of money transacted, the frequency of transactions, locations or stores are good indicators to be monitored. At this point, the analyst has selected and extracted the data in order to start cleaning them. Missing or erroneous values in data will be estimated or corrected or even dropped. This results to a data structure which might not be yet ready for the algorithms. There is a chance that some features are noisy and they may introduce errors to the data mining process. Therefore, different methods are used to remove irrelevant features. Another method, is *data transformation* [3] where altering the attributes of a data set might be useful, for example if we divide the age attribute values from a data set into categories like teenagers , adults , middle-aged we could analyze the data in a more convenient way.

Figure 1.2 1 outlines the sequence of the stages pertaining to the data mining process. During the analytical processing stage, there are different building blocks. Building blocks represent the solution

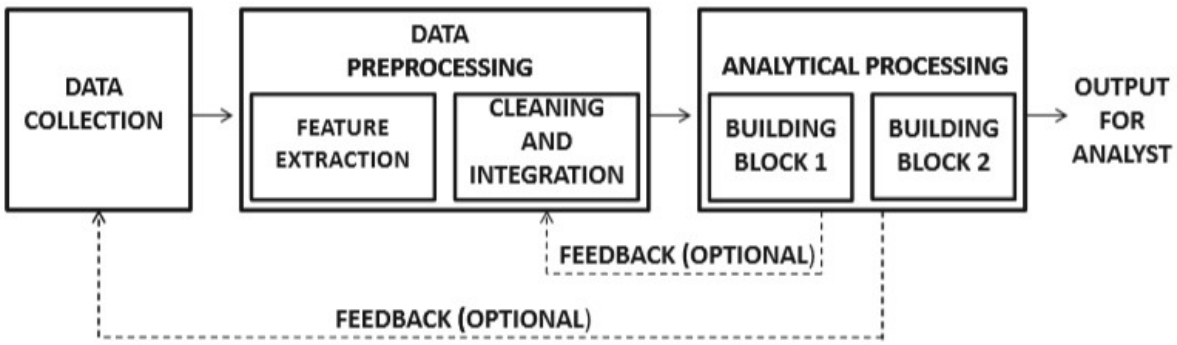


Figure 1.2 1: The data mining process pipeline.

of the current task of the analyst. This could be the creation of clusters or discover associating patterns. No matter, what the challenge is, the previous steps and phases are most likely the same.

A detailed example presented in [3] , demonstrates the data mining process: a retailer wants to make targeted product recommendations for customers on his web store. He has access to web logs and to a demographic information within the retailer database. So, there are two sources with completely different formatted data that can be used. First thing to do is to take a look at the logs format:

98.206.207.157 - - [31/Jul/2013:18:09:38 -0700] "GET /productA.htm HTTP/1.1" 200 328177 "-" "Mozilla/5.0 (Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10B329 Safari/8536.25" "retailer.net

There is little information that is of direct interest to the retailer. The IP, helps him to identify the customers, and the productA.htm. Even if the logs are filtered there would be much unnecessary information. In the feature extraction process, the retailer decides to create a new record for each customer. Each record contains an attribute that refers to the number of accesses to each product description. More attributes will be added that contain demographic information. Therefore, the logs should be processed and the number of accesses need to be aggregated during this *feature extraction* phase. Now that the retailer's data set is ready, the analyst decides that the best way of making recommendations, is by constructing groups of similar costumers. A user is placed into a group based on the product descriptions he tends to access more frequently. Products accessed by other customers in the same group may then be recommended to the user.

In the example above, the data analyst constructed groups of similar customers on the basis of features they have in common. There are more real-life problems where association patterns, classification and outlier detection comprise the means for analysis and data mining, like the ones considered in section 1.3.

1.3 Basic Data Types

As mentioned earlier, data come in various forms and types. Despite their nature there is one feature that separates them into two types, data interdependence.

"Nondependency-oriented data: This typically refers to simple data types such as multidimensional data or text data" [3].In these types of data, there are no dependencies between the data records or the

attributes. A simple example is an attendance book. The attributes of this report would be the names of the attendants, their gender and their attendance id.

"Dependency-oriented data: In these cases, implicit or explicit relationships may exist between data items" [3]. Explicit relationships can usually be found in graphs or networks, where data items are represented with nodes and their relationships with edges. Implicit relationships may exist in continuous measurements. Having a data set with consecutive records of online users in a website, it is expected that successive records are similar.

Non-Dependency-Oriented Data

Multidimensional data is the simplest form of data, with a simple structure as follows:

- Each data object contains a set of records.
- Each record consists of a set of fields or attributes or features.
- Each field should contain a value, suitable for the field's nature.

A definition for multidimensional data set is defined in [3]:

"*A multidimensional data set D is a set of n records, ..., such that each record contains a set of d features denoted by (...).*"

The following subtypes, follow the above structure and are named based on the type of their fields. Depending on their type of fields, we can separate them into the following type of data:

- *Quantitative Data*, where all of the fields are numeric.
- *Categorical Data*, where fields values take on *discrete unordered* values [3] .
- *Mixed Attribute Data*, where some fields contain numerical values and others contain a categorical value.
- *Binary Data*, where the value of a field can be one of at most two discrete values, regardless of their type(numeric/categorical). In some cases, the value of the field could be a flag, that indicates whether this entry should be included in the set. In this case, we call them *Set Data*.
- *Text Data*, where the attributes are the words found in a document, known as terms, and the values are their frequency.

Each type of data has its own benefits and can offer an analyst an ease during the processing. For example, quantitative data are better from a statistical perspective while categorical data can be helpful seperating data to groups. When it comes to real applications though, analysts come across a more complex form of data with a combination of different attributes.

Dependency-Oriented Data

Working with data without any dependencies between data items, is not always the case. In fact, there are types of data, where data items have either implicit or explicit dependencies with each other. As in non-dependency-oriented data, we separate them by their attribute's types:

- *Time-Series Data*, where each record consists of two features. One would be a time stamp and the other one would be a numeric value. A continuous measurement of temperature could be a

good example, with a sensor keeping track of temperature values between fixed time periods. The temperature's values will not be much different when we compare successive records, which indicates the interdependence of time.

- *Discrete Sequences and Strings*. The structure of the data is similar to *Time-Series Data*, with the difference that the numerical attribute is now a categorical. In some cases, instead of categorical attributes, there unordered categorical values. For example, different categories of groceries that form a customer's basket.
- *Spatial Data*. In this type of data, the measurements are based on the distance between two locations. The rest of attributes could be either numerical or categorical. A simple is example could be the identification of the best locations for opening new hospitals based on the population of patients who live in each neighborhood. It must be noted, that there is a subtype called *Spatiotemporal Data*, where records at the same location are re-recorded over time.
- *Network and Graph Data*. As the name implies, our data is a graph that data entries are represented with nodes and are connected by edges, which represents the relationships between them.

1.4 Data Mining Techniques.

As already mentioned in section 1.1. clustering, classification, outlier detection and association pattern mining comprise the four major data mining techniques. Each one may be to correspond to a challenge the data analyst faces in accordance with real life data analysis tasks at hand. The characteristics, the requirements, and the limitations of each one technique are briefly considered in the next lines.

- Clustering

In Clustering, data are to be grouped together based on their similarity without any supervision. This means that clustering algorithms are not trained to recognize data groups before they process the data. The most common problem among clustering algorithms is that many features may be noisy or redundant for the cluster analysis. For this reason, they have to be removed. To decide which features should be removed, there are two categories of models performing feature selection. *Filter models* and *Wrapper Models*. *"In filter models, a specific criterion is used to evaluate the impact of specific features, or subsets of features, on the clustering tendency of the data set"* [3]*."Wrapper models use an internal cluster validity criterion in conjunction with a clustering algorithm that is applied to an appropriate subset of features"*[3]. A variety of clustering algorithms are developed, with K-means algorithm being the most famous.

- Classification

While clustering is the method of putting similar data points into groups without any supervision, the classification is the method of identifying to which of a set of categories, a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known. To classify a new data point, a model is required. The model is trained with a part of data instances, already partitioned into different classes. The rest of them, are used to test the model. The first step of classifications is the feature selection. The most relevant features to class labels, are selected by different methods [3]. Some of the most known classifiers are Decision Trees, Rule-based classifiers and Probabilistic classifiers.

- Outlier Detection

While clustering tries to determine groups of similar data points, outliers are individual data points that do not fit in any group. Hawkins defined an outlier as follows:

"*An outlier is an observation which deviates so much from other observations as to arouse suspicious that it was generated by a different mechanism*."

The utility of outlier detection can be found in several cases. For example, during the data cleaning phase outliers often represent noise in the data, and they should be removed. There are cases though that outliers, or abnormalities as they are called, can be a tool for identifying unusual patterns. A simple example is the detection of credit card fraud [2]. In most outlier detection methods, a model of normal patterns is created. Such model may include clustering, distance-based quantification, or dimensionality reduction. To identify outliers, algorithms produce an output for each data point. The output could be either a numeric value or a binary label. Models may produce *specialized types* [3] of outliers based on a very restricted model of normal patterns.

1.5 Association Pattern Mining

Association pattern mining was first introduced in the context of market basket analysis (MBA). This method focuses on finding association between frequent patterns. In MBA's universe, data refer to a set of transactions. Each transaction, contains sets of items that represent the products in a customer's basket. Each set of items is called an itemset which corresponds to a frequent pattern. Transactions are usually presented in a string form. For example, if the products in a basket were bread, butter and milk, the transaction would have this form *{Bread, Butter, Milk}* or *{11,12,13}* where numbers refer to product IDs.

In the case of the market basket analysis paradigm, retailers are interested on which products are sell well together, and moreover, they are interested in exploiting causality in product sales. More specifically, they want to identify cases whereby customers tend to purchase certain items as a consequence of having already purchased another set of items (cause). This kind of information is presented with *Association Rules*. Each rule involves a *body* (cause) and a *head* (effect) with an arrow ($\Rightarrow$) . For example, *{Bread, Butter}* $\Rightarrow$*{Milk}* where the rule body is the itemset *{Bread, Butter}* and the rule head is the itemset *{Milk}*. In order to quantify the validity (strength) of an association rules, a number of measures have been introduced.

Association rules emerge in the output of the Market Basket Analysis processing of data organized in transaction. The probability of a typical transaction to contain all the items in the body and in the head of the rule is defined to comprise the rule's *Support* measure. It is noted that the *Support* is the only measure that applies both to rules and itemsets.

"The support of an itemset I is defined as the fraction of the transactions in the database $T = \{T_1 \ldots T_n\}$ *that contain I as a subset" [3]*. Formally, the *Support* of an itemset is defined as follows: $supp(I) = count\frac{(I)}{T}$

.

.

All the of rest association measures apply only to rules. The most known is *Confidence* and *Lift*. *Confidence* could be defined as follows:

"Let X and Y be two sets of items. The confidence conf () of the rule is the conditional probability of occurring in a transaction, given that the transaction contains X. Therefore, the confidence conf (X ⇒ Y) is defined as follows: $conf(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$ *"* [3].

In other words, *Confidence* implies the probability a group of products being purchased together. Possible values of *Confidence* range between 0 and 1. When the rules have the same *Confidence*, their *Support* values help analysts to decide which is more valuable.

The dependency level between the itemsets of a rule could determine how much important the rule is. A measure, called *Lift*, is used for this purpose.

"Lift measures how far from independence are A and C. It ranges within [0,+∞[. Values close to 1 imply that A and C are independent and the rule is not interesting. Values far from 1 indicate that the evidence of A provides information about C. Lift is defined as: $lift(A \rightarrow C) = \frac{conf(A \rightarrow C)}{supp(C)}$ *"* [4].

In this thesis, association rules were generated with the *Apriori* algorithm. *Apriori* algorithm uses the downward closure property of itemsets. This property is described in [5] as follows:
*"**Theorem1***
If an itemset is supported, all of its (non-empty) subsets are also supported.

Proof1
Removing one or more of the items from an itemset cannot reduce and will often increase the number of transactions that it matches. Hence the support for a subset of an itemset must be at least as great as that for the original itemset. It follows that any (non-empty) subset of a supported itemset must also be supported.". Thus, if an itemset is not supported then its subsets are also not supported. In *Apriori* algorithm, user specifies a threshold of support for the itemsets. *Apriori* algorithm is displayed inFigure 1.5 1 and works as follows:

Assume that each individual item, extracted from the transactions, is a candidate frequent itemset. Frequent itemsets are categorized, based on its length *k*, in k-frequent itemsets.

At the beginning, algorithm counts the support of all candidate itemsets with $k=1$. Those with lower *Support* than the threshold value, are removed and the rest are the frequent 1-itemsets (F_1). Length of the itemsets is increased by 1 $(k=k+1)$ and for the construction of frequent 2-itemsets, pairs of two items are created based on the frequent 1-itemsets. Candidate itemsets are then pruned based on the threshold value. The remaining itemsets are the frequent 2-itemsets. In order to create candidate 3-itemsets, algorithm pairs frequent 2-itemsets with one item in common. Again, candidate itemsets are pruned contrasting their support values with the threshold value. The process continues with joining frequent k-itemsets to create C_{k+1} itemsets and prune those who do not meet the user specifications. The process will stop when no C_k itemsets remain after the pruning.

Algorithm *Apriori*(Transactions: $\mathcal{T}$, Minimum Support: *minsup*)
begin
  $k = 1$;
  $\mathcal{F}_1 = \{$ All Frequent 1-itemsets $\}$;
  while $\mathcal{F}_k$ is not empty **do begin**
    Generate $\mathcal{C}_{k+1}$ by joining itemset-pairs in $\mathcal{F}_k$;
    Prune itemsets from $\mathcal{C}_{k+1}$ that violate downward closure;
    Determine $\mathcal{F}_{k+1}$ by support counting on $(\mathcal{C}_{k+1}, \mathcal{T})$ and retaining
      itemsets from $\mathcal{C}_{k+1}$ with support at least *minsup*;
    $k = k + 1$;
  end;
  return$(\cup_{i=1}^{k} \mathcal{F}_i)$;
end

Figure 1.5 1: The Apriori algorithm.

1.6 Summary

In this chapter, an introduction to Data Mining process was made. Data categories were described and discussed. Also, the most known data mining methods were presented with most attention given to the Association rule mining process. Apriori algorithm was introduced and explained how the algorithm generates association rules and the meaning of every interest measure of the generated rules.

2 Rules Visualization and Web Application Development in R

2.1 Introduction

Following the association rules discovery stage, analysts wish to visualize the mined association rules. With a graphic representation of the data, analysts can make new observations and be guided to conduct further analysis on the data. For the graphic representation of the rules, graphs or networks are generated that may include graphic marks, images, geometric shapes with different attributes, depending on the data values. Moreover, in networks data are visually connected in order to reveal pre-existing interconnections. The extraction of new information from the data is delayed if the generated graph or network is static. For this reason, libraries that provide interactive graphs or networks are combined with libraries that provide an interactive *User Interface*(UI), in order to develop applications that allow analysts to view and filter their data in real time. Due to the existing varieties of the data and the large number of the mined rules, several graphs and networks have been implemented and will be considered in this chapter.

This chapter is organized as follows. Section 2.1 introduces the graphic visualization of association rules using methods and functions directly implemented in R. In section 2.2 libraries and technologies that provide an interactive UI are presented and discussed.

2.2 Association Rules Visualization Methods in R

The association rules are mined from the R-provided Groceries[4] data set, which contains 9835 transactions. The items are classified to belong to 169 categories. The association rules visualization was performed using methods and functions that are already implemented in the *arulesViz*[5] package. There are four major visualization methods in *arulesViz* package that can be used to visualize association rules: *scatterplot*, *matrix*, *grouped* and *graph* methods. Each on of them is usually passed as an input parameter to *plot*[6] function. Moreover, by specifying the parameter *engine*, the programmer can set the interactivity to the generated graph.

By selecting *scatterplot*, the *plot* function will generate a scatter plot. A scatter plot comprises a two-dimensional graph, in which association rules attributes are placed on its axes. -"Scatter plots focus on interest measures, and rules with similar values for these measures are placed close to each other"- [6]. The most common interest measures to be used for the graph's axes are ***Support*** and ***Confidence***. Each association rule is represented by a dot in the two-dimensional space and additional interest measures may be encoded via color shading, such as the ***Lift***.

Figure 2.2 1 demonstrates the visualization of the mined association rules using the scatter plot method. In this graph, the axes represent the *Confidence* and the *Support* values of the visualized rules. The intensity of each dot's color shading encodes the *Lift* value. By hovering the pointer over the donts in the graph, the user can see all available information on the rule, its *LHS* and *RHS* values, for

4https://rdrr.io/cran/arules/man/Groceries.html

5https://cran.r-project.org/web/packages/arulesViz/index.html

6https://www.rdocumentation.org/packages/arulesViz/versions/0.1-1/topics/plot

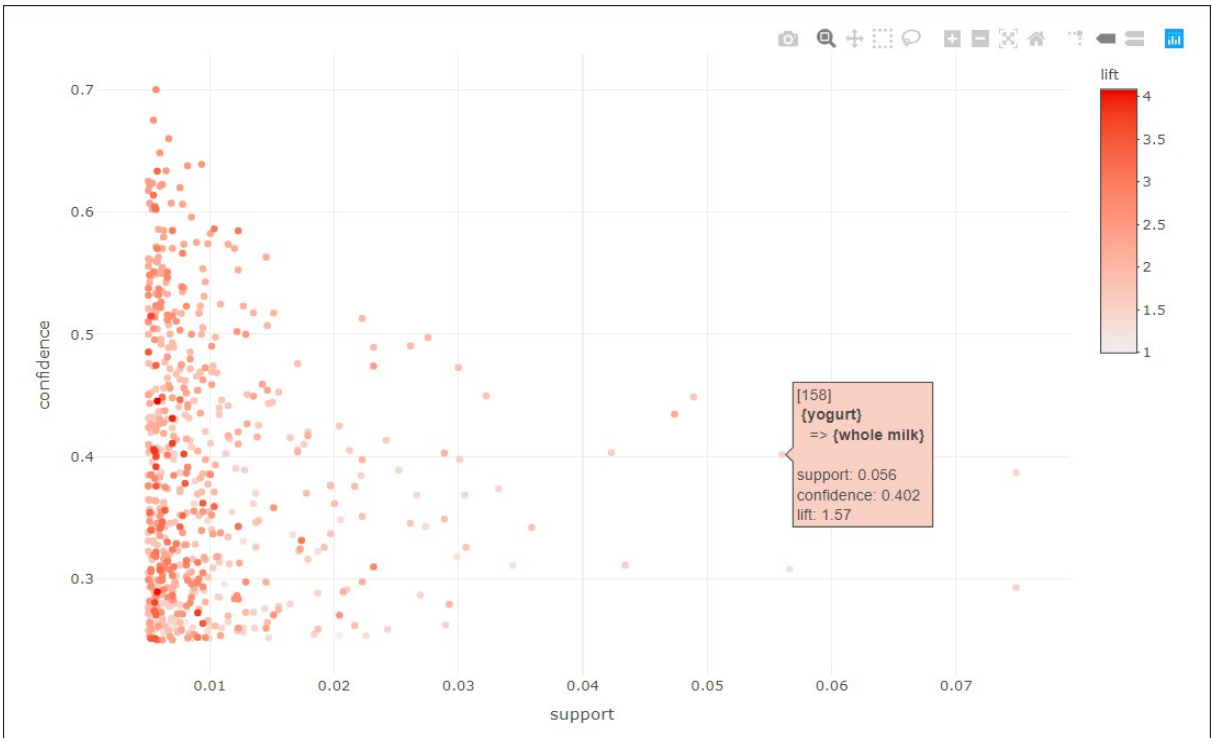


Figure 2.2 1: *Scatter* visualization method from arulesViz.

example. By observing the graph in Figure 2.2 1 for example, one notes that rules tend to be more interesting in the low *Support* values.

Figure 2.2 2 shows an alternative use of the rules color attribute, where each color corresponds to a different rules length. It is be noted, that the *plotly* package provides an extended version of the *plot* function, the *plot_ly*[7] function. The latter enables the user to manipulate other attributes of each point on the graph, such as their size and their shape.

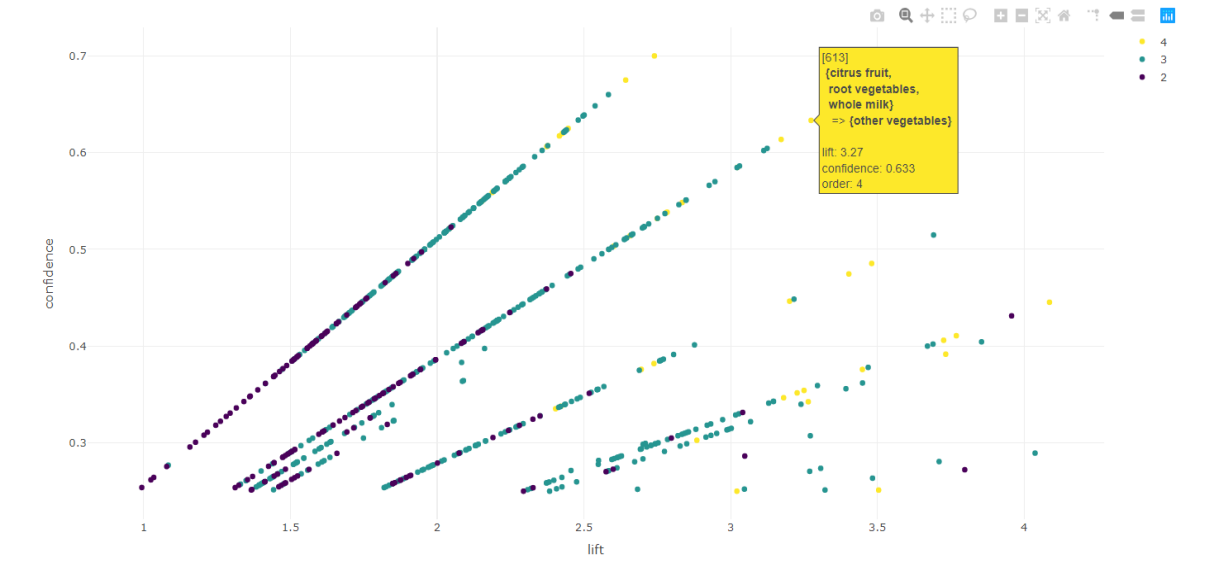


Figure 2.2 2: *Two-key* visualization method from arulesViz.

Two-key visualization method provides a clear view of the relationship between the attributes of the association rules even for large numbers on the latter. In *scatterplot* and *two-key* methods, the number of the visualized rules is 635. The advice is to keep the number of the rules under 1000 [6]. Apart from the ability of handling a large number of rules and make new observations about the data, there is no available visual information about the itemsets involved. Also, the interactivity of the visualization approach considered decreases as the number of visualized rules increases. With large numbers of rules, dots tend to overlap with each other and in order to inspect inspect a certain rule demands high accuracy of the mouse pointer.

Matrix method produces a two-dimensional graph with unique itemsets ,which are extracted from the *LHS* and the *RHS* of every visualized rule, placed as the values of the columns and the rows, respectively. The best order of rows and columns is described as follows: *"The order of rows and columns is arbitrary, however, to improve the ability to analyze the data, we suggest in arulesViz to reorder the matrix such that the row averages decrease from top to bottom and the column averages decrease from left to right. This pushes the rules with higher values of interestingness to the top-left position in the plot."* [6].

Each row contains rules that have their RHS in common and each column contains rules with the LHS in common. It is recommended to keep the number of visualized rules under 1000. In Figure 2.2 3, only 146 rules are visualized with Lift higher than 2.5 and no ordering was applied. With this visualization method, analysts may view rules related in with a certain itemset, ordered by their *Lift* values. The color intensity may represent another interest measure such as *Support* or *Confidence*. A drawback in this visualization method is the lack of information about each row and column. Due to

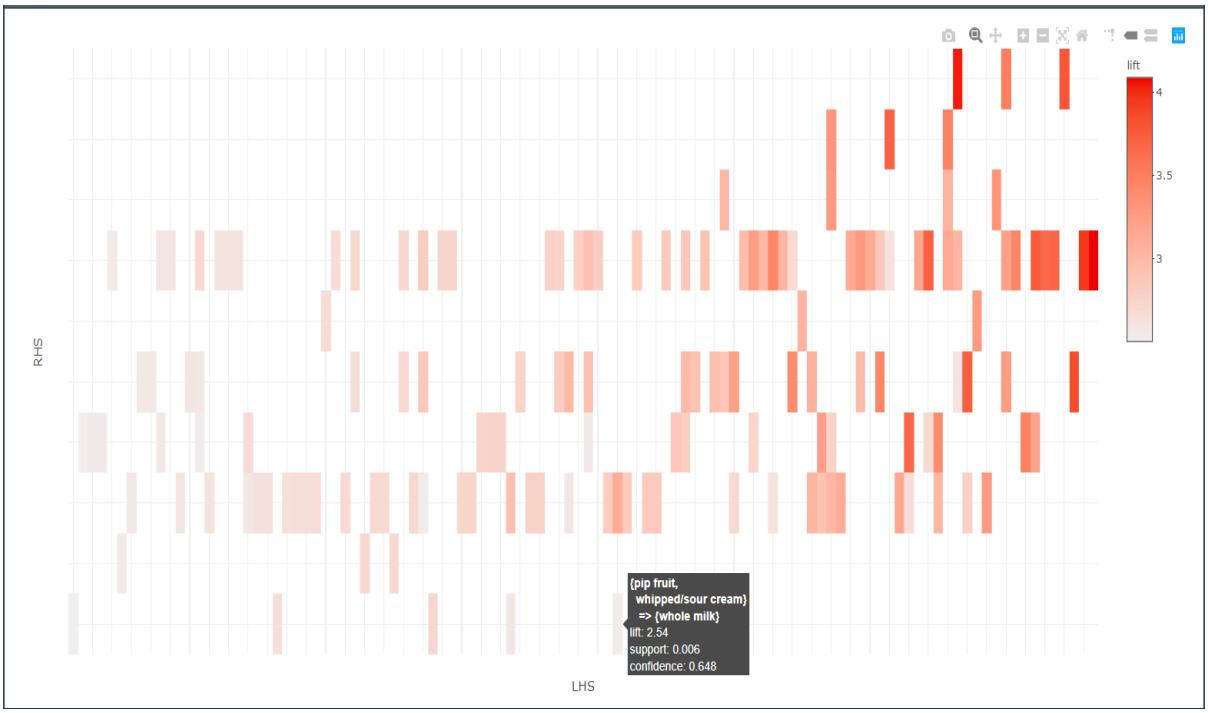


Figure 2.2 3: *Matrix* visualization method from arulesViz.

the arbitrary order of the rules, there is no clear guide point who declares which row corresponds to

each rule. Another drawback of this methods, is the limit of the number of the association rules that may be visualized. The more rules are displayed, the thinner the rectangles ,that represent them, become which leads the user to spend time by zooming in and out, in order to have a clear view of the graph.

Grouped method provides a significantly better way to visualize a large numbers of rules. The unique consequent items are placed at the Y axis of the graph. Antecedents are grouped in clusters based on their consequent. A circle in the graph represent a cluster of rules that have their consequent in common. Interest measures can be used to specify the size or the color of each one circle. It must be noted, that *Apriori* algorithm generates association rules with a single item at their RHS. In case that association rules would have more than one item included in their RHS, this visualization method would not perform.

Figure 2.2 4 and Figure 2.2 6 demonstrate grouped matrix visualizations with the 100 rules visualized having the highest values of *Lift*. The size of each circle is indicative of the average *Lift* value of each cluster, and the color is indicative of the average *Confidence* value of each cluster. At Figure 2.2 4 the user may chooses to inspect a column with a single circle, obtaining a result like the one shown in Figure 2.2 5. In this case there is just one circle in the column, thus the only consequent item listed is {yogurt}, which is common in every rule. In Figure 2.2 6, the user decides to inspect a matching point from a column with multiple circles. This act returns eight rules, whereby their antecedents can be found individually in the group {tropical fruit, root vegetables, +2 items} and their consequent are

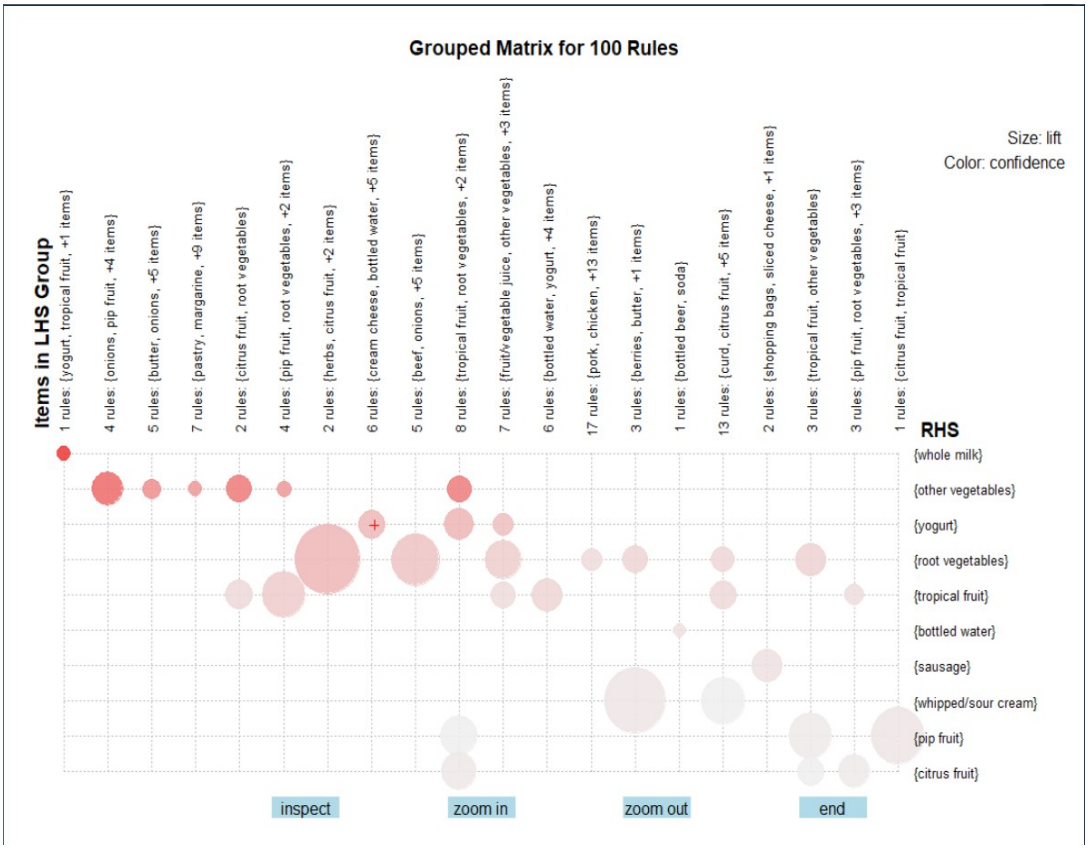


Figure 2.2 4: Selecting a cluster from a *grouped* visualization method from arulesViz.

```
Selected rules:
     lhs                          rhs          support confidence     lift count order
[1] {curd,
     tropical fruit}           => {yogurt} 0.005286164  0.5148515 3.691395    52     3
[2] {fruit/vegetable juice,
     other vegetables,
     whole milk}               => {yogurt} 0.005082850  0.4854369 3.480498    50     4
[3] {cream cheese,
     whole milk}               => {yogurt} 0.006607706  0.4012346 2.876782    65     3
[4] {curd,
     whole milk}               => {yogurt} 0.010064044  0.3852140 2.761917    99     3
[5] {cream cheese,
     other vegetables}         => {yogurt} 0.005286164  0.3851852 2.761710    52     3
[6] {bottled water,
     tropical fruit}           => {yogurt} 0.007115991  0.3846154 2.757625    70     3
```

Figure 2.2 5: Inspecting the selected cluster in Figure 2.2 4.

either {other vegetables}, {yogurt}, {pip fruit} or {citrus fruit}. The results are displayed in Figure 2.2 7.

Figure 2.2 8 demonstrates the ability of visualizing 5000 rules with a high clarity of every circle. In this example, redundant rules were not removed but they were selected, based on their Lift values. Despite the ability to visualize even a large number of rules, the associations between a particular consequent and its antecedents is not available, since the antecedents are grouped together. The

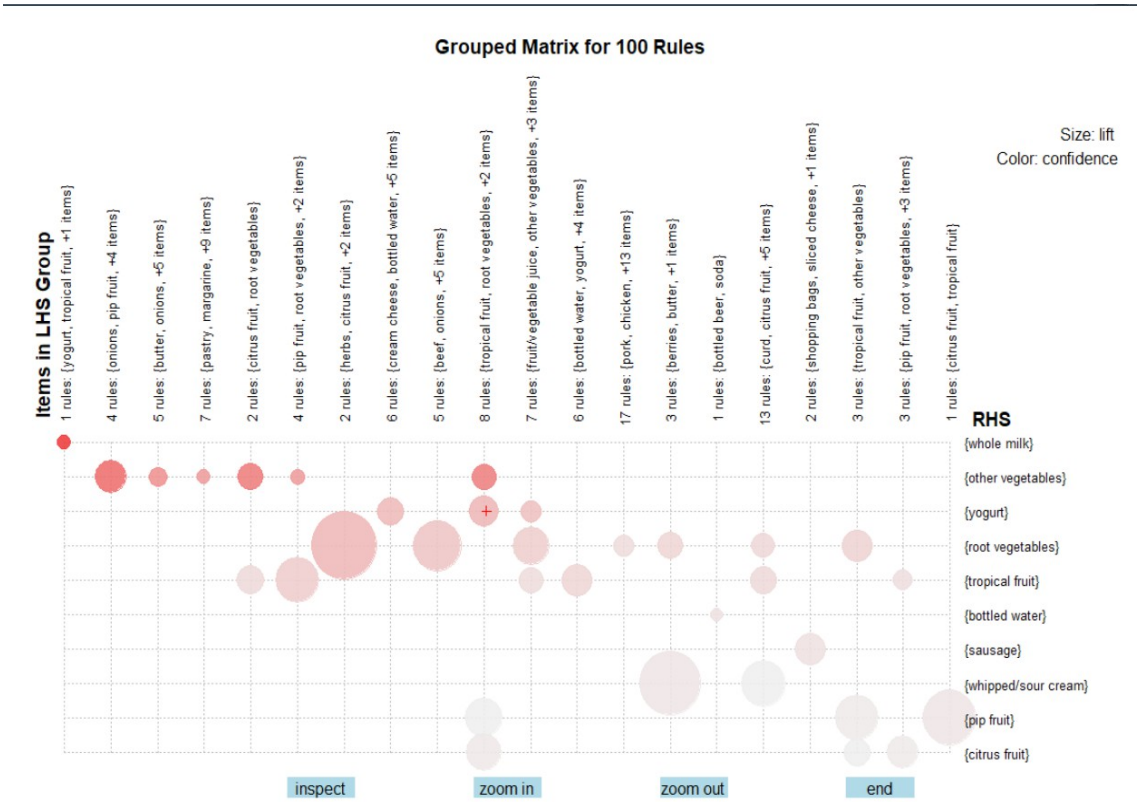


Figure 2.2 6: Selecting one of the column's clusters from *grouped* visualization method from arulesViz.

```
Selected rules:
    lhs                                                rhs                support     confidence lift     count order
[1] {root vegetables,tropical fruit,whole milk} => {yogurt}           0.005692793 0.4745763  3.402629  56    4
[2] {root vegetables,tropical fruit}            => {pip fruit}        0.005286164 0.2512077  3.321412  52    3
[3] {root vegetables,tropical fruit}            => {citrus fruit}     0.005692793 0.2705314  3.269309  56    3
[4] {tropical fruit,whipped/sour cream}         => {yogurt}           0.006201078 0.4485294  3.215877  61    3
[5] {root vegetables,tropical fruit,whole milk} => {other vegetables} 0.007014334 0.5847458  3.022672  69    4
[6] {root vegetables,tropical fruit}            => {other vegetables} 0.012300498 0.5845411  3.021613 121    3
[7] {tropical fruit,whipped/sour cream}         => {other vegetables} 0.007827590 0.5661765  2.926683  77    3
[8] {root vegetables,tropical fruit}            => {yogurt}           0.008132561 0.3864734  2.770947  80    3
```

Figure 2.2 7: Inspecting the results from Figure 2.2 6

interest measure of each rule is also not available, since the size and the color of each circle is respective to an average interest measure value of the included rules in a circle.

Graph method focuses on individual items in the rule set and their interconnections. This method generates a network instead of a graph, which consists of *Nodes* and lines with arrows, the so-called *Edges*. Each *Node* can be an individual item or a rule and the *Edges* indicate whether an individual item is part of the antecedent or the consequent of a rule, depending on their direction.

Figure 2.2 9 shows the visual representation of an association rule in the *graph* method. Tropical fruit and curd are the antecedents of the rule because the arrows are pointing to the rule. Yogurt is the consequent of the rule with an arrow starting from the rule and pointing to the individual item. The *size* of each *Node* is relative to its *Support* value, and the color intensity is relative to its *Lift* value.

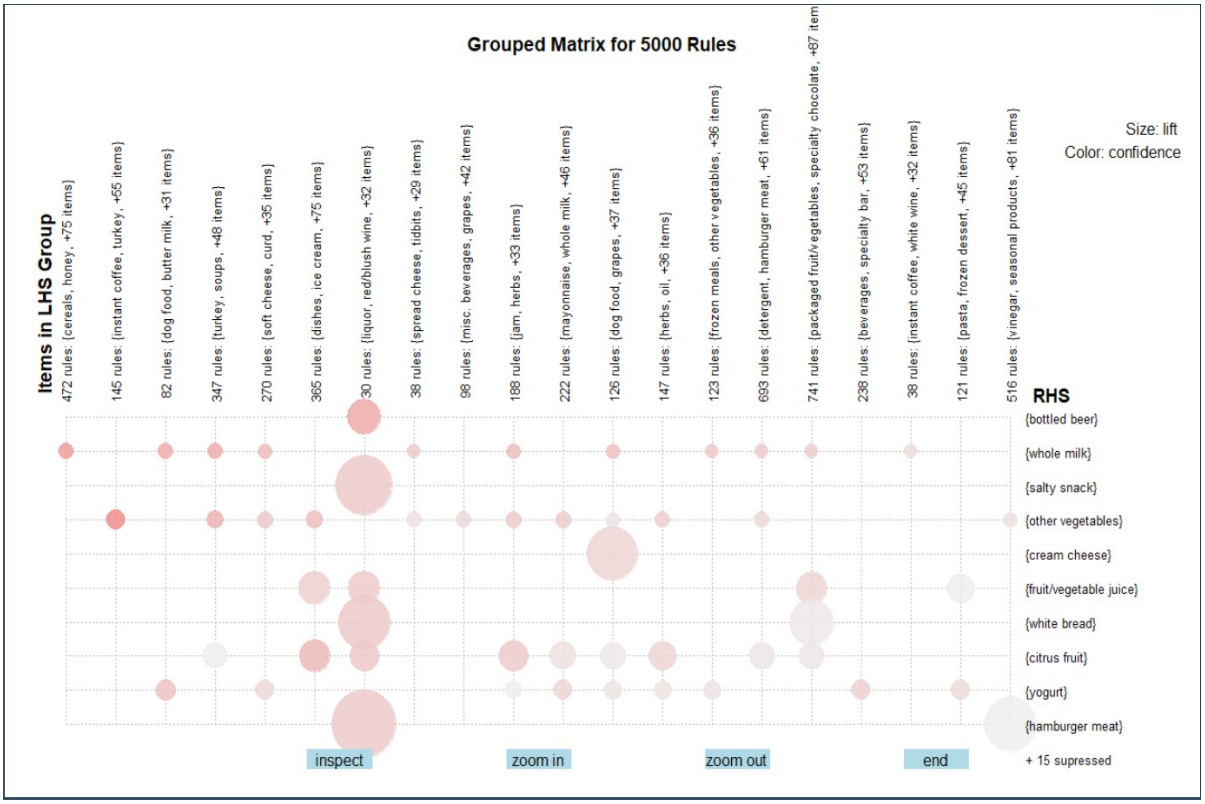


Figure 2.2 8: Visualizing 5000 rules with *grouped* visualization method from arulesViz.

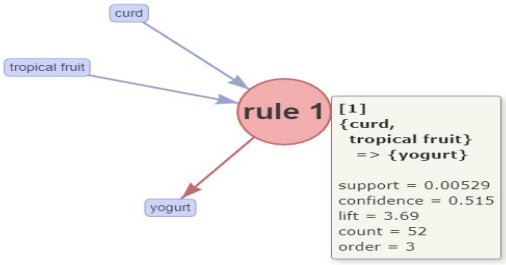


Figure 2.2 9: The visual representation of a rule with *graph* visualization method from arulesViz.

Several packages are used to facilitate interactivity with the network such as *visNetwork*[8] and *Rgraphviz*[9]. Using this method, the LHS and the RHS of a rule are discernible and interest measures are used to specify the *size* or the *color* attributes of a *Node*. Moreover, the items are placed among rules based on whether they are part of their antecedent or their consequent components. This visualization method is not appropriate for visualizing a large number of rules. Visualizing a large number of rules with graph method has a negative impact on the performance of the computer. The recommended number of rules is strongly advised to be under 100 [6]. The number for optimal results should be around 20 like in Figure 2.2 10. A larger number would lead the user to "drag & drop" selected sections of the network in Figure 2.2 11.

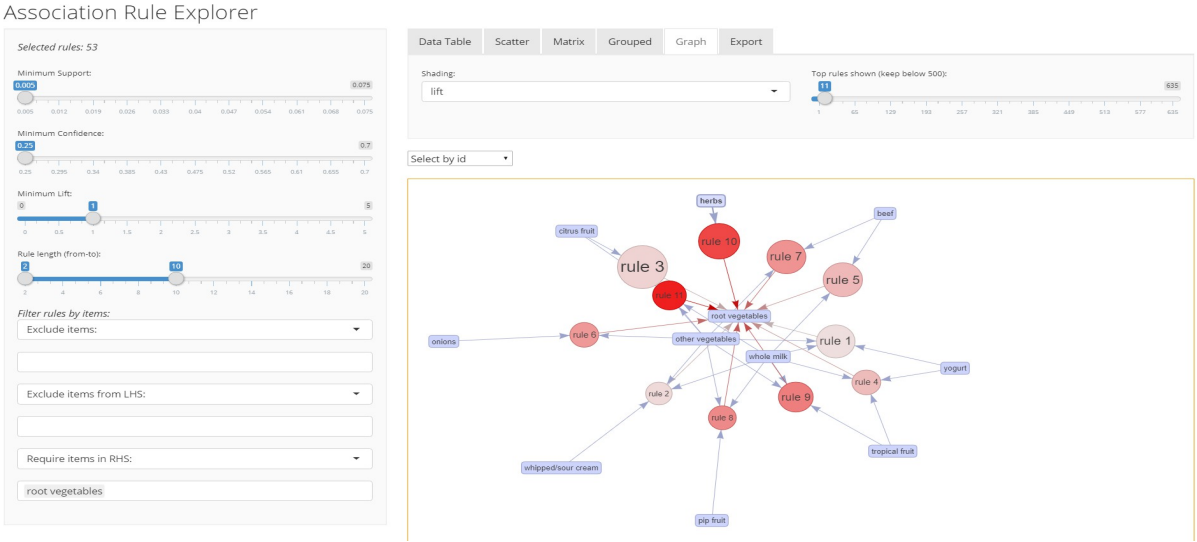


Figure 2.2 10: Visualizing 20 rules with *graph* visualizing method in arulesViz.

8https://cran.r-project.org/web/packages/visNetwork/vignettes/Introduction-to-visNetwork.html

9https://www.bioconductor.org/packages/release/bioc/html/Rgraphviz.html

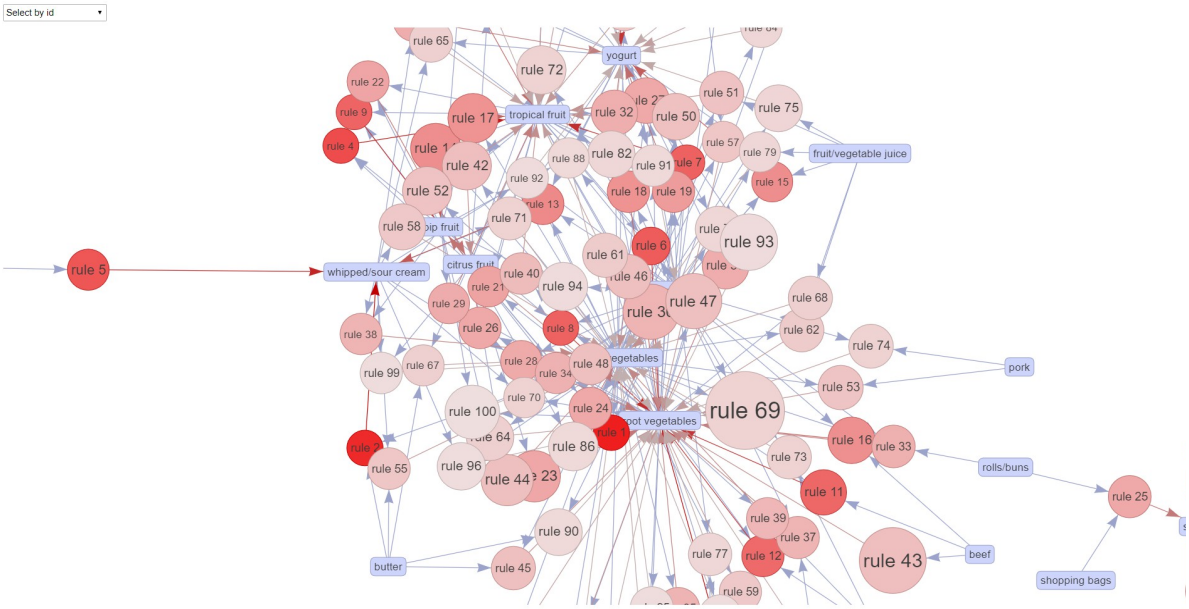


Figure 2.2 11: Visualizing 100 rules using *graph* method from arulesViz.

2.3 Building Interactive Web Applications in R.

An interactive graph or network can provide users the option to obtain more information about the data being displayed. For example, with the *scatterplot* method users can not view the antecedents and the consequent of a rule, unless they hover the mouse over the respective dot. Due to the large number of data, the option to filter, order and interact with the data is necessary. Furthermore, it may be desirable that the visualized data be accessed by analysts and users remotely, which in turn requires the development of a web application. Shiny is an R package that allows the development of interactive web applications, accessed users at any time from anywhere.

Each Shiny application consists of two parts. The *User Interface*(UI) object and the *Server* function. Both are passed on as arguments to the *shinyApp*[10] function. Inside the UI object, the layout instructions take place and every element that appears at the user's screen has to be included, while in the Server function, the reactivity and functionality of the application is handled by the developer.

Two types of functions are used in order to create elements at the users screen and apply changes in the data, *Input* and *Output.* Each function returns an input object and an output object respectively. Input functions are used to create HTML widgets for the user inputs values, while output functions are used to create objects that displays the outcome after the data processing, with the most usual objects to be graphs, networks and tables. Input functions have parameters that are necessary for the construction of a widget, such as an *id*, which is a unique string value inside the code, and others that are related to the widget's type. For example, for slider inputs *max*, *min* and *selected* parameters must be specified while in radio-buttons inputs, instead of *max* and *min* parameters, the developer has to specify the available choices. The selected values can be accessed inside the code using the object *input*.

10https://shiny.rstudio.com/reference/shiny/1.0.5/shinyApp.html

For the output functions, the only necessary parameter is the *ID*, which will be used as a reference to the input object, in the server function to render the data after their processing. When values from input objects are used inside an output object, any change on their values will re-execute the code in which they were used, changing the final output. The re-execution of the code is happening because of the Shiny's reactive programming model. For the reactivity of a shiny application, three objects are responsible, reactive sources, reactive endpoints and reactive constructors. Reactive sources and reactive endpoints could be described as the user inputs and the application outputs.

In order to explain how reactivity works in Shiny, consider a numeric slider input object as the reactive source with ID "Nrules". This input has by default the value 100. For the reactive end point, consider a tableOutput object with ID "table" that displays a number of rules from a static table equal to the input$Nrules value. The input object plays the role of a reactive value while the output object contains a reactive expression that plays the role of the observer. The observer is an expression that includes the reactive value input$Nrules and contains a flag that indicates if the observer is clean or dirty. When the observer is clean, it is considered as descendant of the input object. When the reactive value changes, a few events trigger. First of all, the observer is flagged as dirty. Therefore, he is not considered a descendant of the input object. Secondly, the dirty observer requests the new value from the input object and re-executes. Finally, the output object will return a new view of the static table and the observer becomes again a descendant of the reactive value. The same events occur when there more reactive values and reactive expressions.

In some cases, the number of reactive expressions can cause issues at the development of the application and its performance. A big number of reactive values, used for a common outcome, makes difficult to track an unexpected error. Consider the case, where the output object is a table. The table has to be updated based on the user activity, so the number of the reactive values to consider is related to the latest user actions. If an unexpected outcome appears, the developer will have to track down every reactive value by reproducing every action of the user and observe their changes during the execution of the program. The tracking of these errors could take much time and the solution of them may not solve similar future errors. Now consider a single reactive value to be a part of many reactive expression. A single change on this value will lead to the execution of every related reactive expression. This can reduce the responding speed of the application, which is one of the main features of Shiny, and reach the point that is not responding.

These issues can be resolved by a few methods. First of all, a reactive value can be changed without triggering any further events. This can be done by using the *isolate*[11] function. *isolate* allows the execution of a reactive expression without flagging the observers dirty. This is useful, when a change of a reactive value is needed and use it later on. Another method, is to observe events, using the *observeEvent*[12] function which allows the developer to track the changes of reactive values and handle them as he wants, instead of include them in reactive expressions at output objects.

Another tactic that can be followed in order to avoid reactivity issues, is to utilize JavaScript events. The communication between R and JavaScript, is done with JSON objects. In the case of communicating from R to JavaScript, an R object is encoded to JSON by using the session$sendCustomeMessage[13] function. The function takes two parameters, *type* and *message*. The

11 https://shiny.rstudio.com/reference/shiny/0.11/isolate.html

12 https://shiny.rstudio.com/reference/shiny/1.0.3/observeEvent.html

13 https://shiny.rstudio.com/reference/shiny/1.4.0/session.html

first parameter is an identifier string which will be used by Shiny to execute the proper JavaScript code. The second parameter is an R project that will be encoded to JSON and decoded at the JavaScript side. At the JavaScript side, a JavaScript function is registered to receive messages of the given type. Using the *Shiny.addCustomMessageHandler* function, the developer can utilize the value of the sent R object and apply changes to the User Interface with a custom JavaScript function. In the case of communicating from JavaScript to R, the JavaScript function *Shiny.setInputValue* will create an input object with a reactive value, encode it to a JSON object and sent it to the server side. In the server side, an input *handler*[14] needs to be registered to receive messages of the given type. Even without an input handler in the server side, an input object is created and the developer can use it as a reactive value.

arulesViz includes the *ruleExplorer*[15] function. The latter, utilizes the Shiny package and builds a web application. The data file containing the association rules comprises an input parameter to the *ruleExplorer* function. The association rules may then be visualized with every available method in the *arulesViz* package. Moreover, an interactive UI is provided to the users that allows them to filter and sort the data and then visualize the rules. The a*rulesViz* UI consists of three slide bars that allows users to set limits on rules interest measures. One slide bar provides the option to set a limit on the *length* of the rules. The *length* of a rules is the number of items it contains. Moreover, there are six drop-down boxes that users can use to include or exclude items from the association rules. Finally, graphs and networks that are generated from *arulesViz* visualization methods are in a tabular form. In each tab, users can find more widgets which they may use to apply additional filters. Interacting with *ruleExplorer* UI , users are able to filter the rules, and overcome some of the drawbacks of *arulesViz* visualization methods. For example, using the *"Filter rules by items"* widgets in Figure 2.3 1, users can have an isolated view of rules that are related to certain itemsets. This overcomes the drawback of

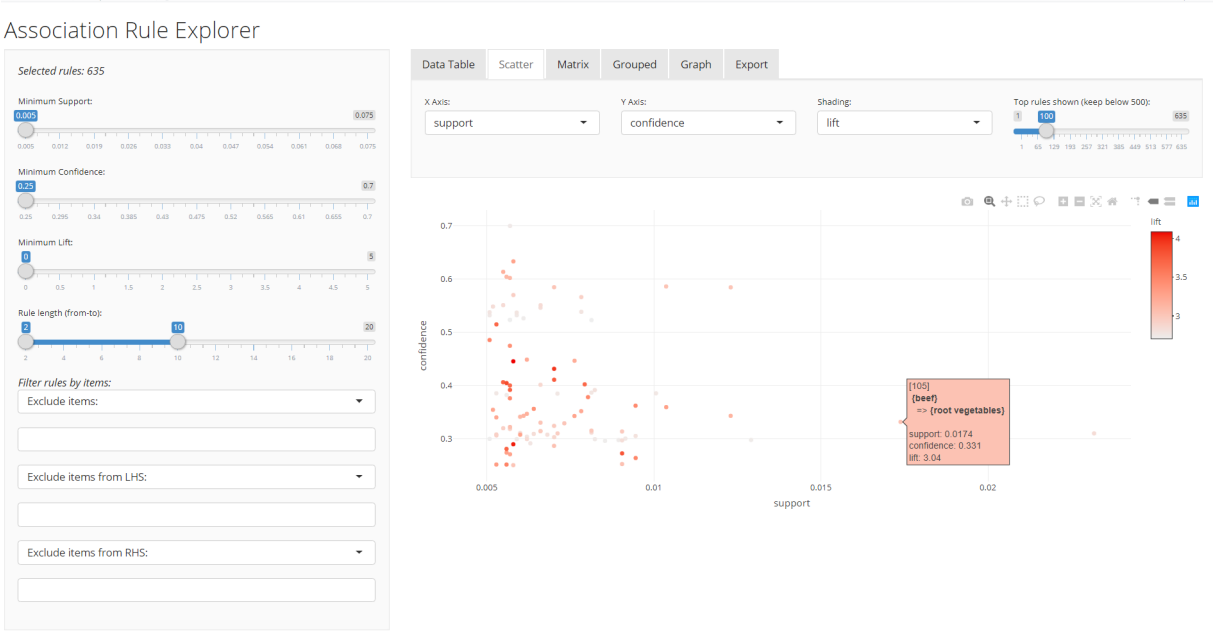


Figure 2.3 1: The ruleExplorer User Interface.

14https://shiny.rstudio.com/reference/shiny/0.12.0/registerInputHandler.html

15https://rdrr.io/cran/arulesViz/man/shiny.html

the *scatterplot* method, which does not takes in account the LHS and RHS of the rules. Finally, *ruleExplorer* provides the option to export the graphic visualization of the rules and practically store instances of the data for further analysis.

2.4 Summary

This chapter has focused on the, *arulesViz* package and the visualization methods it contains. Each one of the latter is suitable for certain kinds of applications. The *Scatterplot* method can visualize a large number of rules. The *Matrix* method can be used to find the most important rules relating to a certain consequent when the number of the rules is relatively small, whereas the *grouped* method can be utilized for the same purpose when the number of rules is large. Last but not least, with the *graph* visualization method analysts can focus on individual ascendant and consequent rules components. The available technologies that used to implement the interactivity between users and the graphic visual representation of data have also been presented. Technologies utilized by the *ruleExplorer* method in order to provide a unique reactive environment.

3 The ARNet Application

3.1 Introduction

As mentioned earlier, each method implemented in R can not visualize every available information about association rules. Every visualization method in *aruleViz* takes into account a maximum number of two interest parameters of the rules and only in few methods the antecedents and the consequent of a rule are part of the visualization. The main contribution of this thesis is the development of a web application Association Rules Network (ARNet), utilizing Shiny and *VisNetwork* to visualize association rules in the form of an informative network the user may interact with. Also, in ARNet is able to display three interest measures of the rules in a two dimension network. Further more, ARNet makes possible for the user to specify desired values on standard itemset or rule measures in order to filter or sort the selected rules.

This chapter is organized as follows. Section 3.1 introduces *VisNetwork* package. It also discusses how someone may generate a network and handle its interactivity. Section 3.2 discusses how *VisNetwork* was utilized in order to provide to the users an interactive network. Section 3.3 introduces ARNet *User Interface* and presents its most special features.

3.2 VisNetwork

The *VisNetwork* package offers a unique visualization method that each developer can utilize to generate interactive networks. The main function utilizes two parameters, the *Nodes*[16] and the *Edges*[17]. Each *Node* represents an element of the data, and *Edges* represent node interconnections. Internally, *Nodes* and *Edges* are two data frames[18] that comply to certain specifications imposed by the code that creates the network. Several parameters may be specified to include more information in the network.

In the *Nodes* data frame, the field that needs be specified for network creation is the one that registers a unique node identifier, “id”. Furthermore, specifying a label or a title to each node helps identify it in the network. Additional node customizing may be applied by specifying the size and the color of each one node. This way, a developer can visualize additional information about the association rules, represented by the connected nodes. There are two ways to specify values for each one node. One way is to set values to the desired parameters on each node individually at *Nodes* data frame creation time. Alternatively, nodes may be organized in groups, using the group parameter, and then the desired settings be specified for each one group. Finally, the visual presentation of each node may be controlled by specifying appropriate *shape* and *image* values fro it. For example, assuming {whole milk} is an itemset and a part of an association rule, an icon showing a glass of milk could replace the label parameter. Additional parameters can be found in the *visNodes*[16] documentation.

In the *Edges* data frame, “from” and “to” fields ,which contain *Nodes* “id”, values need to be specified in order to create the interconnections between nodes and therefore, to display a completed the representation of an association rule. A developer has the option to specify the width and the length of each *Edge* if he decides to present additional information about the connection of the linked *Nodes*.

16https://www.rdocumentation.org/packages/visNetwork/versions/0.1.0/topics/visNodes

17https://www.rdocumentation.org/packages/visNetwork/versions/2.0.9/topics/visEdges

18https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/data.frame

Label and *title* parameters are available too. Specifying the *arrow* parameter, the antecedent and the consequent of an association rule can be clear to the user. In any other case, the *arrow* parameter can be empty or be used to show other kind of relations between linked nodes. Last but not least, *smooth* parameter defines the curvature of each edge, having an impact on the drawing performance of the network when its value is TRUE and the number of *Edges* is large. Additional parameters can be found at visEdges[16] documentation.

Every interaction with the main components of a *visNetwork* network can be handled by *visEventsError: Reference source not found* function. The behavior of each triggered event can be customized by assigning JavaScript activities to a selected event. Events generally are related with the *Nodes* and the *Edges* of the graph like ,*selected* event, which will trigger when the user clicks on a certain *Node*. Utilizing this event, one can get every information stored in *Nodes* data frame and display it to the users screen. The same amount of information can be retrieved by using the *hover* event which is very similar. There are also events related with the network. The most noticeable is the *stabilized* event, which can be used to provide feedback to the users when the network creation process is over and ready to provide every interaction with it. Finally, each event can be set to trigger once, trigger every time after a specific action, or be disabled.

VisNetwork function generates a network, in which physics play a crucial role for its performance. The physics of the network can be specified using the *visPhysicsError: Reference source not found* function. There are four *solvers* available that allow the developer to change the gravity attribute of the network. *BarnestHut, Repulsion, hierarchicalRepulsion* and *forceAtlas2Based and* Each solver has its own parameters. For example utilizing the *avoidOverlap*, *centralGravity* and *gravitationalConstant* parameters of *barnestHut*, the gravity between the *Nodes* can be handled and *Nodes* will not overlap with others. Also, the force that drags *Nodes* to the center of the network may be specified, in order to prevent them from moving outside of the users screen.

One last feature that VisNetwork provides is the option to specify the layout of the network. Using *visLayout*[19] function, *Nodes* can be presented in a hierarchical tree layout. This way, the strongest or the most frequent association rules may be placed at the top of the tree. It is also possible to create a layout that exposes further interconnections between rules.

3.3 Utilizing VisNetwork in ARNet

In order to visualize all the available information about association rules in a two dimensional network, ARNet utilizes most VisNetwork features. In ARNet, an association rules is represented from two Nodes, which connect with and Edge. Depending on the arrow's direction, users can identify the antecedents and the consequent of a rule. Moreover, three interest measures of the association rules are visualized using the size and color parameters of the Nodes and length parameter of the Edges. In order to use those parameters, the values of each one interest measure was normalized.

Association rules visualization in ARNet is based on the antecedents and the consequent of each one rule. Rules are placed into clusters, which consists of a central *Node* and one or more outer *Nodes*. There are two available cluster presentation modes in ARNet, *Fan Out* and *Fan In*. *Fan Out* focuses on antecedent itemsets of each one rule. The central Node of each cluster is a unique itemset extracted from every rule, found in the data set, and outer Nodes are the itemsets found in the consequent part of every rule. *Fan In* focuses on the consequent of each one rule. The central *Node* of each one cluster is

19https://www.rdocumentation.org/packages/visNetwork/versions/0.1.0/topics/visLayout

a unique itemset extracted from every rule, found in the data set, and outer *Nodes* are the itemsets found in the antecedent part of every rule.

Furthermore, the physics of the network and each component of it is parameterized, in order to visualize more rules than a graph method can normally handle. Internally, specific parameters as *avoidOverlap*, is setted to value 0.3 in order to avoid *Node* collision when a cluster consists of more than 20 outer *Nodes*. Also *gravitationalConstant* and *centralGravity* parameters are specified with values -50 and 0 respectively, in order to allow the network to stabilize when the number of visualized rules are bigger than the user screen could handle. With *gravitationalConstant,* the length of each Edge will return to its original state, even if the user stretch it and with *centralGravity* the Nodes will stay at their original spot and not be pulled at the center of the network's interface. As a result, when users drag a cluster to a clear space, they can stabilize them and inspect them with ease.

User interactions with the network are handled using *visEvents* function. Two events are handled in order to provide users the option to interact with the network. The *select* and *click* events. When the user selects a *Node* by clicking on it, the *Node* will be highlighted. Different information appears to the users screen, depending on if the selected *Node* is central or outer. If the selected *Node* is central, users will view every available information about the rules that include the central *Node* at their LHS or RHS, depending on if the cluster presentation mode is *Fan Out* or *Fan In* respectively. If the selected *Node* is outer, users will view every available information about one rule that includes the selected *Node* at its LHS or RHS, depending on cluster presentation mode.

3.4 ARNet User Interface

In order to present to users every available information about the association rules and allow them to interact with the data, ARNet utilizes Shiny package. A user can select a primary rules measure and the number of rules with the highest score on that measure. The application will visualize the *n* number of rules and place them into a number of clusters depending on the selected cluster presentation mode. Moreover, the user can apply limits to every measure, primary or secondary, and rules with measures outside of the limits will be replaced by others, if they are available. After that, the user has the option to turn his focus on a particular itemset. Choosing an itemset as the *LHS* of a rule, ARNet visualizes all the filtered rules, in which the selected itemset is either their *LHS* or a part of it. Choosing an itemset as the *RHS* of the rule, there will not be any supersets of this itemset, since Apriori algorithm generates rules with a single item as the consequent of every rule.

Figure 3.4 1 demonstrates the ARNet's UI which visualize the mined association rules from Groceries transaction file. At the left side of the users screen, there is a group of radio-buttons, which allows users to select a primary rule measure. Also, there are three slide bars, which can be used to apply limits on the rules interest measures. One slide bar is available to specify the number of the desired number of rules that will be visualized and one more group of radio-buttons is available to specify the desired cluster presentation mode. Information about the visualized rules are available through an interactive data table under the generated network. At the right side of the UI, there are two drop down html widgets that users can set a desired itemset and view rules related to it. Finally, there is another slide bar which allows user to filter rules based on their length and a button to export the data table into a CSV file. The visualization of the rules can be exported too, as a PNG file.

Figure 3.4 2 shows the case where a user is interested in association rules with the highest *Support* values. A user can turn his interest on rules that include {yogurt}. The user can use the search area of the data table and it will show any visualized rules related with {yogurt}. By selecting the rows of the

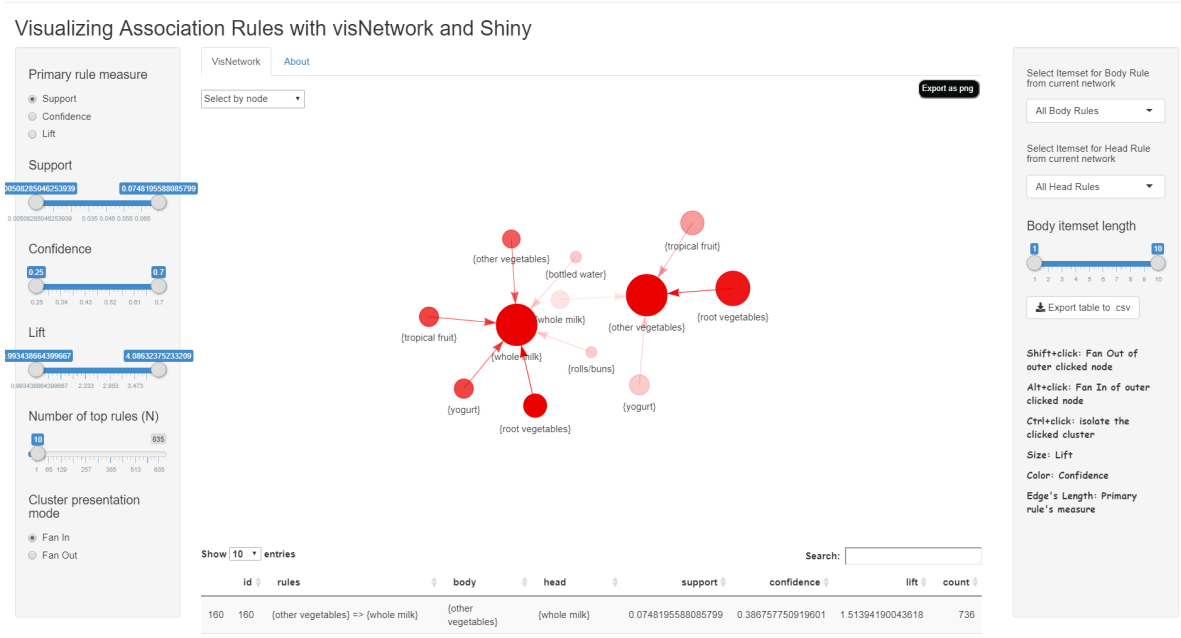


Figure 3.4 1: The ARNet User Interface.

table, *Nodes* with "yogurt" in their labels will be highlighted, since data table shows rules related with the word "yogurt". Note that the number of rules is set at 40, *Support* is the primary rule measure and no further filtering was applied. With different settings, *Nodes* related to "yogurt" may not be included in the visualized rules. At this point, the user may isolate these rules, by setting the {yogurt} itemset as the *LHS* of a rule, for example in Figure 3.4 3, or as the *RHS* of a rule. In the first case, ARNet will isolate the rules that contain {yogurt} in their *LHS*. Further isolation can be applied by the holding Ctrl key and clicking the central *Node* {yogurt}.

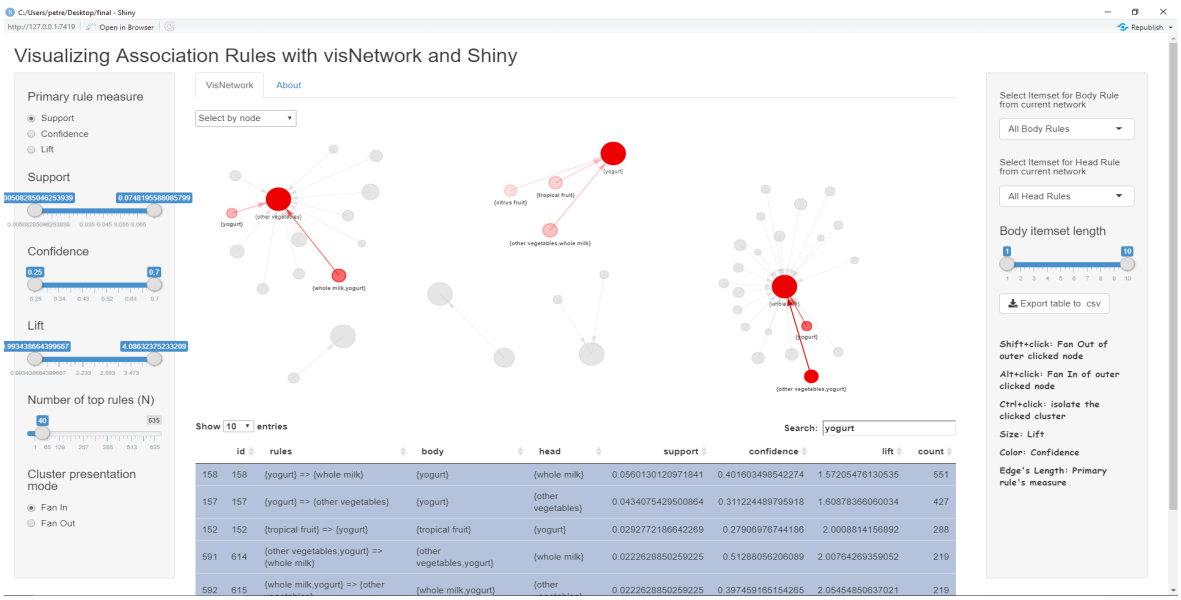


Figure 3.4 2: Searching and inspecting rules related to {yogurt} from the 40 rules with the highest *Support*.

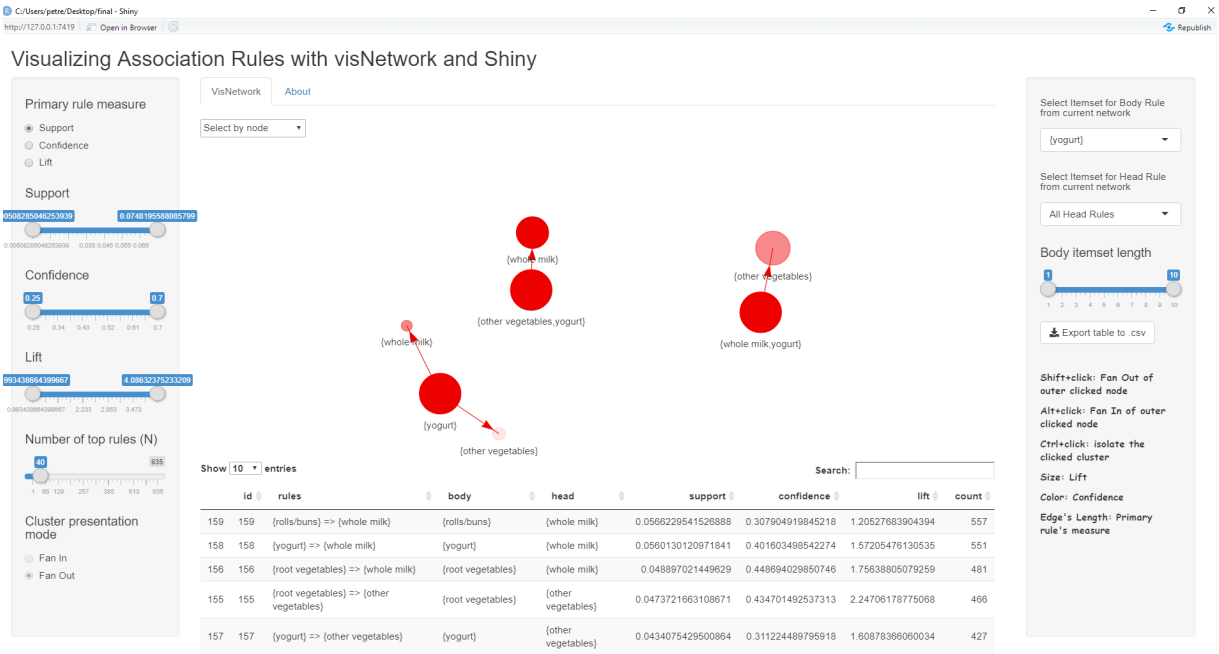


Figure 3.4 3: Setting {yogurt} as the *LHS* of the rules.

In Figure 3.4 4, the isolation of this cluster was an unnecessary step since there are few cluster in the graph, but it can be useful when there are many rules with a particular itemset as their body.

In Figure 3.4 5, {yogurt} is chosen as the *RHS* of the rules. Association rules with {yogurt} itemset as their *RHS* are shown. When a certain cluster is isolated a specific itemset is either the *LHS* or the *RHS* of the visualized association rules. If the itemset is the *LHS* of the rules, the *Edges* will point towards other itemsets and the presentation mode of the cluster is called *Fan Out*. If the itemset is the *RHS* of the rules, the *Edges* will point towards it and the presentation mode of the cluster is called *Fan in*. This

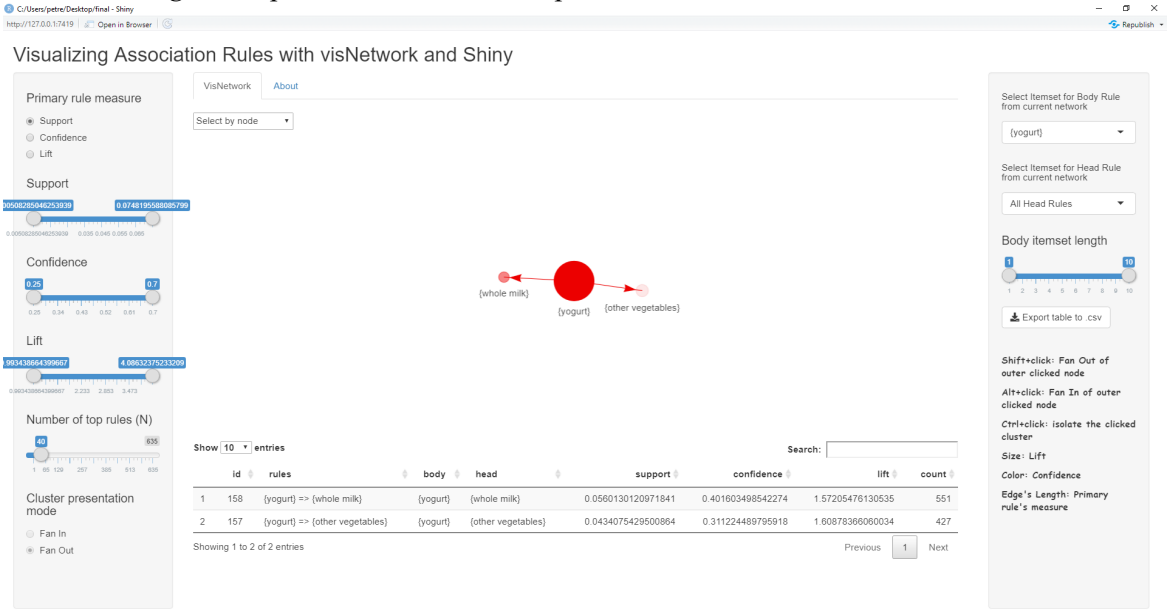
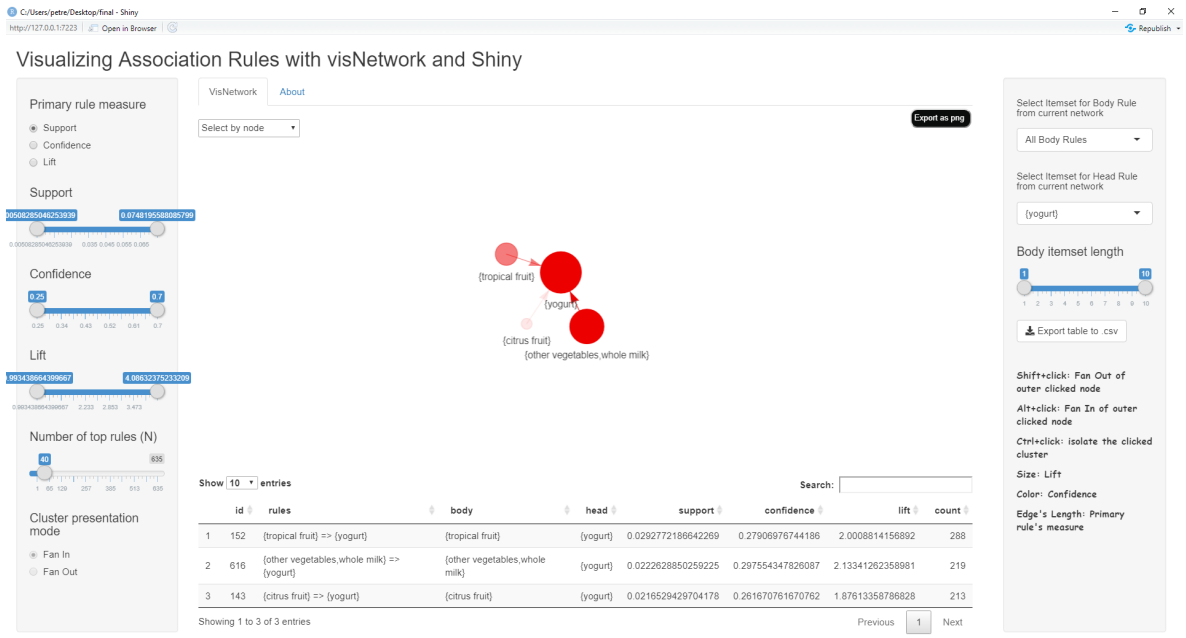


Figure 3.4 4: Using Ctrl+key shortcut to isolate a cluster.

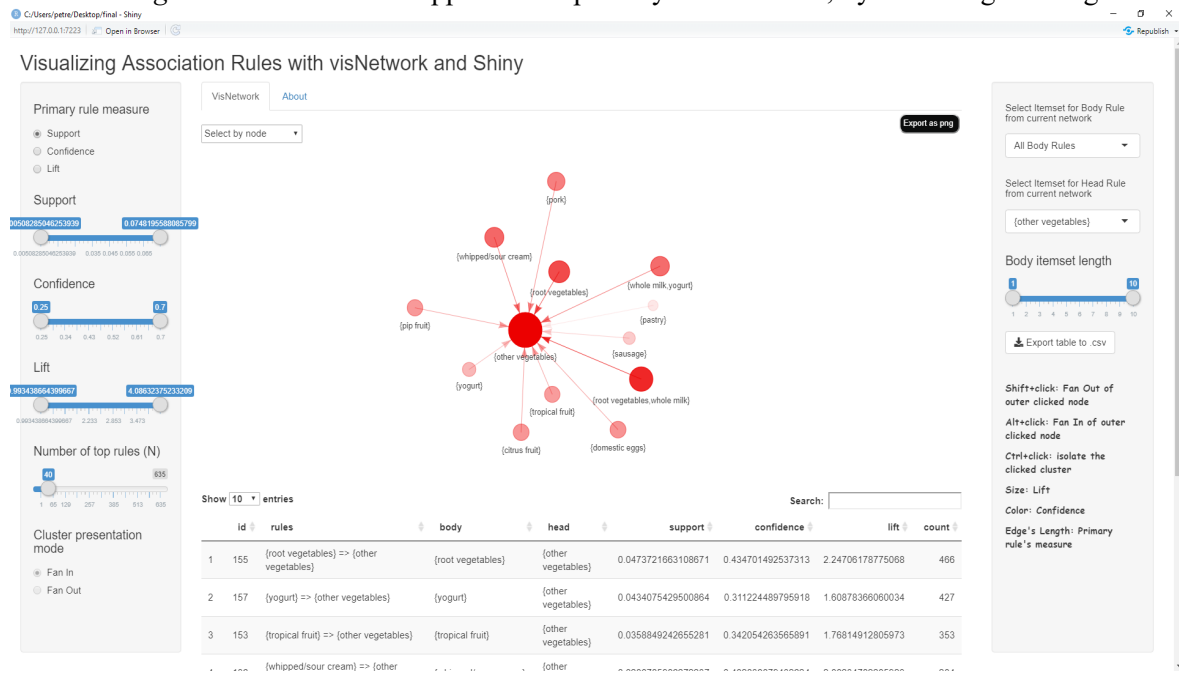


	id	rules	body	head	support	confidence	lift	count
1	152	{tropical fruit} => {yogurt}	{tropical fruit}	{yogurt}	0.0292772186642269	0.27906976744186	2.0008814156892	288
2	616	{other vegetables,whole milk} => {yogurt}	{other vegetables,whole milk}	{yogurt}	0.0222628850259225	0.297554347826087	2.13341262358981	219
3	143	{citrus fruit} => {yogurt}	{citrus fruit}	{yogurt}	0.0216529429704178	0.261670761670762	1.87613358786828	213

Figure 3.4 5: Setting {yogurt} as the *RHS* of the rules.

feature allows the user to get a view of the products that were purchased together, while they can be compared to each other by noticing the differences between the size and color of each on *Node* and the length of each one *Edge*.

Figure 3.4 6 shows the case where the {other vegetables} itemset is set as a requirement for the visualized rules. In this case, the number of rules was raised to 40, in order to contrast them using the *Nodes* and *Edges* attributes. Since Support is the primary rule measure, by observing the length of the



	id	rules	body	head	support	confidence	lift	count
1	155	{root vegetables} => {other vegetables}	{root vegetables}	{other vegetables}	0.0473721663108671	0.434701492537313	2.24706178775068	466
2	157	{yogurt} => {other vegetables}	{yogurt}	{other vegetables}	0.0434075429500864	0.311224489795918	1.60878366060034	427
3	153	{tropical fruit} => {other vegetables}	{tropical fruit}	{other vegetables}	0.0358849242655281	0.342054263565891	1.76814912805973	353

Figure 3.4 6: Setting {other vegetables} as the rules *RHS* with the highest *Support*.

Edges, users can identify the rule with highest *Support* value. Outer *Nodes* that are closer to the center *Node*, represent rules with higher *Support* values. Moreover, users can identify rules with the highest and lowest secondary interest measures by observing the size and color of the Nodes. Since, the human eye is not always accurate, users can interact with the data table.

In Figure 3.4 7, interacting with the data table, visualized rules are sorted by their *Lift* values. In this case, the rule with the highest *Confidence*, has the highest *Lift* too, and the rule with the lowest *Lift*, has the lowest *Confidence* too. Each secondary measure is visualized by a different attribute of the network's components, depending on the selection of the primary measure.

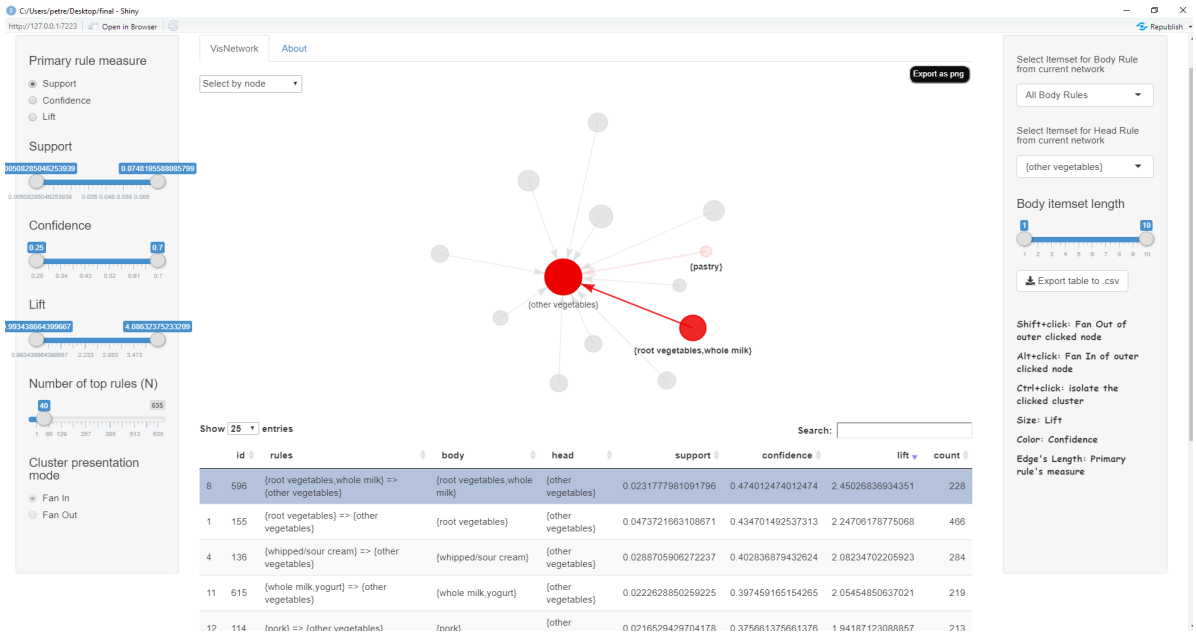


Figure 3.4 7: Highlighting rules with the highest and lowest score on *Confidence* and *Lift*.

If an unexpected error is produced, ARNet provides a quick feedback to users by utilizing *shinyJS*[20] package. Figure 3.4 8 demonstrates the handling of an error, where the user chose to view rules with their body including itemsets consisting of at least 2 items. There are no rules available with that meet the requirements, so ARNet notifies the user with a JavaScript alert.

20https://cran.r-project.org/web/packages/shinyjs/index.html

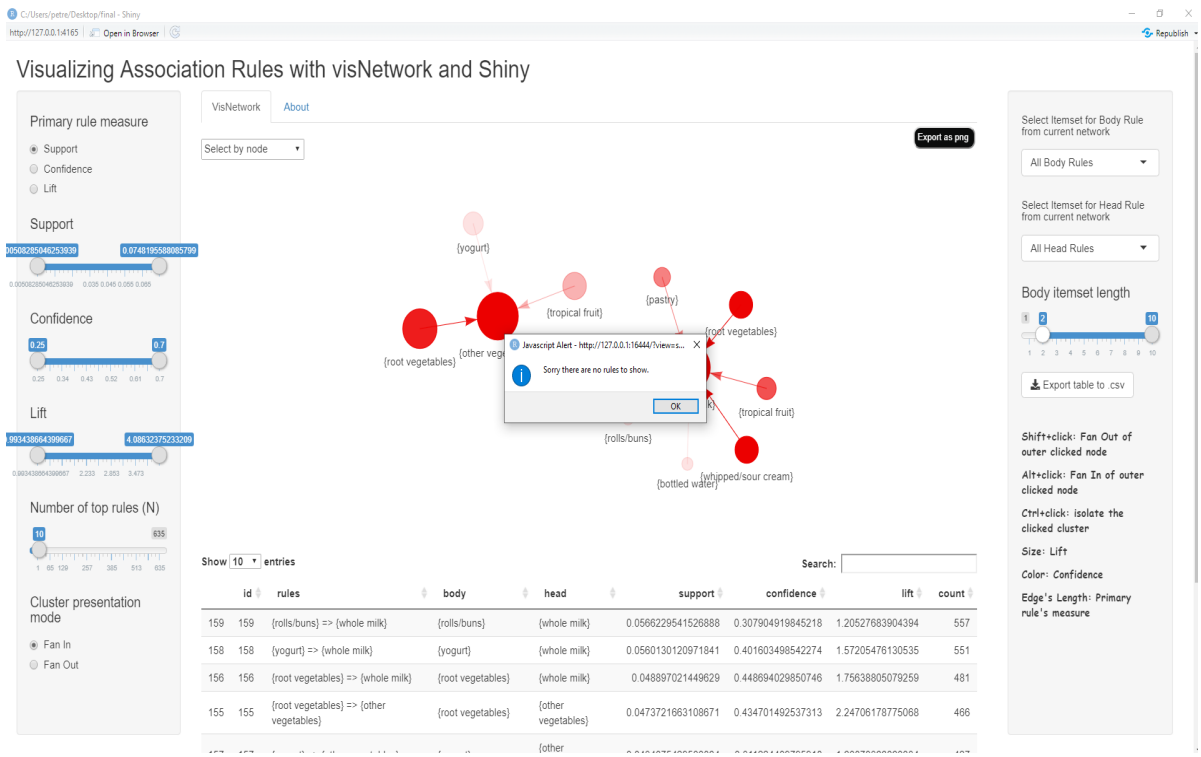


Figure 3.4 8: Providing feedback to the user with *ShinyJS*.

3.5 Summary

In this chapter, *VisNetwork* package was presented which offers a more interactive visualization method than the graph method from *arulesViz*. Also, it offers several options to visualize information about the interest measures of the rules, by specifying the attributes of the network components. Furthermore, *ARNet* was introduced and its visualization method was explained through a few use cases.

4 ARNet vs. ruleExplorer

4.1 Introduction

Since both *ARNet* and *ruleExplorer* are introduced in the previous chapters, it is time to compare them and discover which application is better for the visualization of association rules. In this chapter, each introduced visualization method included in *ARNet* and *ruleExplorer* will be contrasted with each other by visualizing a scaling number of rules and extracted the most possible information by interacting with the UI of each application. Also, the number of steps a user has to follow in order to get the desired results will be taken strongly into account. It is noted that the following criteria are applied. First of all, association rules are selected based on their Lift values and secondly, the minimum confidence of the rules is set to 0.3. Also, there are a few differences between the two applications.

This chapter is organized as follows. In section 4.2 the main differences between the User Interfaces of the *ARNet* and *ruleExplorer* are presented. In section 4.3, 30 association rules are visualized by *ruleExplorer* and *ARNet*, while in sections 4.4 and 4.5 the number of visualized association rules are 100 and 500 accordingly. In these sections, the results are discussed and analyzed. In section 4.6, a summary of *ARNet* and *ruleExplorer* is presented, with their advantages and disadvantages discussed.

4.2 Main Differences Between ARNet and ruleExplorer

The first most noticeable difference between *ARNet* and *ruleExplorer* is the approach on filtering the data. In *ARNet*, if a filter is applied by the user, the number of the visualized rules will not decreased, if it is possible. For example, if the user has set to view 100 rules and a filter is applied , the user will still view 100 rules that fulfill the requirements. On the other hand, *ruleExplorer* will decrease the number of rules, since it considers the visualized rules as the final data set. *ARNet* provides an interactive data table, which users can use as a tool to select and highlight rules on the network, while *ruleExplorer* provides a data table as a presentation method for the rules. Finally, users can explore rules in *ARNet* by clicking on the Nodes, because *ARNet* utilizes *JavaScript* click events.

4.3 Visualization of 30 Rules

Figure 4.3 1 demonstrates a scatter visualization of the rules. There are no hovered points and the measures of every rule can be estimated with good accuracy. On the other hand, no information about the antecedents or the consequent of each rule is available, unless the user decide to hover each point on the graph. This could cost a lot of time without any significant progress done on identifying the rules antecedents and consequent.

Figure 4.3 2 shows the *matrix* visualization version of the rules. Each column refers to a particular antecedent and each row to a particular consequent. With that noted, each part of the rules is visible to the users. This version lacks the visual presentation of the rule's measure, although rules may be ordered by a second measure. If a user wishes to learn more about a rule's measures from the graph, he may hover the respective rectangular of the rule in the graph.

In Figure 4.3 3, the *grouped* visualization of the data is displayed. Since this method is designed to handle a large number of rules and represent them as clusters, it is difficult to extract information about individual rules. Filling a table with rules that belong to a given cluster, could resolve this issue

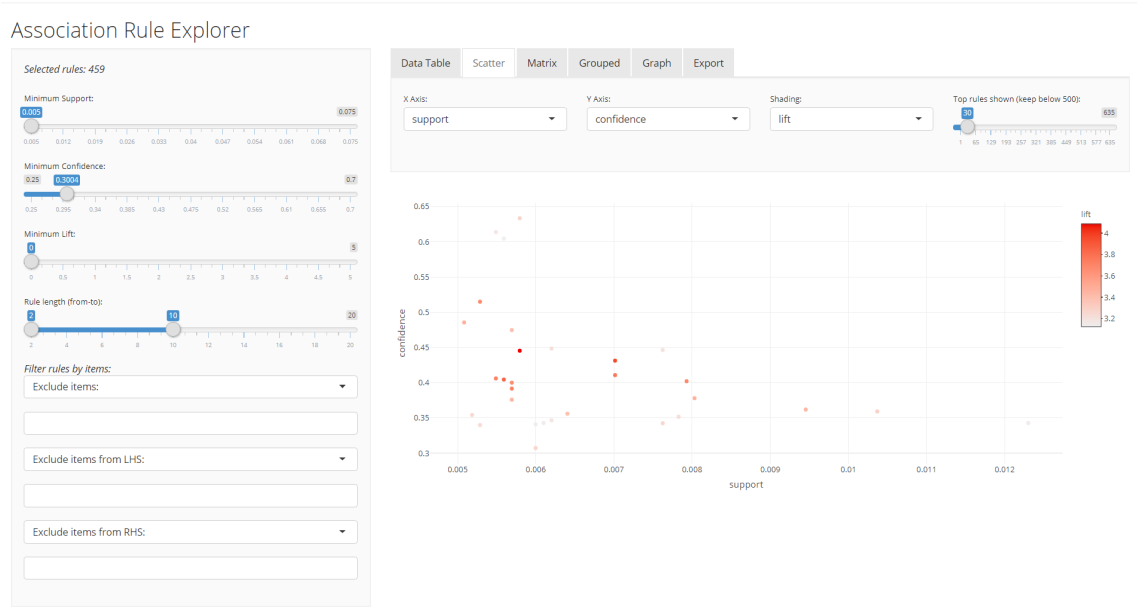


Figure 4.3 1: A scatter plot of 30 rules with highest *Lift* with ruleExplorer.

under the condition that the number of rules in a cluster is small. Otherwise, the user will spend a lot of time searching for specific rules in a large table. Using this method, only two measures are used for the visual separation between clusters, reducing the amount of information that one can extract.

Figure 4.3 4 and Figure 4.3 5 are similar with each other, since *graph* method from *ruleExplorer* and *ARNet* visualization method, are using geometric shapes, and *Edges* to represent rules. With 30 rules, both applications generate a clear representation of the rules, but there are some noticable drawbacks. For example, in the *rulesExplorer* network, there are too many *Edges* that connect each shape together, that makes certain relations quite fuzzy. This can be

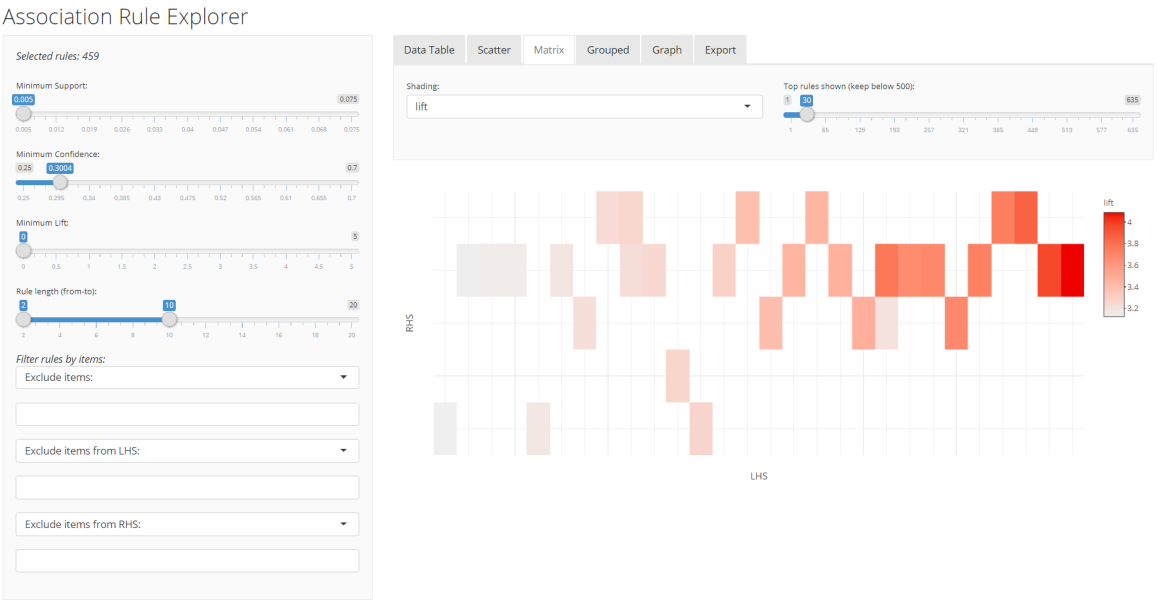


Figure 4.3 2:A *matrix* visualization of 30 rules with the highest Lift with ruleExplorer.

better noticed by looking in the center of the network, where {whole milk}, {other vegetables} and

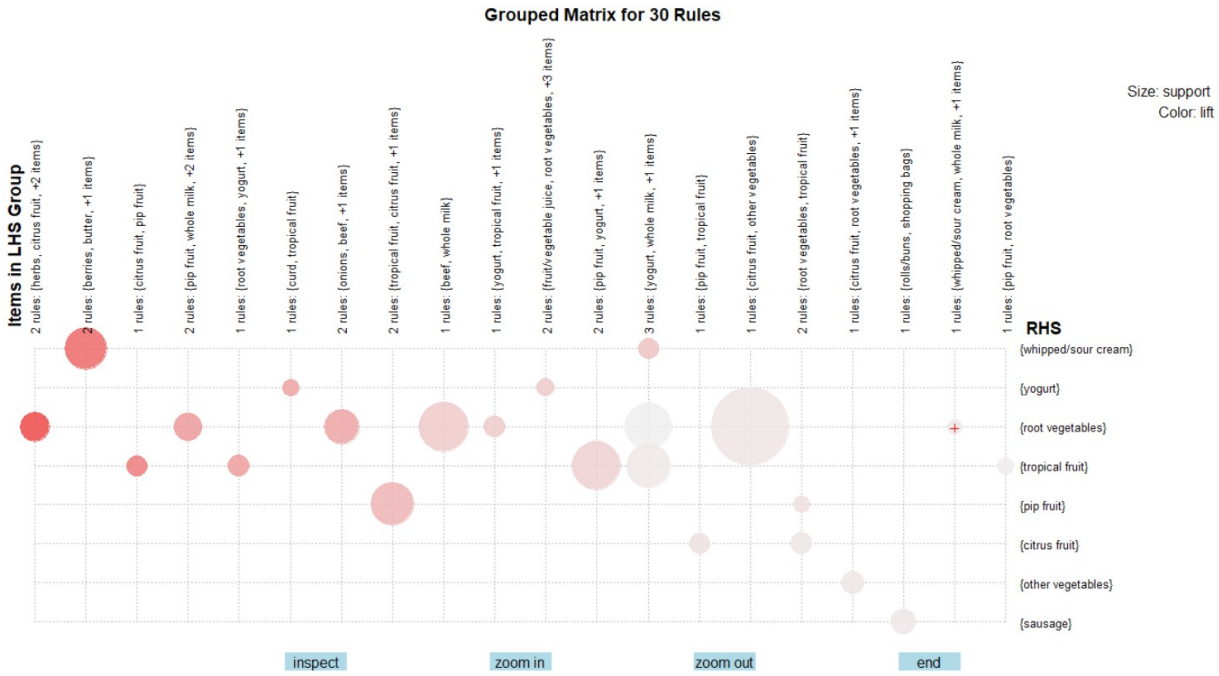


Figure 4.3 3: A *grouped* visualization of 30 rules with the highest *Lift* with ruleExplorer.

{root vegetables} stand. In *ARNet* network, the visual representation of each association handled differently. Instead of a rule connected with its items, itemsets are linked together, representing an association rule. This way, rules can be grouped into clusters based on their common antecedents or consequent and avoid crossed of hovered *Edges*, although it is noticeable, that the labels of each *Node* overlap each other when an itemset contains more than two frequent items. In both networks,

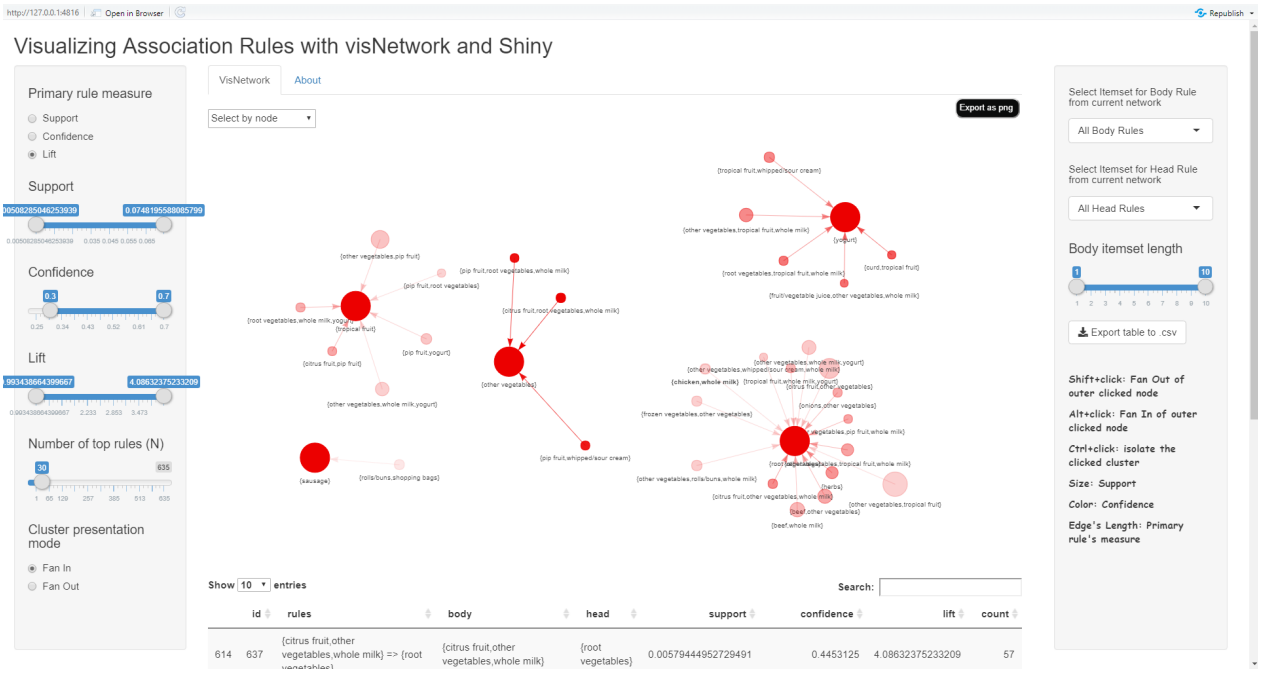


Figure 4.3 4: A visualization of 30 rules with the highest *Lift* with ARNet.

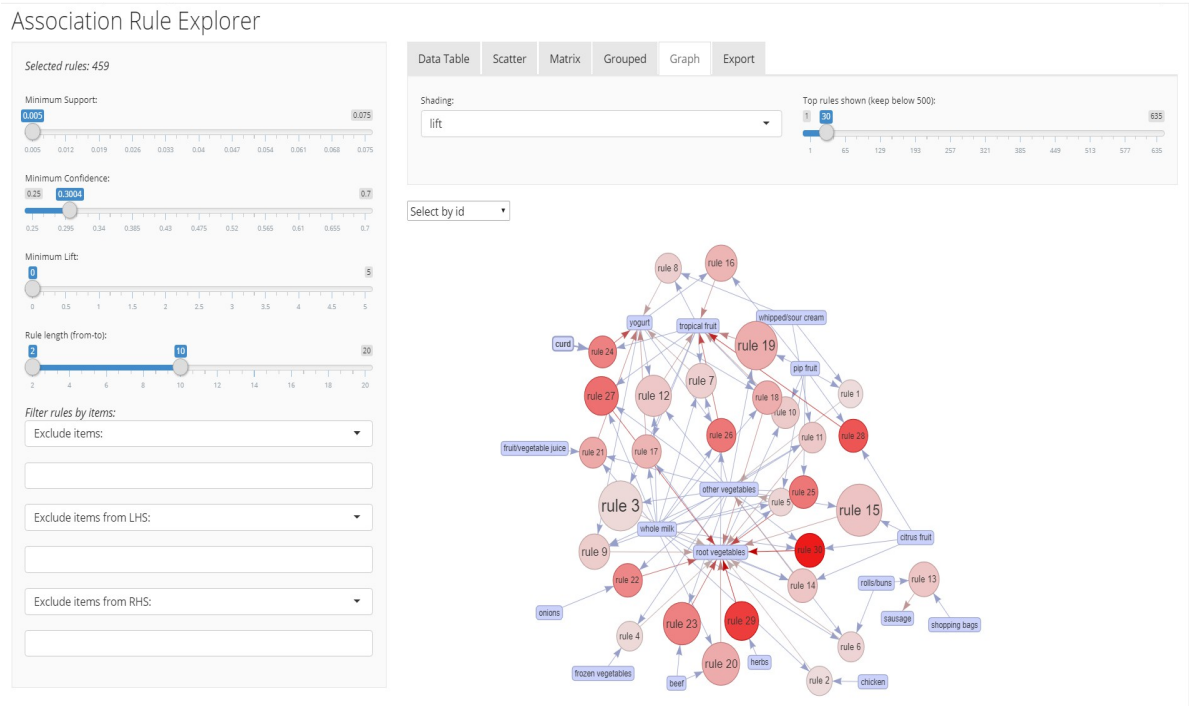


Figure 4.3 5: A *graph* visualization of 30 rules with the highest *Lift* with ruleExplorer.

information about the rule's measures cannot be accurate, but the viewer can identify the difference between them by contrasting the size and the color of each shape, and the distance between the *Nodes* in *ARNet* network. To get accurate information about a rule's measures the user should hover a *Node* in the first case, while in the second case he can click on an outer *Node* and the table will be updated containing all the information about it.

Figure 4.3 6 shows rules that have as consequent the itemset {root vegetables} in a *scatterplot* visualization method, after applying the proper filters. The only information that is still missing from the graph, is rules antecedents. This leads to the conclusion that this method cannot display all parts of a rule, making this method unsuitable to find relations between itemsets. The comparison of rules based on two or more specific itemsets is not a challenge for this visualization method.

Matrix method is not designed for isolating rules with one certain itemset, but the user can view rules that are related to a certain itemset by inspecting a row or a column of the graph. Also, this method takes only one measure into account. This means, that the displayed rules of the can be only compared visually based on one measure.

Grouped method although it is capable to visualize rules with one common component, their representation is not the most optimal. Figure 4.3 7 shows all the filtered rules with {root vegetables} itemset as their consequent. As mentioned earlier, this method groups rules into clusters based on their consequent. In this case each cluster contains one rule, which means that the size and the color of them is calculated from their support and their lift. If a cluster contains more than one rule, the size and the color of each cluster is calculated from the average value of the selected measures of the rules. This leads to the fact, that if a cluster contains more than one rule, visual comparison of rules is impossible.

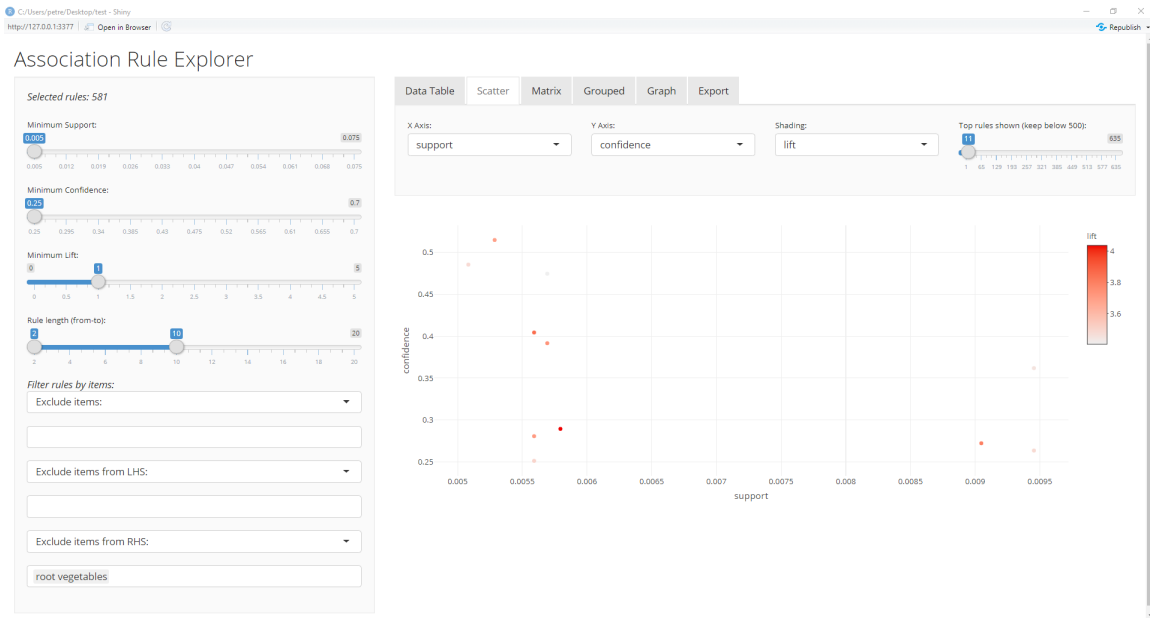


Figure 4.3 6: {root vegetables} as the RHS of the rules in a scatter plot with ruleExplorer.

In Figure 4.3 8, due to the low number of rules, the user easily identifies rules with the highest *Lift* values. Also, the size of each rule corresponds to its *Support* value. The available attributes of the *Edges* are not utilized which leads to the conclusion that a user can compare rules based on maximum two measures. The rules are distributed in the network based on their common components.

Figure 4.3 9 shows rules with {root vegetables} as their consequent. Every measure is used to calculate the distance between linked nodes, their size and their color, providing the user visual

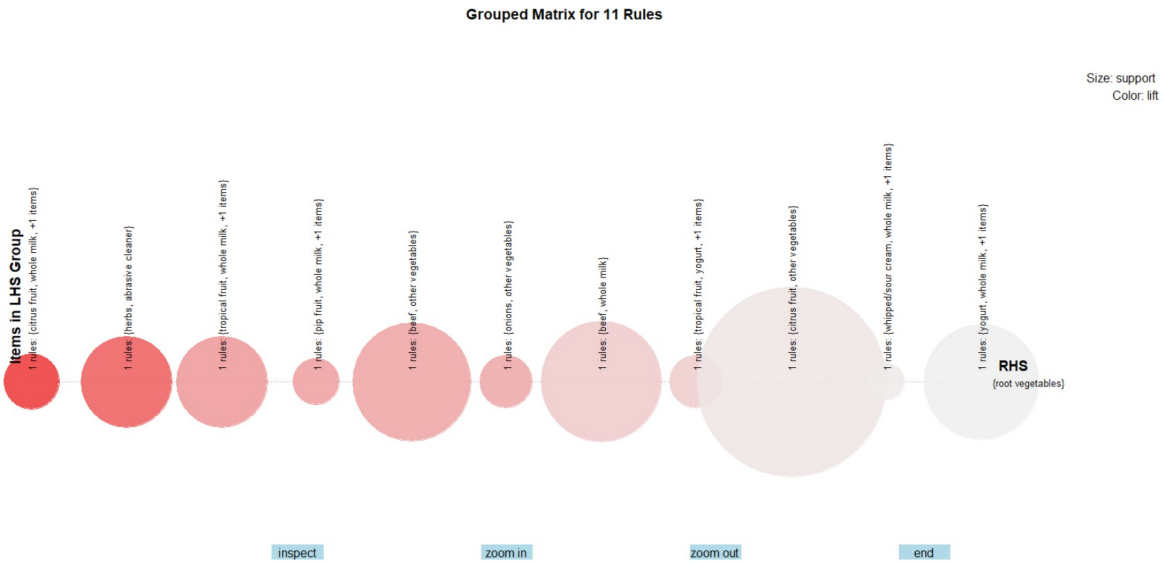


Figure 4.3 7:{root vegetables} as the *RHS* of the rules in a grouped visualization method in ruleExplorer.

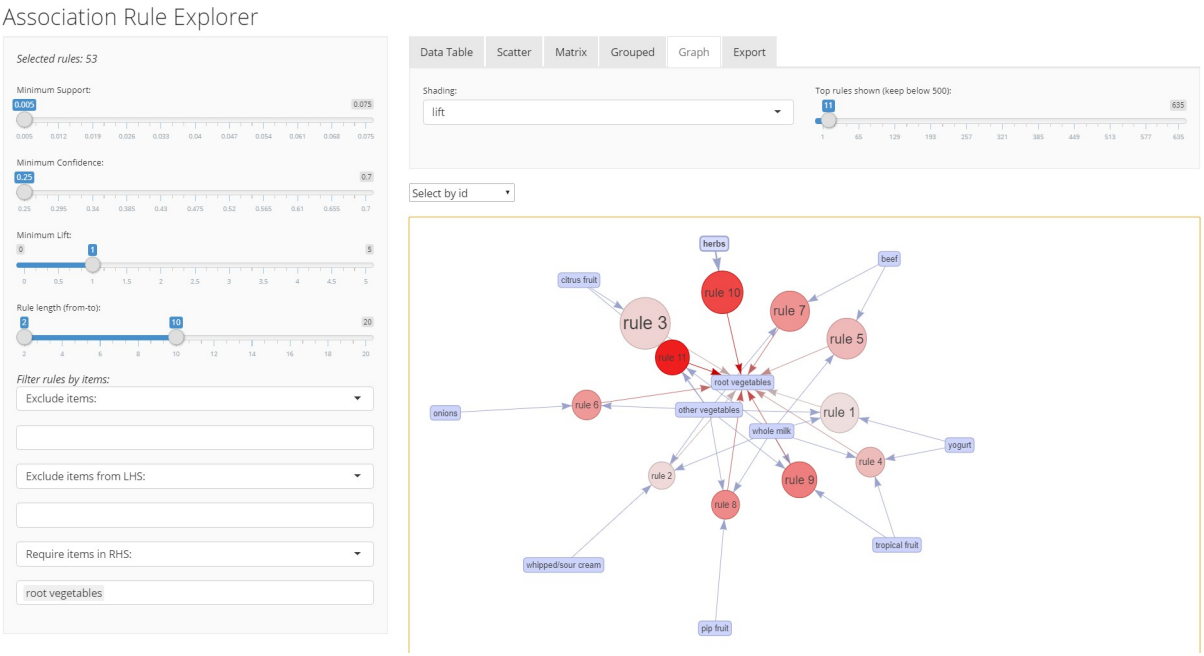


Figure 4.3 8: {root vegetables} as the *RHS* of the rules in a *graph* visualization with ruleExplorer.

information about each rule. The comparison between two or more rules is quick and effective by observing *Nodes* and *Edges* attributes.

So far, each application has been able to produce quality results. Without much effort, the user can identify every available information about the rules in both applications. Also, the user can contrast the visualized rules based on the visual information, and find rules with the highest interest measures very accurately.

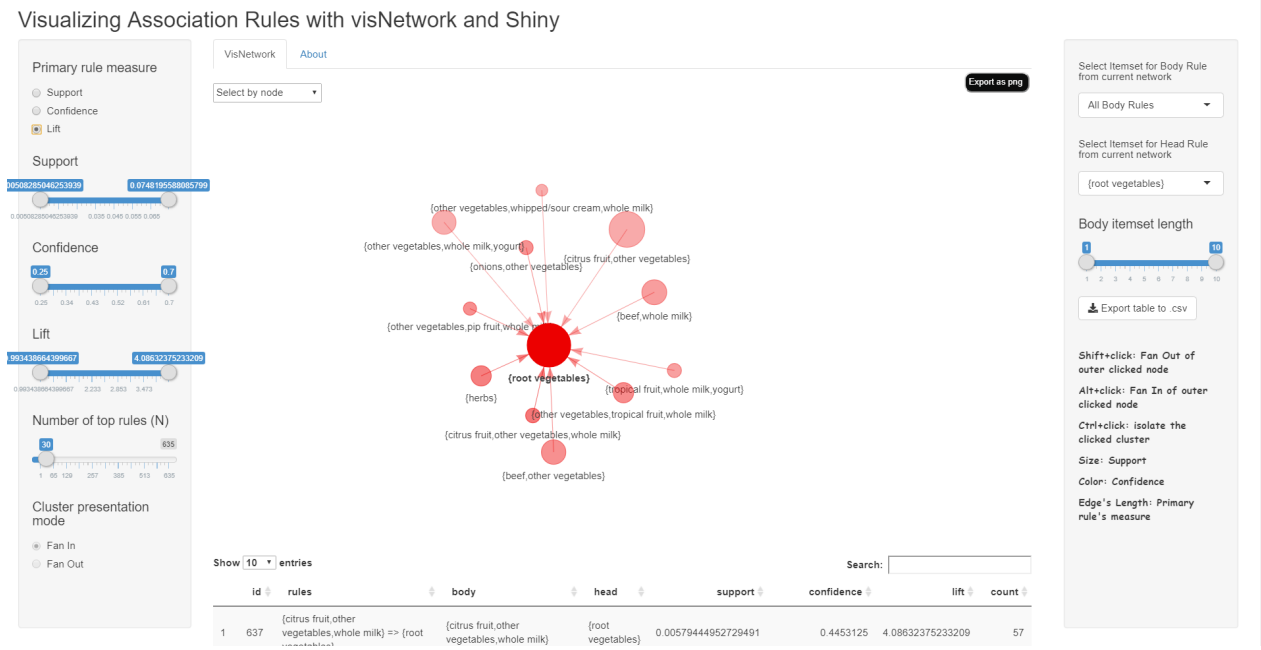


Figure 4.3 9:{root vegetables} as the *RHS* of the rules in ARNet.

4.4 Visual presentation of 100 rules

Setting the number of visualized rules up to 100, certain issues are expected in *graph* and *ARNet* visualization methods. As mentioned earlier, both methods are not designed to visualize more than 100 rules. So, it is expected to see crossed *Edges* ,and at some point *Nodes* overlapping others.

In Figure 4.4 1, there are a lot of crossed Edges. It is also obvious, that some rules do not fit in the users screen. This forces the user to zoom in and out in order to see the hidden rules. On the other hand, there is a dynamic distribution of specific items in the network. For example, {whole milk}, {root vegetables} and {other vegetables} are placed at the center of the network, because those items are included in the antecedent or consequent of the most rules. Raising the number of the rules to 100 has an impact on the amount of information that can be extracted, as the number of *Edges* grown larger. Even with highlighted items or rules, the interconnections between a rule and an item are not clear. The user has to observe rules from a highlighted item, and note if there is a pointing arrow attached to them or not, in order to understand if this certain item is an antecedent or a consequent. This issue is demonstrated in Figure 4.4 2.

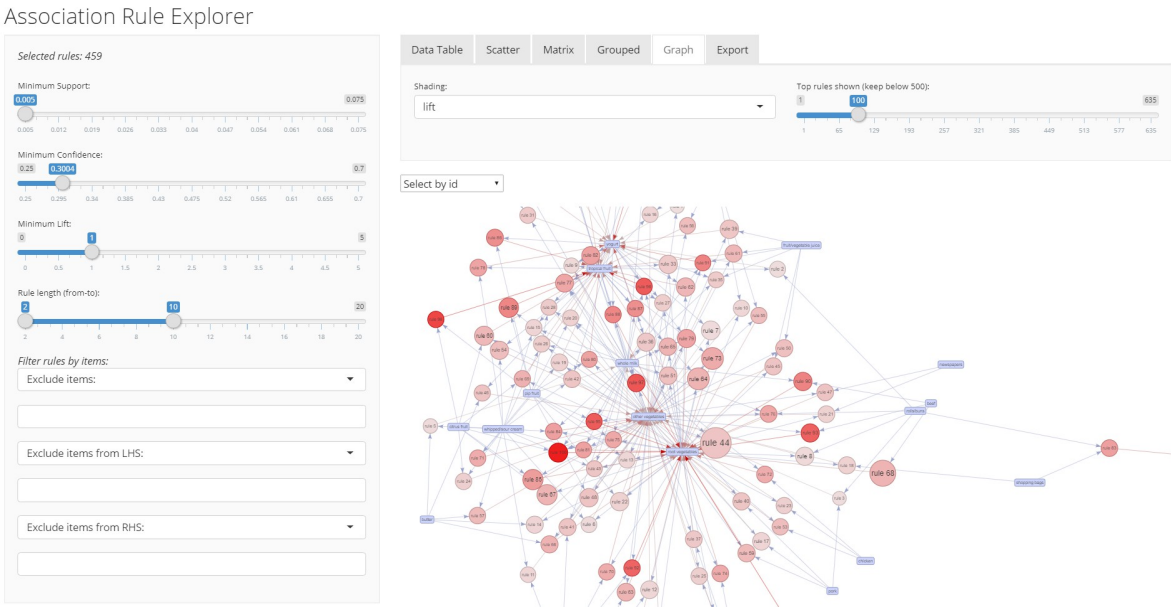


Figure 4.4 1: 100 rules with the highest Lift in a graph-based network in ruleExplorer.

A similar issue is observed with ARNet. The need for magnification and reduction occurs when the number of rules increases. As shown in Figure 4.4 3, the labels of the rules are blurred from a long distance. In order to make the labels appear, the user must zoom in on the graph. The different visualization technique of ARNet, places rules into clusters based on their components, which on one hand organizes the displayed information, but on the other makes the user to spend some time identifying which cluster is related on which itemset.

The rest of the methods can handle better a large number of rules. *Scatterplot* method, presents all the rules with a few points overlapped. The user can still identify rules with the highest interest measures, but there is no guidance on which rule is related to each item. The ability to inspect a certain rule, starts to become challenging but with small effort the user can view all the available information about a certain rule. New information can be extracted with a small raise of the rules. The escalation of the

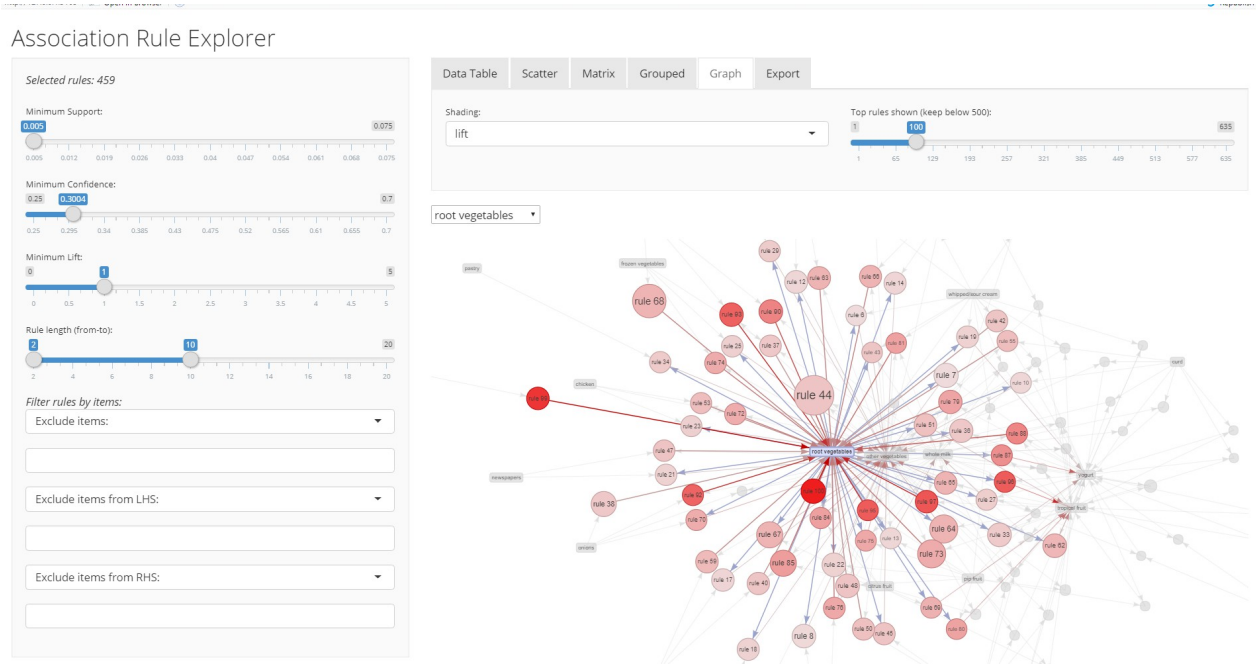


Figure 4.4 2: Hovering {root vegetables} in a *graph* visualization in ruleExplorer.

value of the X axis of the graph has changed, placing rules with the highest Lift values near the lower limits of support and confidence, as shown in Figure 4.4 4. This could an interesting observation for the user, which can be made with ARNet too. If the user observes the size of the nearest nodes in ARNet network, he will notice that their size is smaller than those that are further away. This leads to an hypothesis of a new theorem, that rules tend to be more interesting near the *Support* and *Confidence* lowest limits.

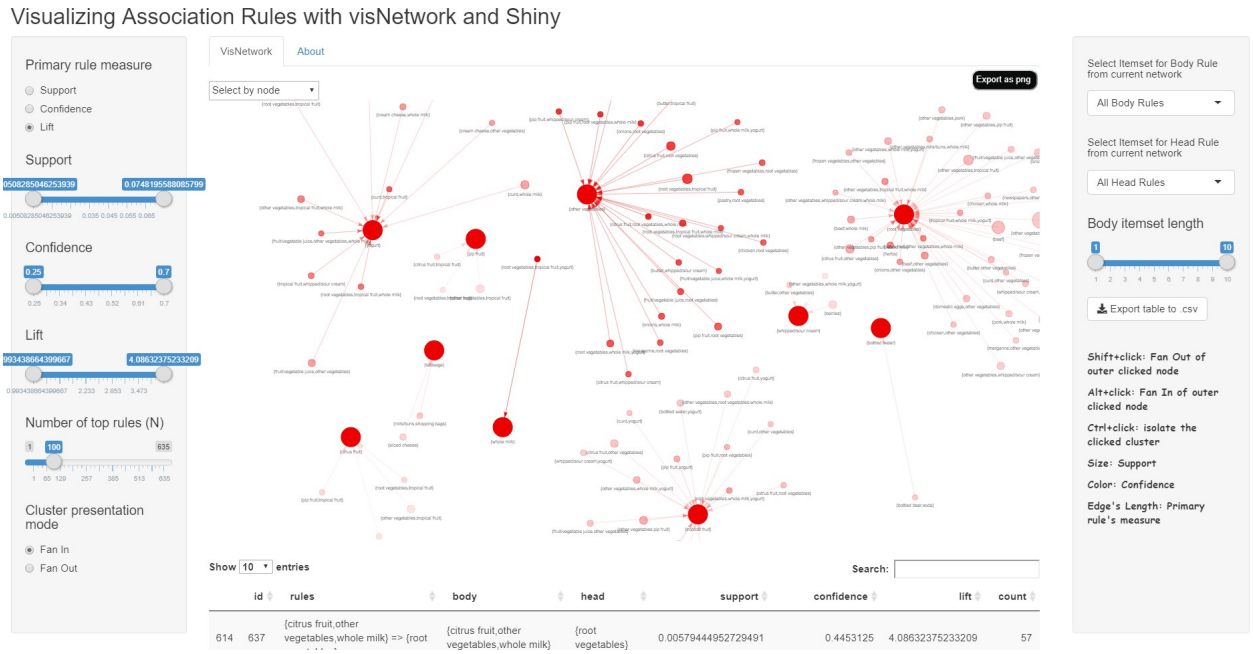


Figure 4.4 3: 100 visualized rules with the highest *Lift* in ARNet.

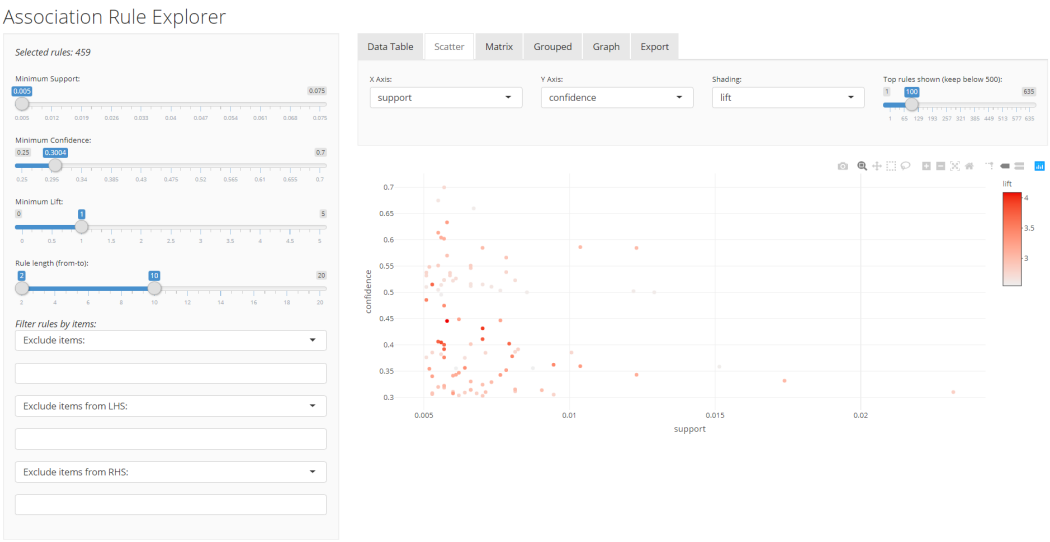


Figure 4.4 4: A Scatter plot of 100 association rules with the highest Lift with ruleExplorer.

The raise of the rules number does not affect the performance of *matrix* method. The number of lines will increase as more individual items appear on the right side of the rules, where 6 out of 100 rules were found. The number of the rules is still manageable for the *matrix* visualization method, which results to discernible sizes of shapes, so a user can inspect any rule without any advanced interaction with the graph, for example zooming in.

In *grouped* visualization method, raising the number of rules results to larger clusters. With larger clusters, the values of each interest measure are harder to track, as the size and the color of each cluster corresponds to the mean value of *Support* and *Lift* of the rules contained in a particular cluster.

After raising the number of visualized rules to 100, the disadvantages of each method begin to appear more clearly to the user. First of all, in *graph* and *ARNet* methods, the user has to spend some time in order to reorganize the components of the networks by zooming in and out or dragging each *Node* into a free space. In *matrix* method, each shape is shrinking, which makes each rule harder to detect. In *grouped* method, each cluster begins to contains more rules and the user cannot completely rely on its size or the color to find rules with the highest measures' values. In *scatterplot* method, points begin to overlap each other, making their inspection harder for the user.

4.5 Visual presentation of all rules

RuleExplorer graph and *ARNet* visualization methods start to under-perform with the increase of the number of rules. With every available rule visualized, it is unlikely to avoid overlapped *Nodes* and *Edges*. Moreover, with the raise of the rules, each generated network demands more time for the users to reorder each component of the generated networks. The maximum number of rules that meet the requirements is 459.

In *ruleExplorer graph* method, the result forces the user to spend time dragging each component or hover certain nodes in order to explore the interconnections between each rule and their items. Moreover, not every component of the network fits to the users screen, such as the labels the *Nodes*. On the other hand, the size and the color of each *Node* is discrete and the user can identify certain

areas of interest and focus them. In Figure 4.5 1, the necessity of zooming in and reorganizing every component of the network is obvious. Even if the user decides to spend time exploring the network, the results are not good enough. The user has to hover every rule, in order to view its measures values, while the large number of rules, creates a long distance between the rules and their items. A better demonstration of this issue is displayed in Figure 4.5 2. In this case the user was interested on which rules are related to {yogurt}. Hovering the item "yogurt", rules related with it will be highlighted.

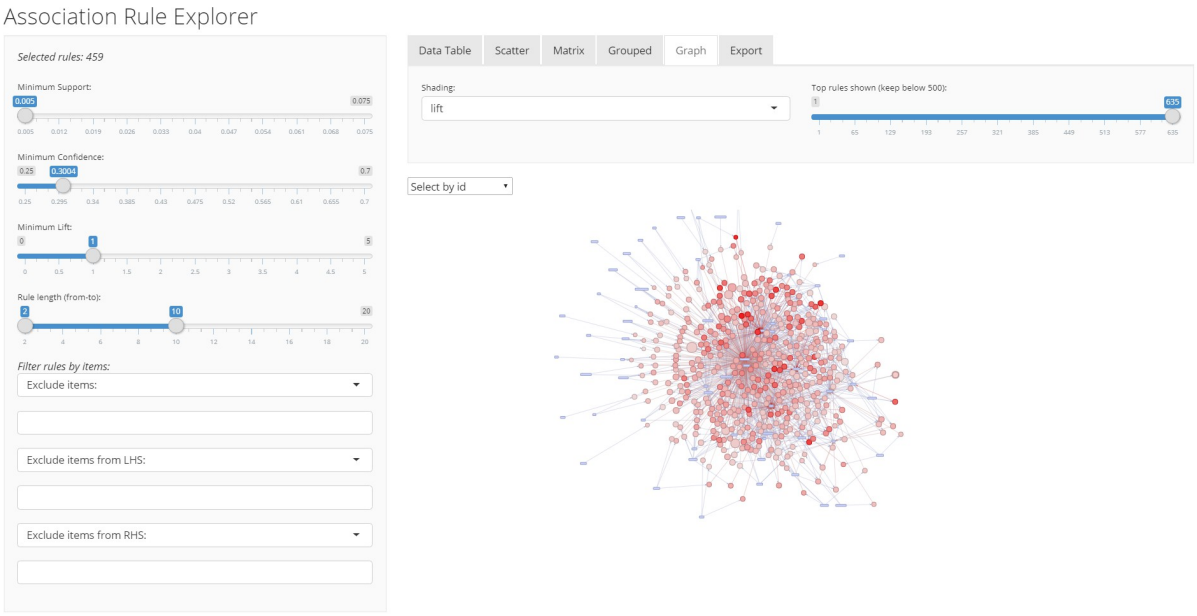


Figure 4.5 1: 459 visualized association rules in a *graph* visualization in ruleExplorer.

ARNet deals with this problem by creating clusters. Depending on the user's cluster presentation mode, the rules will be created on different clusters based on their itemsets, which offers a better view.

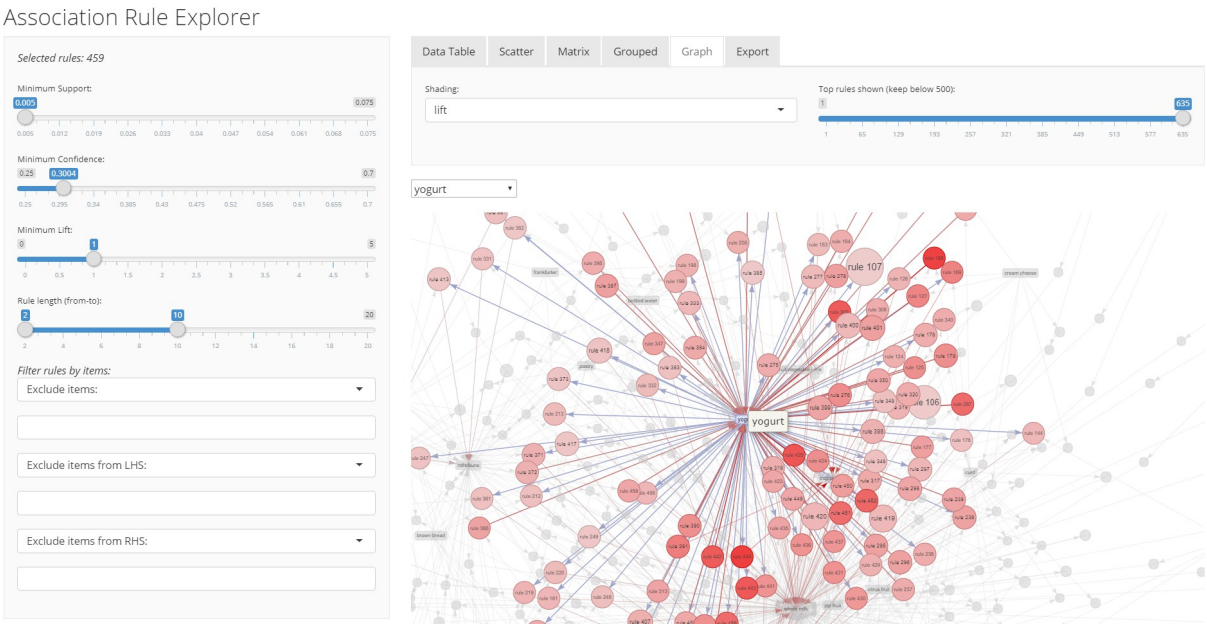


Figure 4.5 2: View rules related to {yogurt} in graph-based network in ruleExplorer.

On the other hand, the user will have to zoom in and explore every cluster on the network, since the distance between two connected nodes represents the primary rule measure and every cluster contains more rules with the growth of the rules number. The results are displayed in Figure 4.5 3. In both networks, the labels of each *Node* disappear after the user zooms out which makes the exploration of the network harder. The option of an isolated view of the rules, which are related to certain items or itemsets, is provided by the drop-down widgets or by the respective keyboard shortcuts. Nevertheless, using both applications the user can identify rules with the highest interest measure values, which is the main goal. After identifying them, the user can interact with the applications UI or the generated networks and isolate the desired rules.

In *matrix* and *grouped* visualization methods, the ability to identify the most interesting rules is still

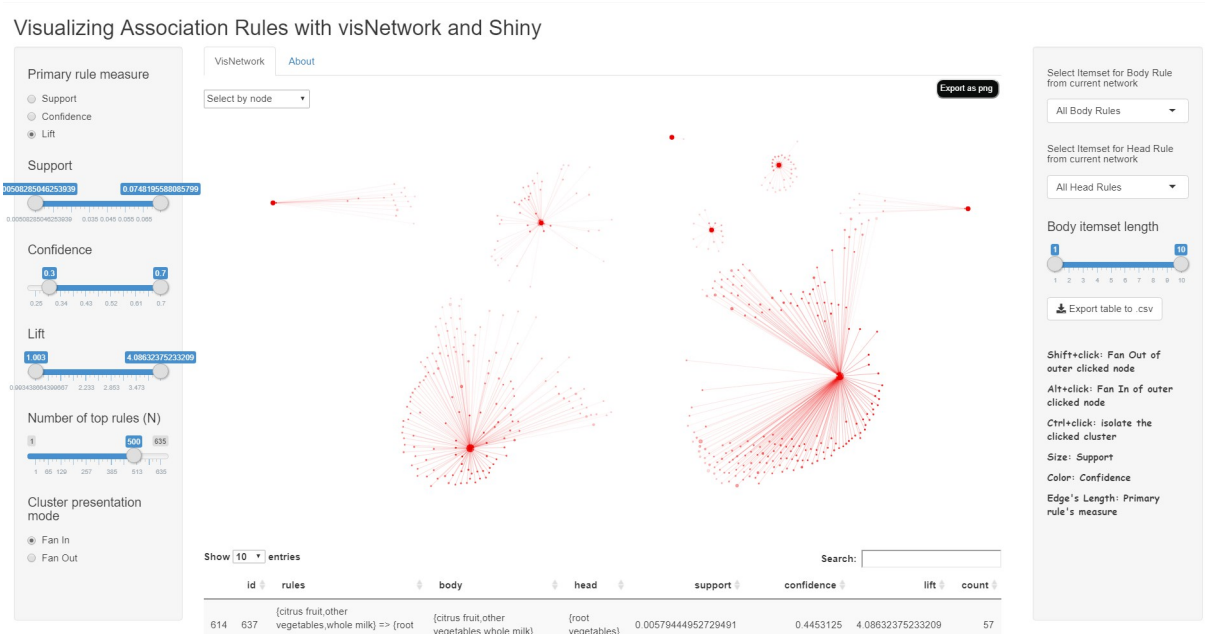


Figure 4.5 3: 459 visualized rules in ARNet.

available. The graph is still explicit with rules to be ordered based on their measures, as show in Figure 4.5 4 and Figure 4.5 5. As mentioned previously, the user will have to inspect the desired rule and search for additional information directly from the data set, which takes more time and needs programming skills. The same issue persists with *grouped* method, while the generated clusters have a bigger negative impact. It is mentioned, that visual information of a given node containing more than 10 rules can be misleading. In this case, a node can contain even 68 rules as shown in. The balance between the number of clusters and the number or rules they contain can be made by the user.

With *scatterplot*, the raise of the rules number did not make a lot of changes as shown in . Rules with highest interest measure values are still distinct. On the other hand, there are many data points that overlap others. This common issue forces the user once again to zoom in certain areas of the graph and proceed to further investigation of the rules. A new observation made with this method by raising the number of rules. It is clearer that rules with higher values on Lift are near the limits of Support values, which was only an assumption when the number of visualized rules was 100. This can be showing in 57, where the most data points are between 0 to 0.01 Support values and those with the highest Lift values are placed near 0.005 Support value.

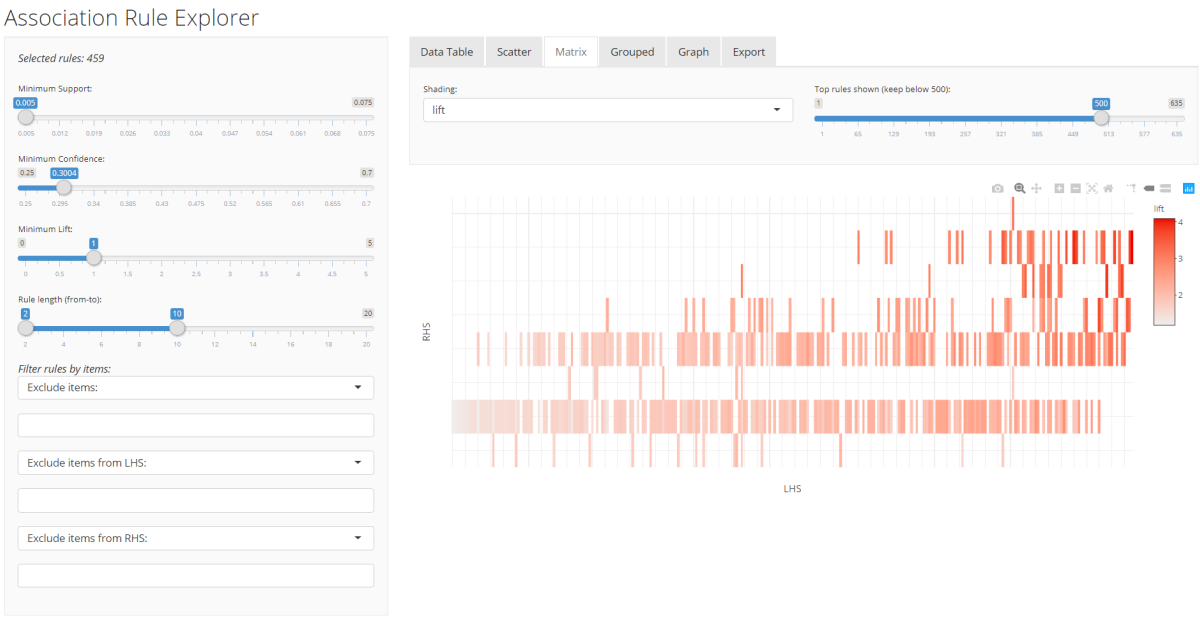


Figure 4.5 4: 459 visualized association rules with matrix-based method in ruleExplorer.

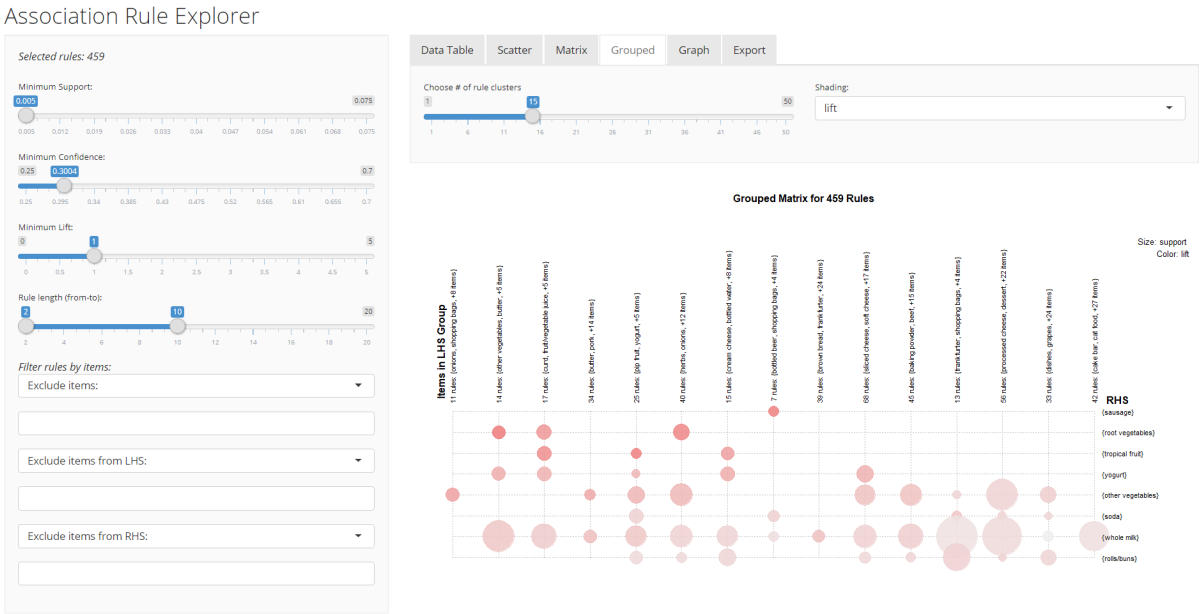


Figure 4.5 5: 459 visualized association rules with *grouped* method in ruleExplorer.

ARNet vs. ruleExplorer

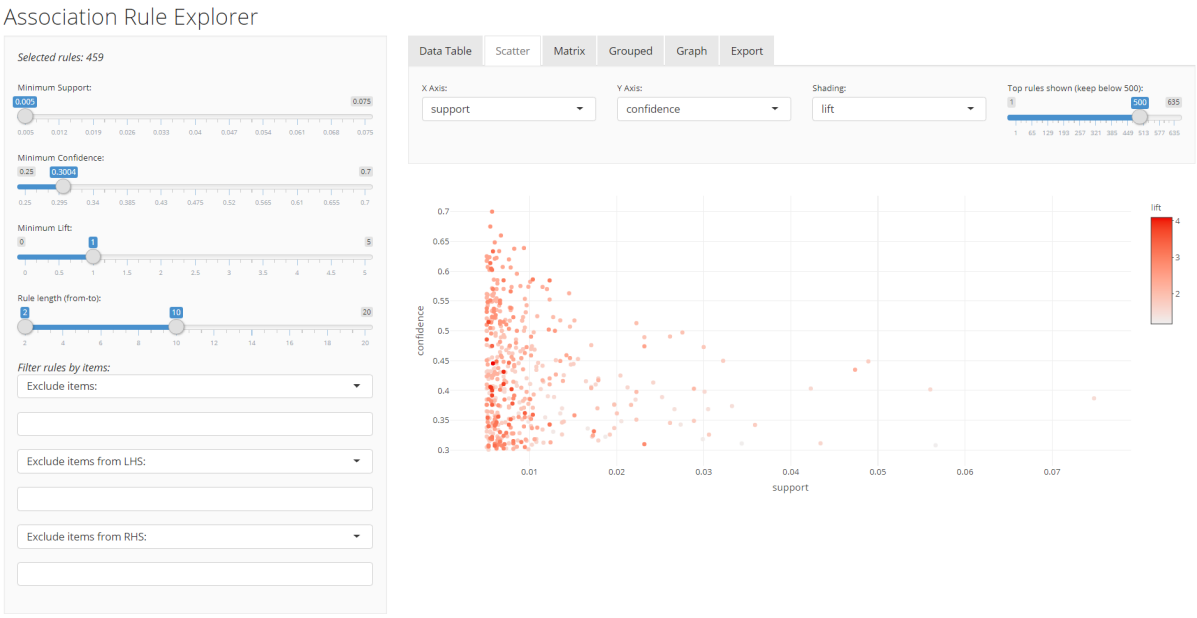


Figure 4.5 6: 459 visualized rules with *scatterplot* method in ruleExplorer.

4.6 Summary

In this chapter, *ARNet* and *ruleExplorer* were contrasted through a few case. *ARNet* is able to visualize association rules in a way that summarizes every advantage of *ruleExplorer* visualization methods. For example, *ARNet* allows users to make new observations about the data such as that interesting rules tend to have *Support* and *Confidence* values near the lowest limits. One can say that *ARNet* visualizes more information than a graph-based network can handle. Issues begin to appear with a large number of rules. For example, the labels of each component of the network may overlap each other, or appear blurred. One the other hand, these issues does not appear in *ruleExplorer* application, but in order to view every point of view of the visualized data, the user will have to spend time by observing every generated graph or network. Due to the features of the utilized package *visNetwork* in *ARNet*, a user would spend less time by interacting with the UI of the network in order to view all the available information about certain rules, than he would spend in *ruleExplorer*.

5 ARNet in Practice: (a) a Retail Store , (b) Academic Exam Grades

5.1 Introduction

As mentioned earlier, the *Support* value of a rule indicates the frequency of occurrence of the union of its antecedent (LHS) and consequence (RHS) itemsets in all transactions. The *Confidence* indicates the probability that a customer will purchase the consequent product on the condition of purchasing the rule's antecedent product(s) . *Lift* reflects the associative binding of the rule's LHS and RHS components (itemsets), excluding the statistical independence (noise).

To better realize the impact of the discovered association rules in the decision making process, consider a retailer who wishes to understand which products attract people to his store, or which products tend to be bought together.

In this chapter, two use cases are presented. In section 5.2, a retailer is using ARNet in order to promote the sales of his store. In section 5.3, a student wishes to find out the courses he is most likely to do well in, on the bases of his academic/exams record so far.

5.2 Retailer's experience with ARNet

So far, the meaning of every interest measure has been discussed and used to produce different graphs. Retailers can use any application visualizes data and make the right decisions to attract more customers and raise their sales. In this use case, the Groceries transaction file is used, which means that generated rules are about product categories. The following procedure does not respond to a retailer’s everyday problem, since there are many other factors that affect the behavior of the customers and are not included in the data. For example, the distance that customers have to travel in order to get to the retailer's store, or the origin of the products. Also, the brand of the products is one aspect that every customer pays attention.

Starting with the use of the *Support* measure, a retailer should look for rules with high *Support* scores. Setting the number of the visualized rules at 100, the retailer can view rules with the highest *Support* values and some rules with the lowest *Support* values. Moreover, due to the cluster presentation mode of ARNet, the retailer can view the top product categories that are included in most transactions. Rules related with these categories contain items that every person would buy daily. As shown in Figure 5.2 1, rule{other vegetables} => {whole milk} has a *Support* value of 0.074 and consists of products that may have with a shorter expiration date than others. For example, there are more rules related with {whole milk} than {soda} which means that most likely people purchase more often {whole milk} than {soda} as shown in Figure 5.2 2. This concludes that customers may prefer to visit retail stores that offer such products on discount and may do the rest of their shopping at these stores too.

Now that more customers might be attracted, the retailer would like to promote the sales of certain products that increase his revenue. Searching for rules with Lift values greater than 1, the retailer may find rules with long and short RHS itemsets. Rules with a long length will most likely have low Support values, which is natural since long itemsets can be found in few transactions. Rules with a long length can be used in order to devise a strategy for cleverly the products on shelves and in hallways. For example, in Figure 5.2 3 the {citrus fruit,root vegetables,whole milk} => {other

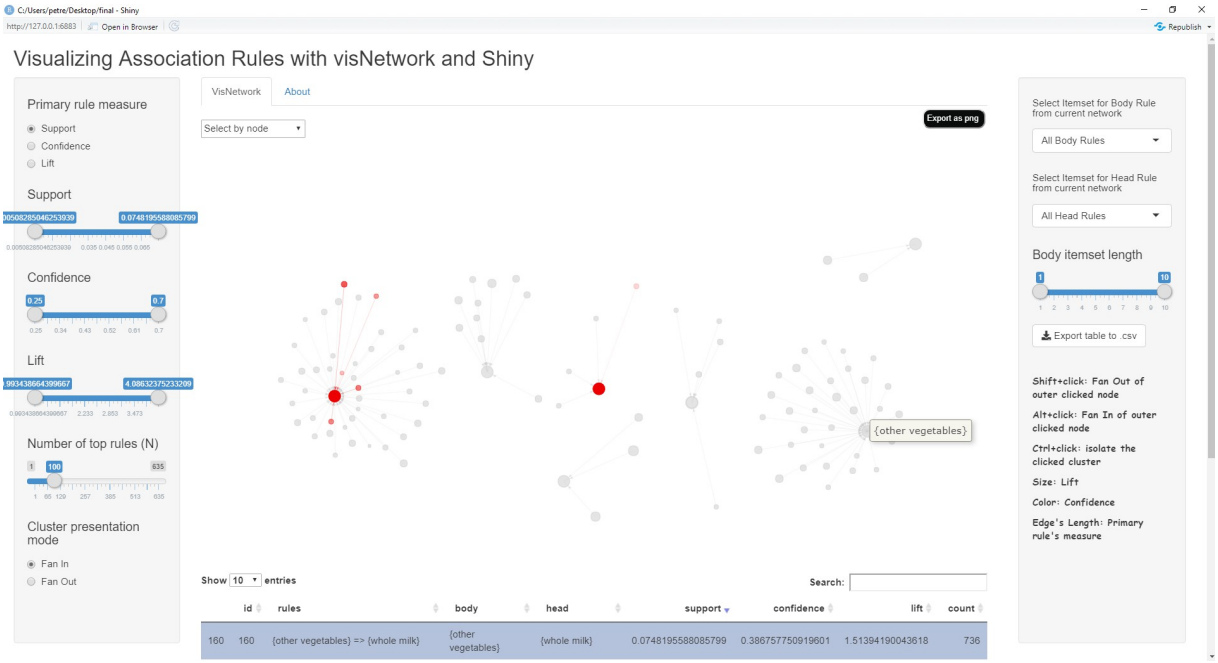


Figure 5.2 1: 100 visualized rules with highest Support values in ARNet.

vegetables} rule has a Lift value of 3.27 and a low Support value of 0.0057, as show in Figure 5.2 3. Notice, that {other vegetables} was one of top product categories in Figure 5.2 1. Moreover, there are rules consisting of the same itemsets but in different sides, which have different values on Confidence and Lift but the same Support value. An example is displayed in Figure 5.2 4, where the rule {fruit/vegetable juice,other vegetables,whole milk} => {yogurt} and the rule {fruit/vegetable juice,whole milk,yogurt} => {other vegetables} consists of the same items, share the same *Support*

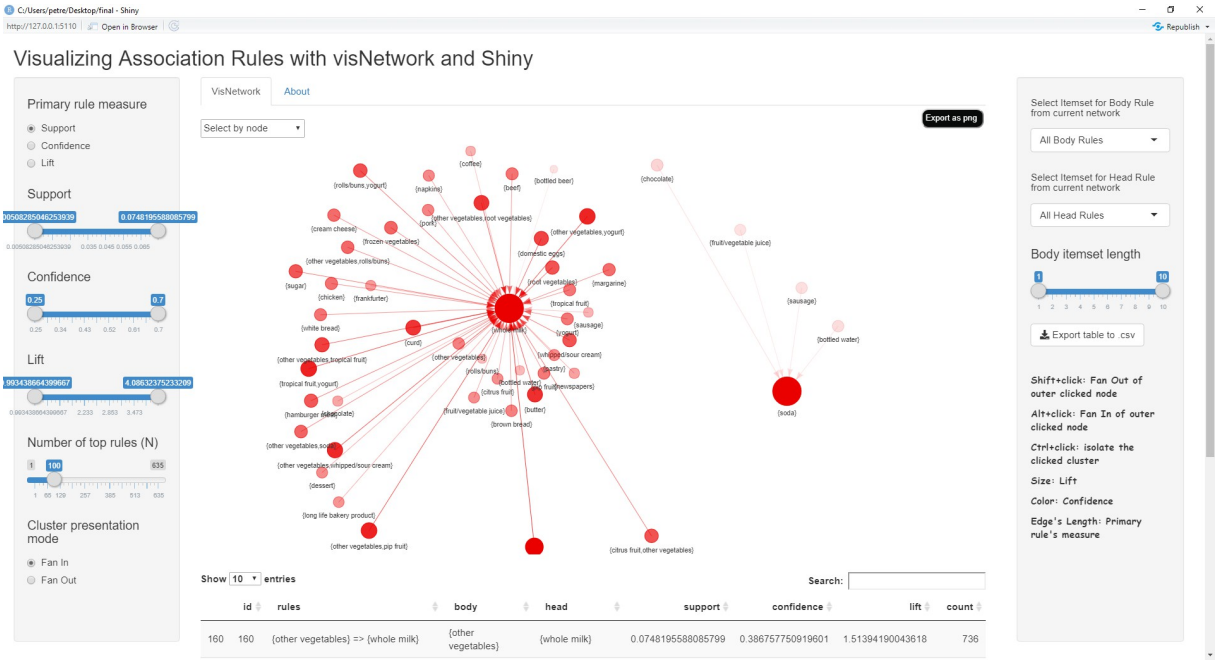


Figure 5.2 2: Rules with {whole milk} and {soda} as consequent among 100 rules with the highest Support values in ARNet.

ARNet in Practice: (a) a Retail Store , (b) Academic Exam Grades

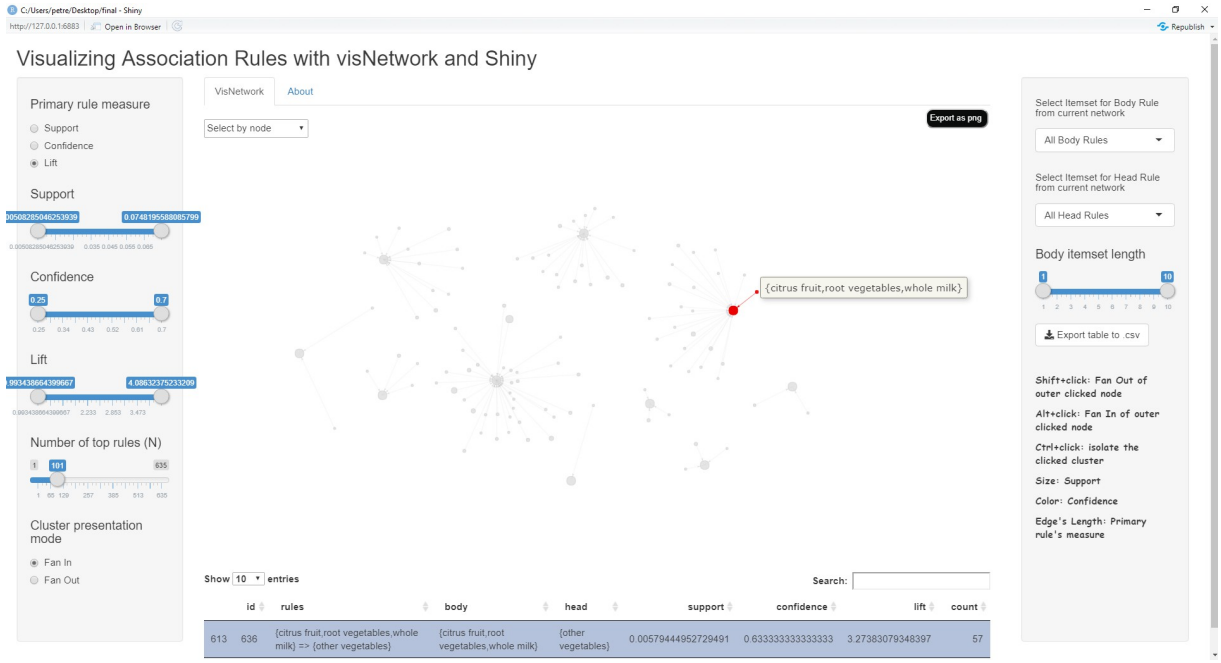


Figure 5.2 3: 100 visualized rules with highest Lift values in ARNet.

value but they differ at *Confidence* and *Lift*. A retailer might be interested to discover why these rules have different values on Lift and Confidence.

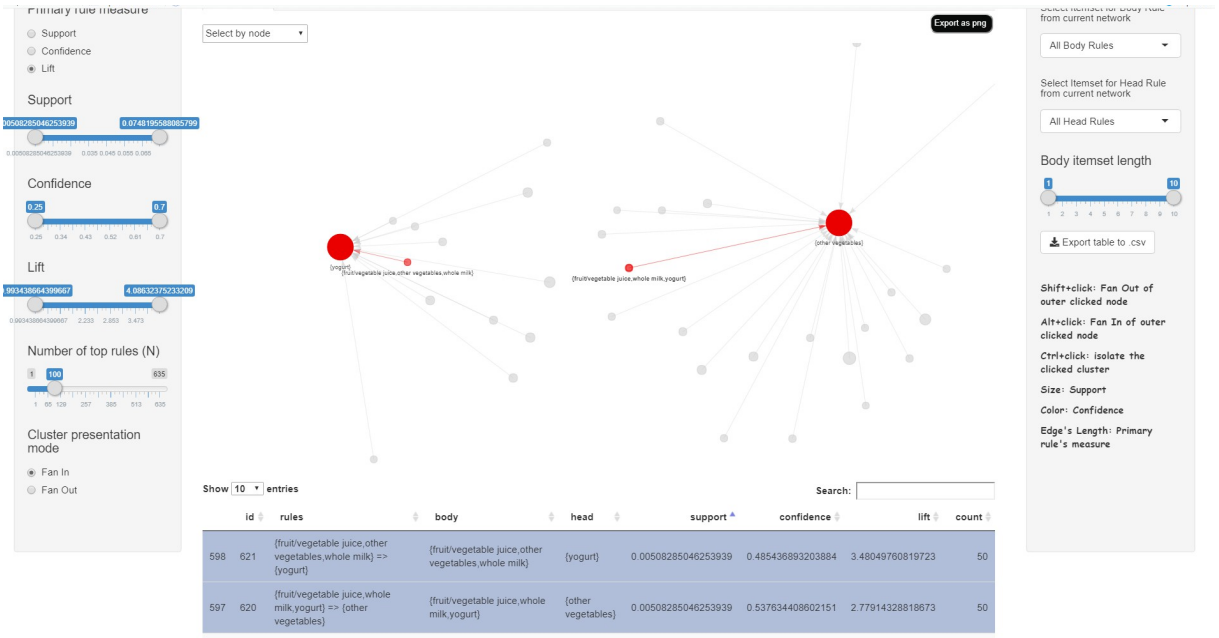


Figure 5.2 4: Two similar rules among 100 rules with the highest Lift values in ARNet.

5.3 Academic Exam Grades

In this case, the data to be visualized are about the performance of the postgraduate students of Department of Information and Electronic Engineering. The data were available from Cardisoft Teaching Assistance and Secretarial Support system. The original data were processed and reformed during the postgraduate thesis of Konstantinos Kelesidis with the supervision of Professor Dimitris A. Dervos [7]. Also, association rules were mined from the final data form using the *Apriori* algorithm. The mined association rules involve (course-code, student performance) pairs. An itemset, for example {01-A-Metria} consists of three fields. The first one is the semester of the course, the second is the encoded name of the course and the third one is the grade students scored. The grade that each student achieved for each exam is codenamed/classified into four (4) categories : *Metria*, *Kala*, *Polu Kala*, *Arista* (i.e. *Average*, *Above Average*, *Very Well* and *Excellent* respectively). A rule consists of just two itemsets, its LHS (body, antecedent) part, and the RHS (head, consequent) ones. IT is interpreted as follows: the LHS (course,grade) part tends to imply the RHS (course, grade) part. Equivalentry, students who pass the course of the LHS with the grade in the LHS, tend to pass the course in the RHS with the grade in the RHS. By creating encoded labels for each itemset, there is more room for additional information to be included in the network. For example, by including the grade of each one course in the label of each one Node, the thickness of each Edge remains free for future use. Also, with this approach clusters contain a small number of rules which makes them distinguishable. The mined association rules network as displayed by ARNet is shown in Figure 5.3 1.

One more association rule interest measure used is the *Conviction*. The *Conviction* measure quantifies the intensity with which the antecedent part of the rule implies the consequent part of the rule (causality) . It ranges from 0.5 to infinity [4]. If the value is equal to 1 then the antecedent is statistically independent from its consequent, while values greater than 1 indicates a stronger relation between the two parts.

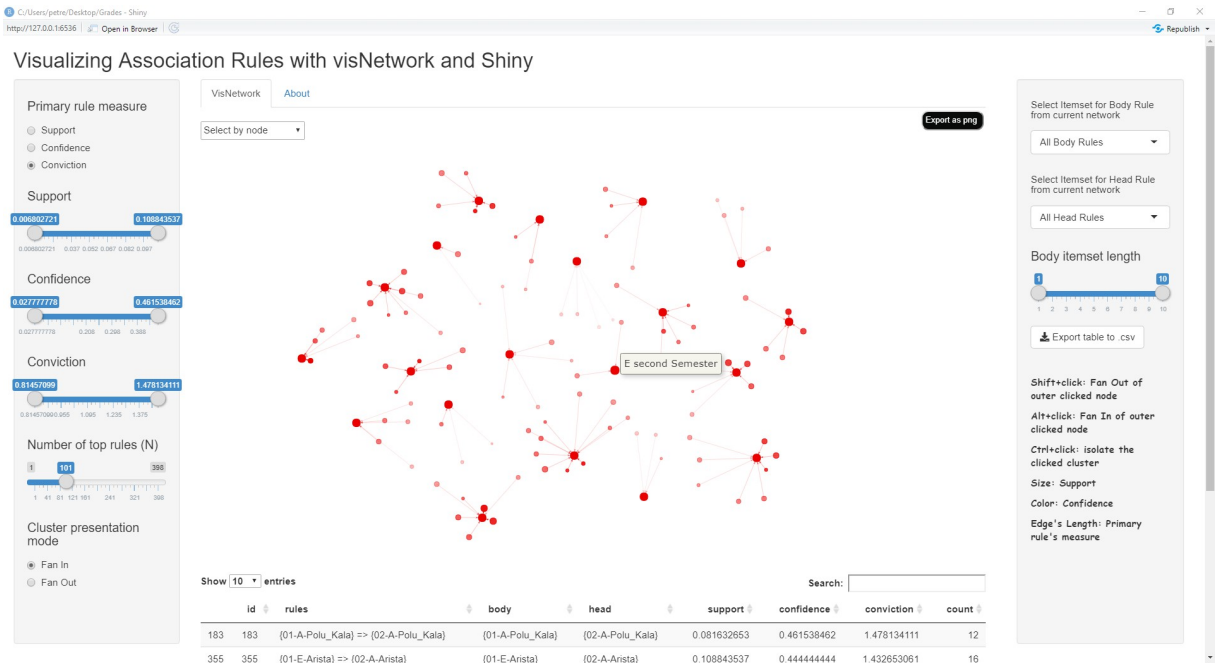


Figure 5.3 1: 100 rules of Academic Exams Grades with the highest conviction in ARNet.

Using ARNet to visualize the mined rules, users can set the cluster presentation mode to "*Fan Out*" using the "*Cluster presentation mode*" radio-buttons and identify courses of the first semester that when passed with good grade tent to imply success in course(s) of the second semester. For example in Figure 5.3 2, students who scored *Arista(Excellent)* at the exams of the second class of the first semester, excelled(*Arista*) in the exams of the first and second class of the second semester with the same score. By analyzing and interpreting the network of association rules, more information can be extracted. For example, the degree of difficulty for a certain course. To evaluate such a hypothesis, one needs to include more information about each course.

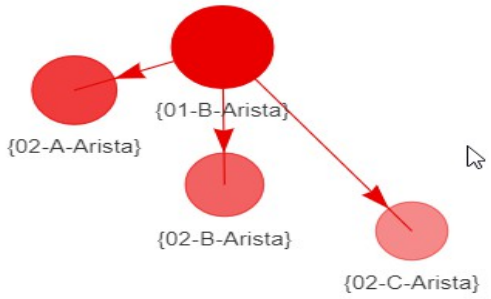


Figure 5.3 2: A Fan Out cluster presentation mode of {01-B-Arista} itemset in ARNet.

Despite the missing visualized data, one may notice that the courses difficulty varies . For example, in Figure 5.3 3, students tend to achieve an "Average" score at the first course of second semester even when they have passed the third course of the first semester with "Very Well". One can state that each course specializes on different technologies that students are not familiar yet and justify the performance of the students. Others may state that not every student spends as much time as it is needed to excel in the exams of some courses. This concludes to the fact that the results of the visualization of the data are not absolute. On the other hand, a visualization of the association rules network with ARNet initiates very useful discussions/debates on how the projected information is realized/interpreted. This fact is inline with the goal initially identified to be worth pursuing in this thesis project.

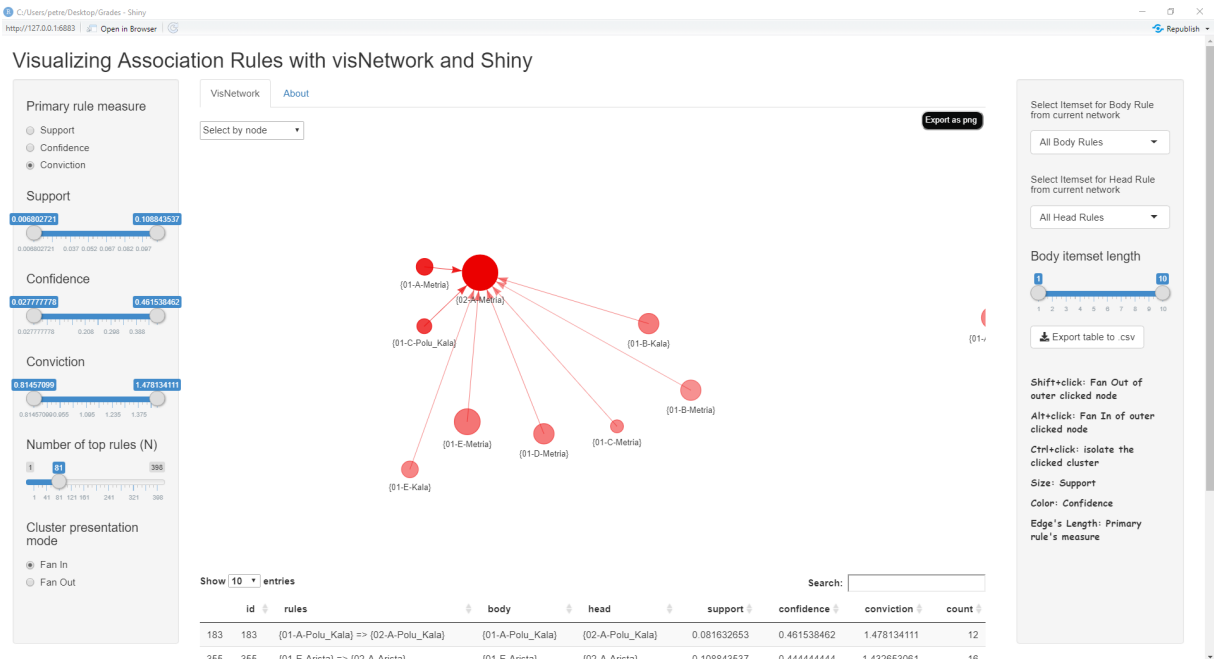


Figure 5.3 3: Rules that have as consequent the {02-A-Metria} among 81 rules in ARNet.

5.4 Summary

In this chapter, two different use cases with different data and different end users were presented. At first, a retailer uses ARNet to find the proper product categories in order to advertise the upcoming discounts on certain products. This way, more customers will visit his store and by placing products close to each other or in different hallways and selves, he can create more opportunities for users to be tempted to buy products that they may have not have included in their shopping lists in the first place. In the second case considered, students and teachers use ARNet to discover how students exams performance in courses of the first semester tend to affect their performance in courses in the second semester of their study. In each case considered, data are seen to be missing in order to reach/make solid conclusions just by observing and interpreting the association rules network as it is visualized by the ARNet application.

6 Conclusion and Future Work

Following the data processing stage and the extraction of new information from the data, a visualization of the association rules network is necessary in order to review the progress made and understand the data. In this thesis, most of the existing (R provided) data visualization methods have been considered. In addiction, an extra (network) data visualization method is introduced, code-named ARNet. The goal of this thesis, was to develop a visualization method that not only combines all the advantages of the existing visualization methods but to also provide even more possibilities to reveal additional information. The *ARNet* method makes possible the visualization of more than three attributes of the data involved. So far, only the interest measures were included in the visualization of the association rules but it is possible to visualize other attributes too. For example, in the case of the visualization of the association rules, mined from the data of the academic exams, the possibility to visualize the grade of each course using the thickness of each Edge was considered. Moreover, it utilizes *visNetwork*, a package/library that supports an extended interactive network view that allows users to explore the data with ease, and to save instances of the data in order to contrast and compare them with other, relevant, data samples.

There exist many possibilities for improving the ARNet method and application. First of all, it can be improved to become more adaptive to various types of input data. Certain data input may cause problems to the ARNet visualization process; the issue remains to be tackled in a future (improved) version of the method. Also, the use of the Shiny reactive programming model can be improved in order to provide a more user friendly filtering approach. As mentioned previously, when the user applies filters on the interest measures of the rules, ARNet will display the same number of rules if there is the same number of rules that meet the set requirements. This may confuse users or dissatisfy them with the final result. It is worth noting that there exists many of features of the *visNetwork* package that are not utilized by ARNet. For example, the thickness of the Edges is a feature that one can utilize to visualize an additional interest measure. Another example, is the use of icons for each Node that it may help users to identify the identity of each cluster.

Finally, the ARNet visualization of the association rules of the Groceries data is available in https://kpetrell.shinyapps.io/final/ and the visualization of the association rules of the Academic Grades data is available in https://kpetrell.shinyapps.io/Grades/ .

7 Bibliography

[1] L. Software, *Data Mining and Its Importance,* Loginworks Software, 2014. [Available]. https://www.loginworks.com/blogs/217-data-mining-and-its-importance/

[2] Randula Korolage, *Data Mining Techniques for Credit Card Fraud*, Moratuwa, 2019

[3] Charu C. Aggarwal , *Data Mining, The Textbook*, New York: Springer, 2015

[4] Paulo J. Azevedo, Al´ıpio M. Jorge, *Comparing Rule Measures for Predictive Association Rules*, 2007

[5] Max Bramer, *Principles of Data Mining, Third Edition*, London: Springer Nature, 2016

[6] Michael Hahsler, *arulesViz: Interactive Visualization of Association Rules*, The R Journal, 9, 163-175, 2017

[7] Konstantinos Kelesidis, *A Prototype for the Analytical Processing of Academic Examinations Data*, MSc Thesis, Supervisor: D.A. Dervos, Department of Information and Electronic Engineering, International Hellenic University, 2020

8 Appendices

8.1 ARNet code

app.R

petre

2020-09-16

Loading the necessary libraries.

```
library(shiny)
library(visNetwork)
library(dplyr)
library(shinyjs)
library(DT)
library(stringr)
library(stringi)
library(rsconnect)
library(BBmisc)
```

Identifying the user click input.

```
jscode <- '
$(function() {
  $(document).click(function(e) {
  if(event.shiftKey){
   Shiny.onInputChange( "clickType" , 2)
  }
   else if(event.altKey){
   Shiny.onInputChange( "clickType" , 3)
   }else if(event.ctrlKey){
   Shiny.onInputChange( "clickType" , 4)
   }
   else{
  Shiny.onInputChange( "clickType" , 1)
  }
  });
});
```

Preparing Nodes dataframe for the Start page of ARNet depending on the cluster presentetion mode input.

```
nodesAllRules <- function(topRules, status, prParam) {
  if (status == "in") {
    totalRows <- length(unique(topRules$head)) + nrow(topRules)
    Nodes <-
      data.frame(
        id = rep(0, totalRows),
        node = rep(0, totalRows),
        label = rep(0, totalRows),
        size = rep(0, totalRows),
        color = rep(0, totalRows),
        title = rep(0, totalRows)
      )
    Nodes$node <- c(unique(topRules$head), topRules$body)
    Nodes[, "label"] <- Nodes[, "node"]
    for (row in 1:length(unique(topRules$head))) {
      Nodes[row, "id"] <- -row
      Nodes[row, "size"] <- 30
      Nodes[row, "color"] <- "rgba(235, 0, 0, 1)"
    }
    if (prParam == 1) {
      normSize <- normalize(topRules$lift,
                            method = "range",
                            range = c(8, 25))
      normColor <-
        normalize(topRules$confidence,
                  method = "range",
                  range = c(0.1, 1))
      for (row in 1:nrow(topRules)) {
        Nodes[(row + length(unique(topRules$head))), "id"] <-
          topRules[row, "id"]
        Nodes[(row + length(unique(topRules$head))), "size"] <-
          normSize[row]
        Nodes[(row + length(unique(topRules$head))), "color"] <-
          paste("rgba(235, 0, 0,", normColor[row], ")")

      }
    } else if (prParam == 2) {
      normSize <-
        normalize(topRules$support,
                  method = "range",
                  range = c(8, 25))
      normColor <-
        normalize(topRules$lift,
                  method = "range",
                  range = c(0.1, 1))
      for (row in 1:nrow(topRules)) {
```

```
      Nodes[(row + length(unique(topRules$head))), "id"] <-
        topRules[row, "id"]
      Nodes[(row + length(unique(topRules$head))), "size"] <-
        normSize[row]
      Nodes[(row + length(unique(topRules$head))), "color"] <-
        paste("rgba(235, 0, 0,", normColor[row], ")")

    }
  } else if (prParam == 3) {
    normSize <-
      normalize(topRules$support,
                method = "range",
                range = c(8, 25))
    normColor <-
      normalize(topRules$confidence,
                method = "range",
                range = c(0.1, 1))
    for (row in 1:nrow(topRules)) {
      Nodes[(row + length(unique(topRules$head))), "id"] <-
        topRules[row, "id"]
      Nodes[(row + length(unique(topRules$head))), "size"] <-
        normSize[row]
      Nodes[(row + length(unique(topRules$head))), "color"] <-
        paste("rgba(235, 0, 0,", normColor[row], ")")

    }
  }
}
else if (status == "out") {
  totalRows <- length(unique(topRules$body)) + nrow(topRules)
  Nodes <-
    data.frame(
      id = rep(0, totalRows),
      node = rep(0, totalRows),
      label = rep(0, totalRows),
      size = rep(0, totalRows),
      color = rep(0, totalRows)
    )
  Nodes$node <- c(unique(topRules$body), topRules$head)
  Nodes[, "label"] <- Nodes[, "node"]
  for (row in 1:length(unique(topRules$body))) {
    Nodes[row, "id"] <- -row
    Nodes[row, "size"] <- 30
    Nodes[row, "color"] <- "rgba(235, 0, 0, 1)"
  }
  if (prParam == 1) {
    normSize <- normalize(topRules$lift,
```

```
                method = "range",
                range = c(8, 25))
  normColor <-
    normalize(topRules$confidence,
              method = "range",
              range = c(0.1, 1))
  for (row in 1:nrow(topRules)) {
    Nodes[(row + length(unique(topRules$body))), "id"] <-
      topRules[row, "id"]
    Nodes[(row + length(unique(topRules$body))), "size"] <-
      normSize[row]
    Nodes[(row + length(unique(topRules$body))), "color"] <-
      paste("rgba(235, 0, 0,", normColor[row], ")")
  }
} else if (prParam == 2) {
  normSize <-
    normalize(topRules$support,
              method = "range",
              range = c(8, 25))
  normColor <-
    normalize(topRules$lift,
              method = "range",
              range = c(0.1, 1))
  for (row in 1:nrow(topRules)) {
    Nodes[(row + length(unique(topRules$body))), "id"] <-
      topRules[row, "id"]
    Nodes[(row + length(unique(topRules$body))), "size"] <-
      normSize[row]
    Nodes[(row + length(unique(topRules$body))), "color"] <-
      paste("rgba(235, 0, 0,", normColor[row], ")")

  }
} else if (prParam == 3) {
  normSize <-
    normalize(topRules$support,
              method = "range",
              range = c(8, 25))
  normColor <-
    normalize(topRules$confidence,
              method = "range",
              range = c(0.1, 1))
  for (row in 1:nrow(topRules)) {
    Nodes[(row + length(unique(topRules$body))), "id"] <-
      topRules[row, "id"]
    Nodes[(row + length(unique(topRules$body))), "size"] <-
      normSize[row]
    Nodes[(row + length(unique(topRules$body))), "color"] <-
```

```
        paste("rgba(235, 0, 0,", normColor[row], ")")
      }
    }
  }
  for(row in 1:nrow(Nodes)){
    Nodes[row,"title"]<- Nodes[row,"label"]
  }
  return(Nodes)
}
```

Preparing Nodes dataframe for the Start page of ARNet depending on the cluster presentetion mode input.

```
edgesAllRules <- function(topRules, status, Nodes, pRparam) {
  if (status == "in") {
    Edges <-
      data.frame(
        from = rep(1, nrow(topRules)),
        to = rep(1, nrow(topRules)),
        arrows = c("to"),
        length = rep(1, nrow(topRules))
        # value = rep(1, nrow(topRules))
      )
    for (row in 1:nrow(topRules)) {
      test <- data.frame(filter(Nodes, node == topRules[row, "head"]))

      Edges[row, "to"] <- test[1, "id"]
      Edges[row, "from"] <- topRules[row, "id"]
    }
  }
  else if (status == "out") {
    Edges <-
      data.frame(
        from = rep(1, nrow(topRules)),
        to = rep(1, nrow(topRules)),
        arrows = c("to"),
        length = rep(1, nrow(topRules))
        # value = rep(1, nrow(topRules))
      )
    for (row in 1:nrow(topRules)) {
      test <- data.frame(filter(Nodes, node == topRules[row, "body"]))
      Edges[row, "from"] <- test[1, "id"]
      Edges[row, "to"] <- topRules[row, "id"]
```

```
  }
 }
 if (pRparam == 1) {
  for (row in 1:nrow(topRules)) {
   Edges[row, "length"] <-
    (topRules[row , "support"] * 1000) + (row * 5) + 50

   # Edges[row, "value"]<- (topRules[row , "support"] * 100)
  }
 } else if (pRparam == 2) {
  for (row in 1:nrow(topRules)) {
   Edges[row, "length"] <-
    (topRules[row , "confidence"] * 100) + (row * 5) + 50
   # Edges[row, "value"]<- (topRules[row , "confidence"] * 10)
  }
 } else if (pRparam == 3) {
  for (row in 1:nrow(topRules)) {
   Edges[row, "length"] <-
    (topRules[row , "lift"] * 10) + (row * 5) + 50
   # Edges[row, "value"]<- (topRules[row , "lift"]) * 2
  }
 }
 return(Edges)
}
```

UI layout of ARNet.

```
ui <- fluidPage(
 useShinyjs(),
 tags$head(
  tags$script(HTML(jscode)),
  tags$link(rel = "stylesheet", type = "text/css", href = "assocStyle.css")
 ),
 # App  ----
 titlePanel("Visualizing Association Rules with visNetwork and Shiny"),
 # Sidebar layout with input and output definitions ----
 sidebarLayout(
  # Sidebar panel for inputs ----
  sidebarPanel(
   id = "leftPanel",
   radioButtons(
    "ruleParameter",
```

```
    label = h4("Primary rule measure"),
    choices = list(
      "Support" = "1",
      "Confidence" = "2",
      "Lift" = "3"
    ),
    selected = 1
  ),
  uiOutput("supportB"),
  uiOutput("confidenceB"),
  uiOutput("liftB"),
  sliderInput(
    "numRules",
    label = h4("Number of top rules (N)"),
    min = 1,
    max = nrow(df),
    value = 10,
    step = 1
  ),
  radioButtons(
    "fanMode",
    label = h4("Cluster presentation mode"),
    choices = list("Fan In" = "in", "Fan Out" = "out"),
    selected = "in"
  )
  ,
  width = 2
)
,
# Main panel for displaying outputs ----
mainPanel(id = "midPanel",
          fluidRow(
            id = "mainRow",
            column(10, offset = 0,

                   tabsetPanel(
                     type = "tabs",
                     tabPanel(
                       "VisNetwork",
                       visNetworkOutput("network", height = "70vh"),
                       dataTableOutput("table")
                     ),

                     tabPanel("About", includeHTML("about.html"))
                   )),
            column(
              2,
```

```
        sidebarPanel(
          uiOutput("bodies"),
          uiOutput("heads"),
          sliderInput(
            "itemsetLength",
            label = h4("Body itemset length"),
            min = 1,
            max = 10,
            value = c(1, 10)
          ),
          downloadButton("downloadData", "Export table to .csv"),
          br(),
          br(),
          br(),
          tags$div(
            id = "rightBot",
            tags$p("Shift+click: Fan Out of outer clicked node"),
            tags$p("Alt+click: Fan In of outer clicked node"),
            tags$p("Ctrl+click: isolate the clicked cluster"),
            htmlOutput("info"),
            tags$p("Edge's Length: Primary rule's measure")
          )
        )
      )
    ))
  )
)
```

Constructing visNetwork visualization, utilizing visEvents(), visOptions(), visPhysics() and visInteraction functions.

```
  # network ----------------------------------------------------------------
  output$network <- renderVisNetwork({
    req(input$ruleParameter)
    req(input$numRules)
    req(input$bodyRule)
    req(input$headRule)
    req(input$itemsetLength)

    topRules <- data.frame()
    topRules <- switch(
      input$ruleParameter,
      "1" = values$compleTotal[order(-values$compleTotal$support), ],
      "2" = values$compleTotal[order(-values$compleTotal$confidence), ],
      "3" = values$compleTotal[order(-values$compleTotal$lift), ]
```

```
  )
  topRules <- head(topRules, input$numRules)
  values$currentRules = topRules
  values$currentBody <- unique(topRules$body)
  values$currentHead <- unique(topRules$head)
  if (input$bodyRule == "All Body Rules" &&
     input$headRule == "All Head Rules" &&
     session$userData$trackActivity$lastClick != 4) {
    if (!is.null(session$userData$allRules) &&
       input$itemsetLength[2] - input$itemsetLength[1] ==
max(str_count(session$userData$allRules$body, ",")))
    {

    } else{
      values$Nodes <-
       nodesAllRules(topRules, input$fanMode, input$ruleParameter)
      values$Edges <-
       edgesAllRules(topRules,
                     input$fanMode,
                     values$Nodes,
                     input$ruleParameter)
    }
  }
  else if (input$bodyRule != "All Body Rules" &&
          session$userData$trackActivity$lastClick != 4 &&
          input$itemsetLength[1] == 1) {
    values$Nodes <-
     nodesOneRuleBody(topRules, input$bodyRule, input$ruleParameter)

    values$Edges <-
     edgesOneRuleBody(topRules,
                      values$Nodes,
                      input$bodyRule,
                      input$ruleParameter)

  } else if (input$headRule != "All Head Rules" &&
           session$userData$trackActivity$lastClick != 4 &&
           input$itemsetLength[1] == 1) {
    values$Nodes <-
     nodesOneRuleHead(topRules, input$headRule, input$ruleParameter)

    values$Edges <-
     edgesOneRuleHead(topRules,
                      values$Nodes,
                      input$headRule,
                      input$ruleParameter)
  }
```

```
  visNetwork(values$Nodes, values$Edges)  %>%
    visEvents(
      click = "function(values$Nodes) {
            Shiny.onInputChange('clicked_node_id', values$Nodes.nodes);
            Shiny.onInputChange('clickType', 0);

            ;}"
    ) %>% visEdges(smooth = FALSE) %>%
    visOptions(
      highlightNearest = list(enabled = TRUE, labelOnly = FALSE),
      selectedBy = list(variable = "node")
    ) %>%
    visPhysics(
      solver = "barnesHut" ,
      barnesHut = list(
        gravitationalConstant = -50,
        centralGravity = 0,
        avoidOverlap = 0.3
      )
    ) %>% visInteraction(selectConnectedEdges = FALSE) %>% visExport("network", type = "png")
})
```

The construction of the interactive DataTable of ARNet.

```
  # table Output ----------------------------------------------------------
  output$table <- DT::renderDataTable({
    req(input$bodyRule)
    req(input$headRule)
    req(input$numRules)
    # print(session$userData$trackActivity$lastClick)
    if (!is.na(session$userData$trackActivity$lastClick)) {
      if (session$userData$trackActivity$lastClick == 1) {
        if (is.null(input$clicked_node_id)) {
          if (is.null(session$userData$allRules)) {
            values$printTable <- values$specificRules
          } else{
            values$printTable <- session$userData$allRules
          }
        } else{
          values$printTable
        }
      } else if (session$userData$trackActivity$lastClick == 2 ||
                 session$userData$trackActivity$lastClick == 3) {
```

```
    values$printTable <- values$specificRules
   } else if (session$userData$trackActivity$lastClick == 0) {
    if (is.null(session$userData$allRules)) {
     values$printTable <- values$specificRules
    } else{
     values$printTable <- session$userData$allRules
    }
   } else if (session$userData$trackActivity$lastClick == 4) {
    values$printTable <- values$specificRules
   }
  } else{
   values$printTable <- values$specificRules
  }

  if (is.null(input$clickedType) &&
     is.null(input$clicked_node_id) &&
     (input$bodyRule == "All Body Rules" &&
      input$headRule == "All Head Rules")) {
   # print(is.null(session$userData$allRules))
   if (is.null(session$userData$allRules)) {
    values$printTable <- values$currentRules
   } else{
    values$printTable <- session$userData$allRules
   }
  }
  DT::datatable(values$printTable)

})
observeEvent(input$table_rows_selected, {
 peakTable <- values$printTable[input$table_rows_selected,]
 # View(peakTable)
 visNetworkProxy("network") %>% visSelectNodes(id = c(peakTable$id))
})
```