Dependency Visualization in Data Stream Profiling

Bernardo Breve, Loredana Caruccio*, Stefano Cirillo,
Vincenzo Deufemia, Giuseppe Polese

Department of Computer Science, University of Salerno
via Giovanni Paolo II, 132, 84084 Fisciano (SA)
{*bbreve,lcaruccio,scirillo,deufemia,gpolese*}*@unisa.it*

Abstract

Data stream profiling concerns the automatic extraction of metadata from a data stream, without having the possibility to store it. Among the metadata of interest, functional dependencies (FDs), and their extensions relaxed functional dependencies (RFDs), represent an important semantic property of data. Nowadays, there are many algorithms for automatically discovering them from static datasets, and some are being proposed for data streams. However, one of the main problems is that the stream nature of data requires a different paradigm of monitoring, since the "big" number of (R)FDs that might hold on a given dataset continuously change as new data are read from the stream. In this paper, we present a tool for visualizing RFDs discovered from a data stream. The tool permits to explore results for different types of RFDs, and uses quantitative measures to monitor how discovery results evolve. Moreover, the tool enables the comparison among RFDs discovered across several executions, also proving visual manipulation operators to dynamically compose and filter results. A user study has been conducted to assess the effectiveness of the proposed visualization tool.

Keywords: data stream profiling, big data visualization, metadata visualization, continuous discovery, relaxed functional dependencies

This is a post-peer-review, pre-copyedit version published in Big Data Research (BDR) Journal, Elsevier. 25: 100240 (2021). The final authenticated version is available online at: https://doi.org/10.1016/j.bdr.2021.100240

*Corresponding author

1. Introduction

In the last few years, big data analytics has drawn the attention of database researchers and stakeholders from many application domains, and this trend is expected to continue in the future, since the volume of daily generated data is continuously growing. This entails the necessity to adopt efficient techniques to integrate, inspect, and clean the big datasets feeding data analytics and machine learning processes. To this end, metadata have been recognized as a fundamental mean to assess the quality and integrity of data, ranging from basic statistics on domain distributions and cardinalities to more complex properties of data, such as data dependencies [1]. The automatic discovery of such metadata from big data collections is the goal of data profiling [1], which is preparatory to most big data analytic tasks.

Among the big data analytic tasks, particular attention is devoted to those aiming to learn predictive models from data streams. For instance, many countries are interested in monitoring the streams of data concerning their electricity consumption, aiming to predict the electrical power provisioning needed in the near future. Similarly, predictive models from streams of sensor-generated data are invoked in several domains, such as civil constructions, healthcare, and so on. However, as opposed to static datasets, in this context it is not possible to store the whole set of training data, hence it is necessary to dynamically profile data streams and learn models from them, trading off with precision. To this end, it would be useful having tools capable of visualizing the dynamic evolution of the metadata characterizing the profile of a data stream.

In this paper, we focus our interest on data stream profiling. To this end, among the different metadata characterizing the profile of a data stream, we focus on functional dependencies (FDs), and their extension relaxed functional dependencies (RFDs). The latter relax some constraints of canonical FDs by admitting the possibility for them to hold on a subset of data (also referred to as RFDs relaxing on the *extent*), and/or by relying on approximate paradigms to compare pairs of tuples (also referred to as RFDs relaxing on the *attribute com-*

parison) [2]. After many proposals of algorithms for automatically discovering (R)FDs from datasets [3, 4, 5, 6], there are some recent proposals of algorithms dealing with their discovery from dynamic datasets [7, 8]. However, due to the continuous variation of RFDs that might hold in different instants of time, it is necessary to devise methods for monitoring their evolution as the stream keeps providing new data to be profiled. One way to do it is to graphically visualize the evolution of holding RFDs over time. To this end, we present STRADYVAR (STReAm DependencY VisuAlizeR), a tool for analyzing and comparing RFDs dynamically extracted from data streams, enabling users to monitor their evolution over time. In particular, withSTRADYVAR the user can (*i*) visualize the trend related to the number of RFDs holding on a data stream over time, (*ii*) visualize an overview of the correlation between attributes included in the discovery results over time, (*iii*) compare the RFDs resulting from different executions of a discovery process on the same or different data streams, and (*iv*) directly manipulate discovery results by composing those of different executions and/or grouping RFDs according to a specific Right-Hand-Side (RHS) attribute.

The paper is organized as follows. Section 2 describes related visualization approaches and tools. Section 3 presents definitions and theoretical foundations of FDs and RFDs. Section 4 presents STRADYVAR, whereas Section 5 details the user study performed to evaluate the effectiveness of the proposed visualization strategies. Finally, summary and concluding remarks are included in Section 6.

2. Related Work

The availability of efficient RFD (or FD) discovery algorithms yields the necessity to manage big result sets, most of which differing only for some parameters of RFDs. Only recently such algorithms are becoming capable to scale over big data sources, but there are not many solutions in the literature for handling the complexity related to the visualization of possibly huge numbers of discovered RFDs. Among these, one of the most effective is Metanome [9], a data profiling platform embedding several algorithms to automatically discover complex meta-

data, including functional and inclusion dependencies. Moreover, it embeds various result management techniques, such as list-based ranking techniques, and interactive diagrams of discovery results. To this end, a representative scalable platform for analyzing data profiles is Metacrate [10], which permits the storage of different meta-data and their integration, enabling users to perform several ad-hoc analysis. In this context, the first proposal for visualizing large sets of RFDs is described in [11]. It presents several metaphors for representing RFDs at different levels of detail. Starting from a high-level visualization of attribute correlations, details are interactively revealed, also including details on the RFD relaxation criteria.

A related research context is represented by visual data mining. In the literature, many approaches and tools aiming to improve the understanding of data mining algorithms and their results (see [12] for a survey). Among these, it is worth to mentioning Association Rules (ARs) visualization approaches, since the concept of AR is somehow related to that of RFD. For instance, several tools/packages have been designed to visually inspect the set of ARs [13, 14]. Moreover, ad-hoc visualization techniques have been introduced, such as the hierarchical matrix-based [15], or the hybrid matrix- and graph-based [16]. They typically show summarized representations of result sets, and provide mechanisms to filter results. On the other hand, due to the huge quantity of possible rules, in [17] a visual tool for searching rules has been introduced. It interactively guides users in the definition of the target rule by drawing details of minimum, current, and potential support/confidence measures, based on already or potentially selected attributes. Finally, also in the context of ARs, an enhanced visualization technique has been defined in order to highlight changes among different sets of ARs [18].

Although all of these approaches represent effective tools to visualize and explore properties and metadata after the execution of mining/discovery algorithms, our proposal enables users to monitor how RFDs change over time and to perform result comparisons across different executions. This meets the recent need of researchers to focus on more complex scenarios and issues, such as the

possibility to explore how data change [19] and to perform continuous profiling [1]. For these reasons, even if in the literature several time-related visualization approaches/tools have been proposed [20, 21, 22], they focus on static scenarios. Thus, to the best of our knowledge STRADYVAR is the first proposal that enables the exploration and comparison of RFD discovery results in dynamic scenarios.

3. Preliminaries

Before describing the proposed tool, we will review the definition of FD, and the general definition of RFD.

FD *definition.* Let $\mathcal{R}$ be a relational database schema defined over a set of attributes $attr(\mathcal{R})$, and r an instance of $\mathcal{R}$, then we use $\{A, B, C, \dots\}$ to denote single attributes in $attr(\mathcal{R})$, $\{X, Y, W, \dots\}$ sets of attributes in $attr(\mathcal{R})$, $t[A]$ and $t[X]$ the projection of t onto A and X, respectively. An FD over $\mathcal{R}$ is a statement $X \rightarrow Y$ (X implies Y), such that, given an instance r of $\mathcal{R}$, $X \rightarrow Y$ is satisfied in r if and only if for every pair of tuples (t_1, t_2) in r, whenever $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$. X and Y represent the Left-Hand-Side (LHS) and Right-Hand-Side (RHS) of the FD, respectively.

Starting from the *canonical* definition of FD, over 35 definitions of extended FDs have been provided in the literature, which has been generalized yielding the concept of *Relaxed Functional Dependency* (RFD) [2]. In particular, RFDs rely on two main relaxation criteria. The first one generalizes the equality constraint used into the FD definition to compare the projections of pair of tuples over a subset of attributes, introducing the possibility to consider a predicate evaluating the distance or similarity between them. Thus, it involves a similarity or distance function, such as the Jaro or the Edit distances [23], whose results are verified according to a comparison operator and a threshold.

The second relaxation criterion, defined as relaxation on the *extent*, admits the possibility that a dependency might hold on a subset of tuples rather than on the entire dataset. In particular, the satisfiability degree of an RFD can be formally specified by means of conditions restricting the application domain of

the RFD or through a *coverage measure*. The latter measures the minimum number or percentage of tuples on which the RFD must hold in order to be considered valid. Examples of widely used coverage measures are confidence and $g3$-error [24].

RFD *definition.* Consider a relational database schema $\mathcal{R}$, and a relation schema $R = (A_1, \ldots, A_m)$ of $\mathcal{R}$. An RFD φ on $\mathcal{R}$ is denoted by

$$\mathbb{D}_c : X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} Y_{\Phi_2} \tag{1}$$

where

- $\mathbb{D}_c = \left\{ t \in dom(R) \mid \bigwedge_{i=1}^{m} c_i(t[A_i]) \right\}$
  with $c = (c_1, \ldots, c_m)$, and each c_i is a predicate on $dom(A_i)$ filtering the tuples on which φ applies;
- $X = B_1, \ldots, B_h$ and $Y = C_1, \ldots, C_k$, with $X, Y \subseteq attr(R)$ and $X \cap Y = \emptyset$;
- $\Phi_1 = \bigwedge_{B_i \in X} \phi_i[B_i]$ ($\Phi_2 = \bigwedge_{C_j \in Y} \phi_j[C_j]$, resp.), where ϕ_i (ϕ_j, resp.) is a conjunction of predicates on C_i (C_j, resp.) with $i = 1, \ldots, h$ ($j = 1, \ldots, k$, resp.). For any pair of tuples $(t_1, t_2) \in dom(R)$, the constraint Φ_1 (Φ_2, resp.) is true if $t_1[B_i]$ and $t_2[B_i]$ ($t_1[C_j]$ and $t_2[C_j]$, resp.) satisfy the constraint ϕ_i (ϕ_j, resp.) $\forall\ i \in [1, h]$ ($j \in [1, k]$, resp.). Each ϕ constraint is composed of a similarity/distance function, an operator, and a similarity/distance threshold.
- Ψ is a coverage measure defined on $dom(R)$, quantifying the number of tuples violating or satisfying φ. It can be defined as a function $\Psi : dom(X) \times dom(Y) \to \mathbb{R}^+$, where $dom(X)$ is the cartesian product of the domains of attributes composing X.
- ε is a threshold indicating the upper bound (or lower bound in case the comparison operator is $\geq$) for the result of the coverage measure (a.k.a. satisfiability degree threshold).

Given $r \subseteq \mathbb{D}_c$ a relation instance on R, r *satisfies* the RFD φ, denoted by $r \models \varphi$, if and only if: $\forall\ t_1, t_2 \in r$, if Φ_1 indicates true, then *almost always* Φ_2 indicates true. Here, *almost always* is expressed by the constraint $\Psi \leq \varepsilon$. The latter maps the semantic interpretation of the *implication property* for RFDs. A more general definition covering more types of RFDs has been provided in [2].

In the following, we will use the acronym RFD to refer to FDs, RFDs relaxing on the tuple comparison only, RFDs relaxing on the extent only, by means of a coverage measure, and RFDs relaxing on both. In fact, they can all be described through the equation (1). In this study, we do not consider RFDs relaxing on the extent through a condition of domain values.

For RFDs relaxing on the tuple comparison only, when no couple of tuples yields an RFD violation, the expression $\Psi(X, Y) = 0$ is omitted from the notation. Moreover, for sake of simplicity, in what follows we always consider the $g3$-error as coverage measure, the operator $\leq$ for tuple comparison, the edit distance for textual attributes, the absolute difference for numerical ones. Finally, without loss of generality, we can consider RFDs with a single value on the RHS.

As an example, in a database of tweets, it is likely to have the same number of followed accounts (#Friends) for accounts with the same name and location. Thus, an FD {Name, Loc}→ #Friends might hold. However, this property might also hold for names and locations stored using different abbreviations and/or similar numbers of friends, hence the following RFD might hold:

$$\{\mathsf{Name}_{\leq 2}, \mathsf{Loc}_{\leq 4}\} \rightarrow \mathsf{\#Friends}_{\leq 10}$$

Moreover, since accounts might change location during their account's life, or there might be changes in the number of friends, the previous RFD should tolerate possible exceptions. This can be modeled by introducing a different coverage measure into the RFD:

$$\{\mathsf{Name}_{\leq 2}, \mathsf{Loc}_{\leq 4}\} \xrightarrow{\psi(Name,Loc,\#Friends) \leq 0.03} \mathsf{\#Friends}_{\leq 10}$$

Among all RFDs holding on a given relation database schema $\mathcal{R}$, only a subset of them can be considered as the most meaningful ones. In fact, in the context of FDs, Armstrong's inference rules have been theoretically defined in order to derive the minimal set of FDs. The latter represents the set of FDs from which all valid ones can be derived. Into the context of RFDs, an RFD $X_{\Phi_1} \xrightarrow{\psi(X,A)\leq\varepsilon} A_{\phi_2}$ is said to be *minimal* if and only if 1) it is non-trivial, i.e. no attribute is common to its LHS and RHS; 2) it has the minimum possible number of LHS attributes; 3) it has LHS attributes with maximum possible threshold values; and 4) it has the RHS attribute with minimum possible threshold value.

4. A tool for analyzing and comparing RFD discovery results

In this section, we describe the proposed system STRADYVAR, whose architecture is presented in Section 4.1. Successively, we provide details on (*i*) the real-time monitoring visual interface, by also explaining the interactions that a user can perform with it (Section 4.2), (*ii*) the feature enabling the comparison between two different executions on multiple streams (Section 4.3), (*iii*) the interface enabling the exploration of resulting RFDs in terms of correlation between attributes (Section 4.4), and (*iv*) the feature enabling the dynamic exploration of RFDs possibly resulting from different executions (Section 4.5).

4.1. System architecture

There are several issues related to discovery processes that lead to specific choices for the system architecture: 1) the amount of RFDs processed at each point in time can be huge, 2) the presence of several visualization components could require frequent updates in a short time, and 3) discovery algorithms rely on different implementation technologies. To this end, STRADYVAR is based on a client-server architecture, and it has been designed to enable users to monitor results during the execution of RFD discovery algorithms through a responsive visual interface. Moreover, its modules are standalone and share information with other ones by using the JSON standard. In particular, it

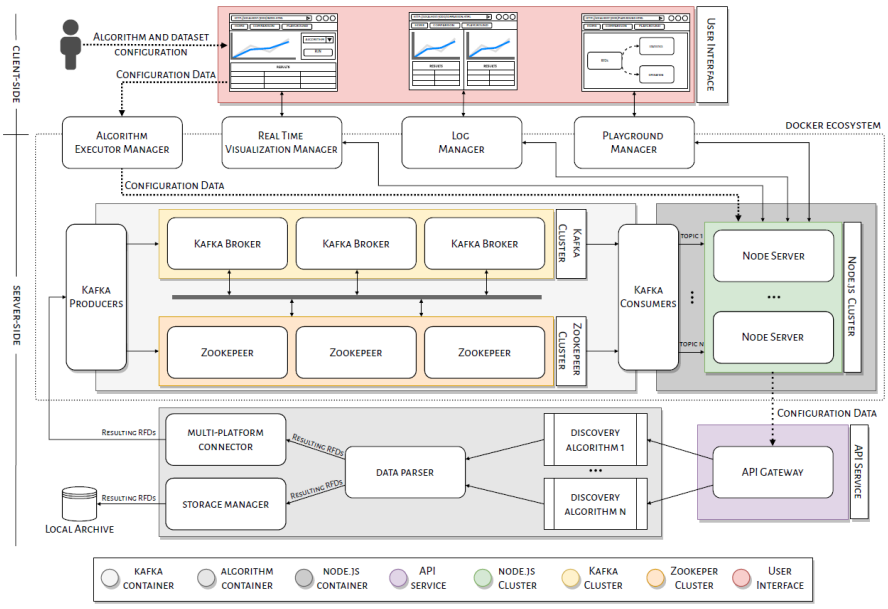

Figure 1: STRADYVAR architecture.

ensures high modularity and maintainability, for enabling the substitution of any back-end component, provided that its output is formatted according to the JSON standard defined for the interaction.

Figure 1 shows the architecture of STRADYVAR. The client communicates with the back-end modules through live queries, and is implemented as a web application consisting of three different interfaces: the first one enables the real-time monitoring of discovery processes; the second one enables the user to compare the results of two executions of the same discovery algorithm, or of two different ones; finally, the third component enables users to visually manipulate results across different executions.

In general, STRADYVAR permits either to load a dataset or a configuration file connecting to a data stream, and depending on the choice, select a discovery algorithm for datasets or data streams. Its back-end is based on a microservices architecture, exploiting the power and the flexibility of Docker[1] containers in

[1] https://www.docker.com/

order to create a highly scalable platform. In particular, the core of STRADYVAR is a long-lived Node.js application, running distributed containers while keeping a live connection with other services. Although the architecture is flexible and scalable, in order to quickly process a large number of messages we integrated the broker Apache Kafka[2], which is a high-throughput and low-latency platform for handling real-time data feeds. In particular, Kafka temporarily stores key-value messages originating from several processes called *producers*.

Kafka requires the use of Zookeeper[3] servers to coordinate several producers and consumers. In fact, the architecture of STRADYVAR integrates these services into multiple docker containers, which directly communicate with the Node.js instances through Kafka consumers. Moreover, to make STRADYVAR compatible with most FD and RFD discovery algorithms, it is necessary to uniform the syntax of the RFDs, regardless of possible thresholds. In particular, a parser module receives in input an RFD, manipulates the RFD syntax to extract a JSON version of it, so as to store it in a local file and in a Kafka topic. All the selected technologies for both client- and server-side support real-time updating of data.

4.2. RFD*s visualization*

Systems for data relationship visualization vary in what they display and how users can interact with them. However, most of these data visualization systems are not intended for the dynamic processing of data, restricting themselves to a static representation of large datasets. However, in the context of continuous data profiling, useful insights can be gained by monitoring how RFDs evolve over time. Thus, the dynamic representation of a large portion of data requires the application of interactive graphs, capable of highlighting the arrival of new information without missing any pre-existing information.

Figure 2 shows the real-time monitoring interface of STRADYVAR, which permits to execute an incremental discovery algorithm, and to monitor the trend and the details of validated RFDs in real-time during the execution process. In

[2]https://kafka.apache.org/
[3]https://zookeeper.apache.org/

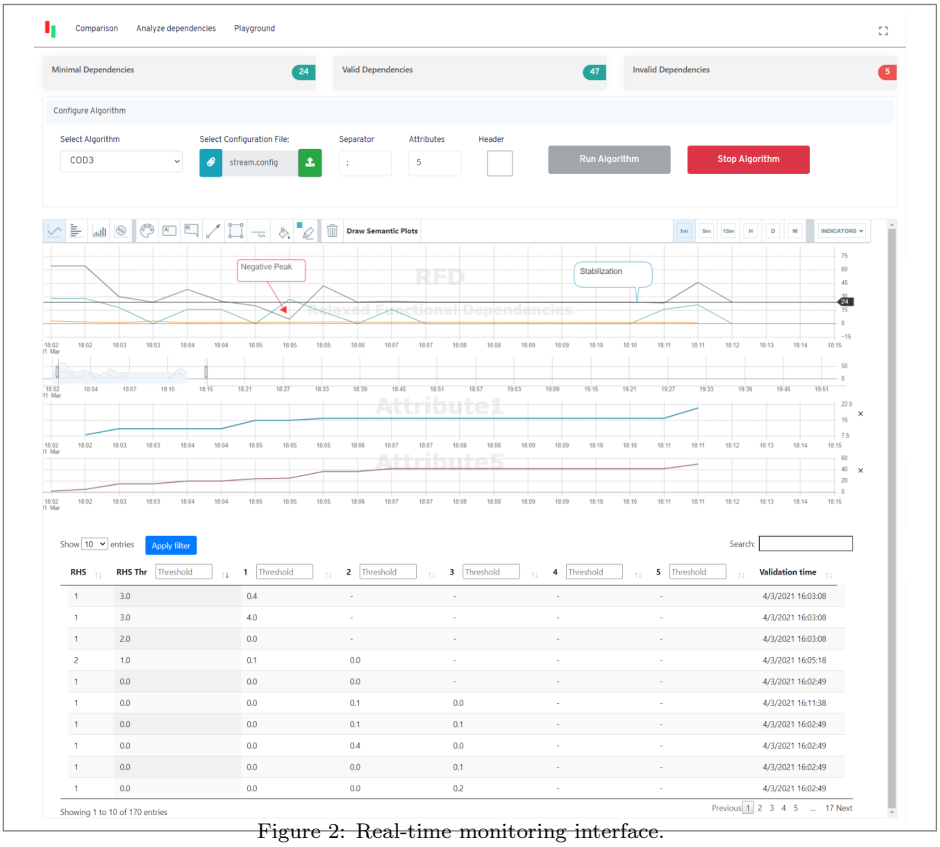

Figure 2: Real-time monitoring interface.

particular, on the top-left it contains the *line plot*, which is responsible for displaying information about the evolution of valid RFDs discovered over time. It can reveal stability or variability in the number of discovered RFDs, possibly highlighting variability in the correlation among data. Moreover, it becomes particularly useful in comparing trends across different executions and/or filtering results according to specific time periods. More specifically, for each time instant the line plot contains a black line, a blue line, and possibly an orange line, showing the number of holding RFDs, of invalidated RFDs, and of holding RFDs selected according to searching and/or filtering criteria, respectively. The plot is divided into two sections, the upper one is the main line plot, showing the evolution of the number of valid RFDs on its y-axis and the timestamp on its x-axis; although it is a replica of the upper one, the bottom line plot enables

the interaction with the user, allowing him/her to select a time interval through a brush operation, in order to visualize details on how the number of discovered RFDs changed during the selected period. Both these line plots are continuously updated so that at any time instant the user can see how the trend changes as the RFD discovery process progresses.

The *dependency table* displayed at the bottom of Figure 2 details the discovered RFDs. Each row represents an RFD, whereas each column represents an attribute of the dataset. In particular, the column labeled "RHS" contains the indices of the RHS attribute of each RFD. Instead, the other columns display details on all the attributes appearing in an RFD. In particular, the last column contains the time instant in which an RFD has been discovered, and the whole table is continuously updated. In other words, the dependency table is able to describe the implication property of an RFD and it also shows the difference thresholds, which specifically characterize each RFD.

Finally, the real-time monitoring interface also provides statistical counters (see the top of Figure 2) displaying the number valid, invalid, and minimal RFDs, at a given time instant.

Interaction in depth. As mentioned above, the user can interact with the interface through the bottom line plot by selecting time intervals. The selection is highlighted with a light blue rectangle on the bottom line plot. During the monitoring process the top line plot reacts consequently, reducing the scale on

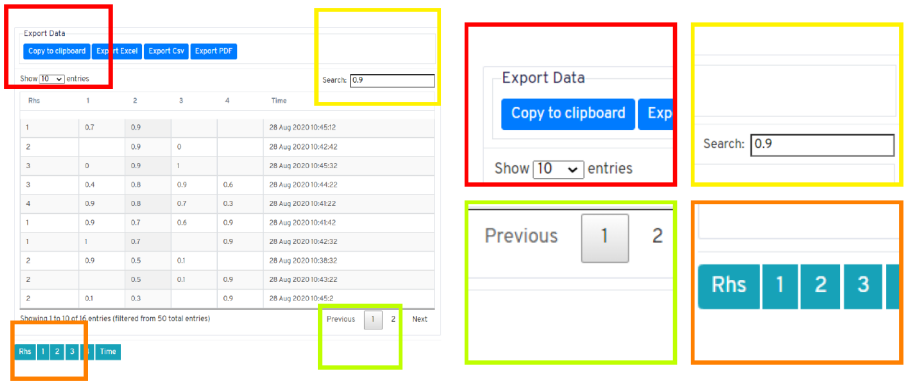

Figure 3: Interacting with the dependency table.

the x-axis, in order to adapt the range boundaries specified by the user and zoom on the line plot (see Figure 2). At any time instant, the user can verify whether the discovery process is still in progress by stopping the brushing process. Moreover, as shown in Figure 2, the user has the possibility to annotate the line plot with several visual components. In fact, on the top of the line plot there is a toolbar through which the user can i) draw lines, arrows, polygons, and so on, ii) add some textual notes, and/or iii) modify the color of the plot (see the last two buttons on the toolbar). Finally, the toolbar offers the possibility to select some attributes. When an attribute is selected, a novel line plot is added at the bottom of the general one, showing the number of holding RFDs that contain the selected attribute as RHS.

Concerning the dependency table, Figure 3 highlights all the interaction functionalities offered by STRADYVAR. In order to allow the user to reduce the table content, we added a search text field (Figure 3, yellow rectangle) enabling a global search over the attributes of the table, or a data filter by searching on a specific column of the table. Based on the text the user inserts in the search text field, the table content is adapted, showing only the rows satisfying all enabled selection criteria. Moreover, a new line is added on the line plot, showing the number of holding RFDs for each time instant, also satisfying the selected criteria. By default, the RFDs are sorted by means of the utility function defined in [25], but users can also define column-based sorting criteria by clicking on the chosen column and selecting either an ascending or descending order.

We also added a paging functionality (Figure 3, green rectangle), allowing the user to decide the number of rows to be displayed in a single table page. Moreover, the buttons below the table allow the user to decide which column to hide or show in the table (Figure 3, orange rectangle). By pressing one of such buttons, the user can hide the corresponding column and make the associated rectangle turn black. By clicking on it again, it reverses this process.

Finally, at any time instant, it is possible to copy and/or export the data contained into the dependency table to an external file (Figure 3, red rectangle). The export function supports several formats, such as Excel, CSV, and

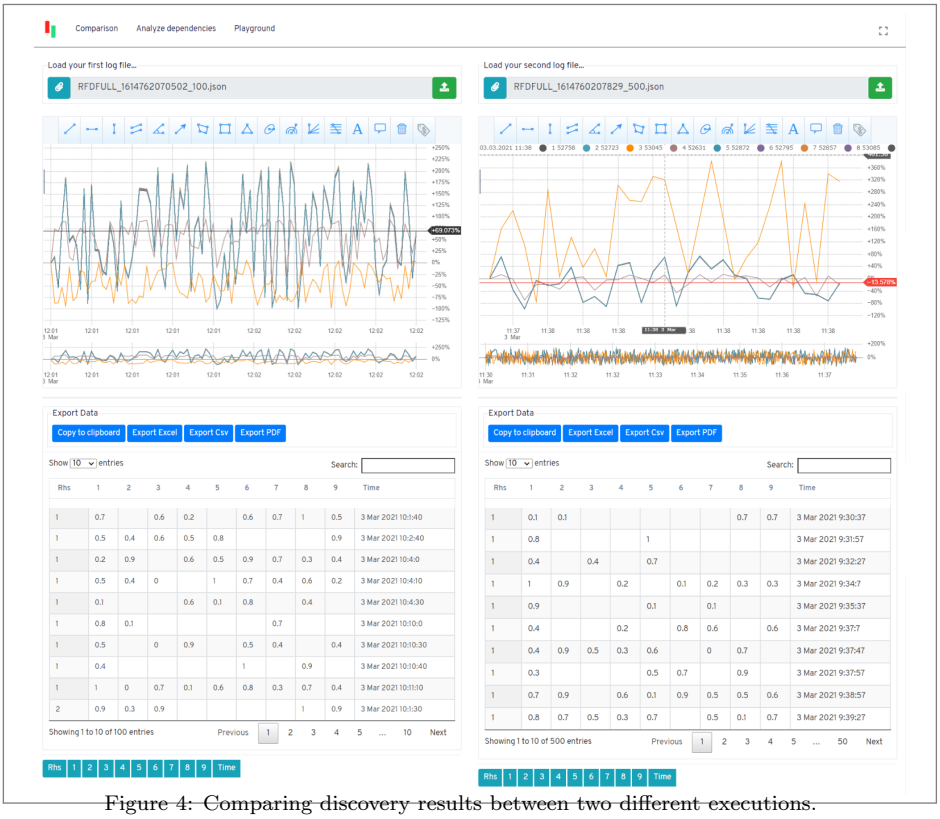

Figure 4: Comparing discovery results between two different executions.

PDF. The exported file will include all the data that have not been filtered out according to the user interactions.

By interacting with the dependency table also the line plot is updated. In fact, it is possible to visualize a proper line that shows the trend in the number of RFDs according to searching/filtering criteria specified by the users. It is worth noting that some searching/filtering criteria are specific for RFDs, such as those involving the difference thresholds.

4.3. Comparing RFDs

An important feature of STRADYVAR is the possibility to compare RFDs extracted across two different executions. It can involve different data analyzed in different time periods or results produced by different incremental discovery algorithms, which is made possible by loading log files from different executions.

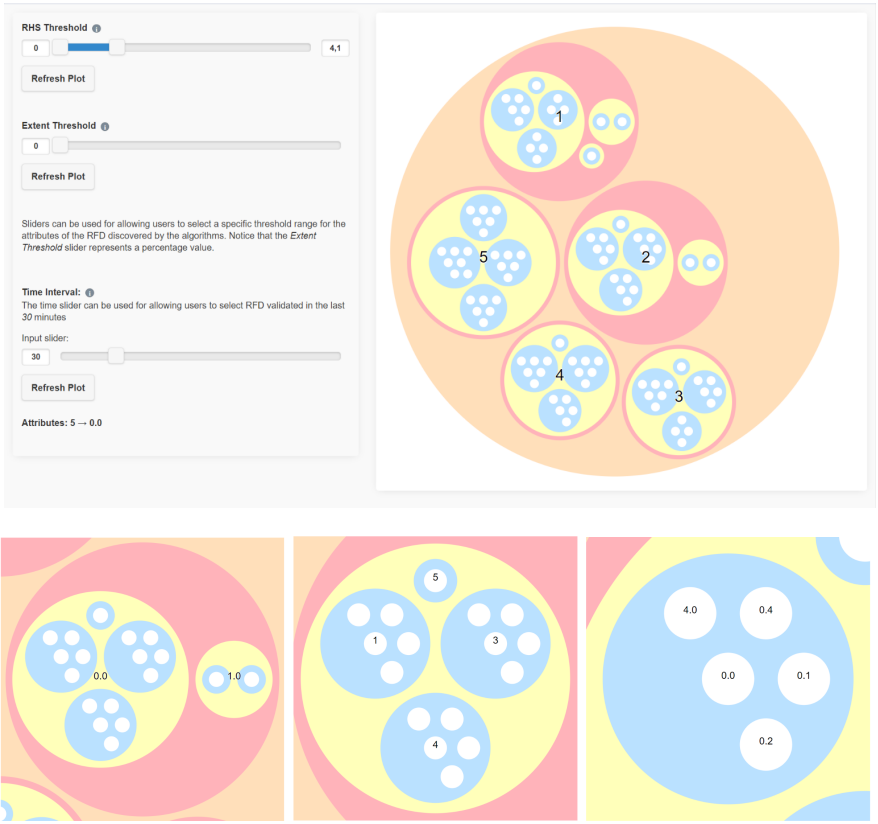

Figure 5: Analyzing the correlation among attributes according to holding RFDs.

Figure 4 shows the visual interface devoted to the comparison of discovery results. As we can see, both the line plot and the dependency table are duplicated. They contain the evolution of the number of RFDs (line plot), and details on RFDs holding during the execution loaded from a specific log file. In particular, each line plot shows lines concerning the evolution in the number of holding RFDs, and for each attribute the evolution of the subset of holding RFDs having it as RHS. Finally, by interacting with the bottom line plots, the user can select the specific time intervals to be analyzed from each execution. This enables the user to verify commonalities across different executions of discovery algorithms, comparing them in terms of both the changes in the number of discovered RFDs, and the RFDs validated at a specific time interval of the execution.

4.4. Correlation analysis

The monitoring interface (Figure 2) permits to show holding or invalidated RFDs according to the time variable and user-defined selection criteria. STRADYVAR also provides an additional overview of discovery results in terms of correlation among attributes.

Figure 5 shows the visual interface providing a general overview of attribute correlations. It represents each attribute as a circle, which in turn contains circles according to its characteristics and those of holding RFDs. In particular, the hierarchical chain of the circle containment, ranging from the most general to the most specific, is defined as: i) RHS attribute, ii) RHS difference threshold, iii) LHS attribute, and iv) LHS difference threshold. The correlation plot interactively responds to mouse clicks by zooming the circle the user would like to analyze (see Figure 5 at bottom). Several sliders on the left permit to filter out results according to an RHS difference threshold range (i.e., the difference thresholds bounding attributes on the RHS) or a satisfiability degree threshold (i.e., the upper bound of the percentage of admitted errors), which represent specific properties of RFDs. Moreover, the user can move the time interval to change the analyzed resulting set according to the time instant producing it.

4.5. Playground

The real-time monitoring and the possibility to compare discovery results with respect to the execution of possibly different discovery algorithms represent valid means through which users can effectively explore result details and peculiarities. One of the main issues in analyzing results of RFD discovery algorithms is the possibly huge quantity of returned RFDs. To this end, STRADYVAR provides interactive features enabling the user to reduce the set of discovered RFDs by removing columns from the dependency table, filtering results, and exporting extracted RFDs. The *playground* module provides additional functionalities to support post-execution analysis of discovered RFDs. This is made possible through a visual editor equipped with several visual components named

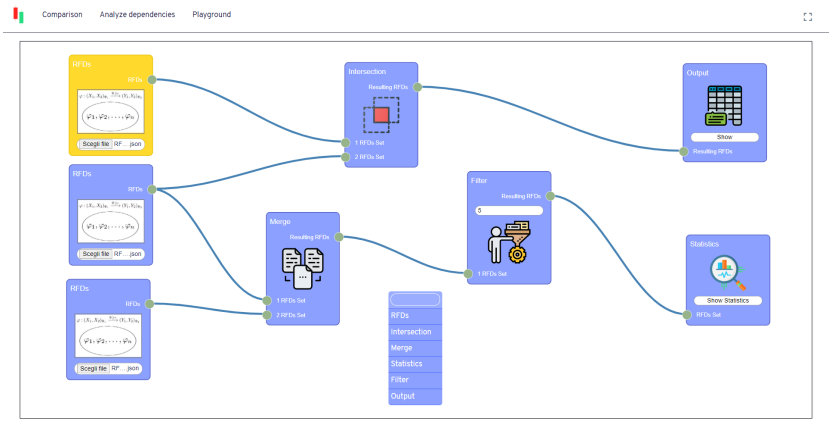

Figure 6: Overview of the Playground visual editor.

blocks, which enable transformations, and comparative/statistical analysis of RFDs discovered during one or multiple executions.

Figure 6 provides an overview of the playground module, whose aim is to enable the user to link different blocks to form a sequence of operations, so as to determine a data flow (composed of RFDs) that is gradually updated on the basis of involved operations. Multiple flows can be created at the same time; the only constraint is that each flow has to start with one or more sets of discovered RFDs as input, and it has to terminate in one of the blocks enabling the visualization of some outputs. Blocks show both the number of required inputs (on the left) and the expected outputs (on the right), if any, by means of tiny circles, named *input* and *output* connectors, respectively. The user can require to insert a specific block and connect it to other blocks by simply clicking on the playground. More specifically, the playground integrates different types of blocks, each representing a specific operation that can be applied to one (or more) set(s) of RFDs. Details on the blocks are provided in the following:

1. the block *RFDs* permits to consider a set of RFDs as input, enabling the selection of the execution results, and giving the possibility to forward the set of holding RFDs.
2. The block *Intersection* permits to forward only the RFDs shared between two sets. Consequently, it contains two inputs and one output connector.
3. The block *Merge* permits to forward all RFDs included in at least one of

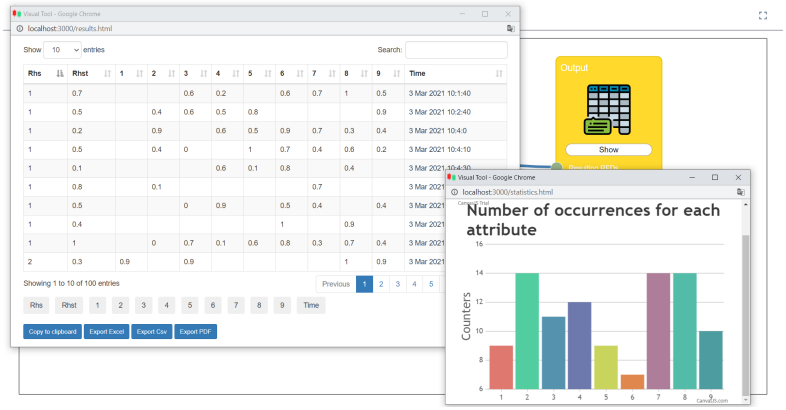

Figure 7: An example of the Statistics block usage.

the sets provided in input, like with a union operation. Consequently, it contains two inputs and one output connector.

4. The block *Filter* permits to specify an attribute used to filter and forward only the RFDs having it on their RHS. Consequently, it contains one input and one output connector.

5. The block *Statistics* permits to visualize the number of RFDs linked to its input connector, grouped by each RHS attribute by means of a bar plot. An example of results that can be obtained by using this block is shown in Figure 7.

6. The block *Output* permits to visualize the RFDs linked to its input connector, by means of a dependency table, and on which the user can apply all types of interactions described above. An example of results that can be obtained by using this block is shown in Figure 7.

5. User Study

The user study presented in this section aims to show that STRADYVAR makes the analysis of discovered RFDs over time a simple and effective process.

5.1. Method

We evaluated the proposed STRADYVAR with ten users (2 MS students, 4 Ph.D. students, and 4 company's DBAs) who performed all tasks involving the

Survey	Alias	Question
Background	**Q1**	Gender
	Q2	Qualification
	Q3	Level of knowledge of Data Profiling and metadata
	Q4	Level of knowledge of FDs
	Q5	Level of knowledge of RFDs
Evaluation	**Q6**	I quickly and easily executed different discovery algorithms over several types of datasets
	Q7	I completed the task T1 quickly and easily
	Q8	I completed the task T2 quickly and easily
	Q9	I completed the task T3 quickly and easily
	Q10	I completed the task T4 quickly and easily
	Q11	The tool is simple to use
	Q12	The tool is simple to learn
	Q13	I was able to retrieve back the process, whenever I made a mistake
	Q14	The user interface is pleasant and informative
	Q15	The tool made transparent the execution of the underlying discovery algorithms
	Q16	I understood the RFD syntax
	Q17	By using the tool, I can analyze the correlation among different attributes
	Q18	By using the tool, I can analyze the differences among different discovery results
	Q19	The tool presents all the features I expected
	Q20	I am in general satisfied about the tool
	Q21	In the future, I would like to use the tool
	Q22	What is your general impression about the tool?
	Q23	Do you have any improvements to suggest?
	Q24	Which feature of the tool did you like the least?
	Q25	Which feature of the tool did you like the most?

Table 1: Questions proposed to participants.

execution of two discovery algorithms [8, 25], on the public datasets Abalone and Bridges, drawn from the UCI Machine Learning repository[4], and two 8 hour sessions over the Twitter data streams. Statistics about participants have been collected through a background survey, as shown in Table 1(Q1-Q5). In particular, about 70% of recruited people were men, most of which graduated. Through a Likert scale, they declared a medium level of knowledge on data profiling, FDs, and RFDs (see Figure 8, Q3-Q5). Prior to the evaluation, they underwent a 30-min tutorial on data profiling and STRADYVAR. After performing a baseline task required to run several discovery algorithms by considering both data stream API configurations, and real-world datasets, we requested participants to perform the following tasks:

T1 *Monitoring:* Select a time interval of one hour ending with a negative peak, search for an attribute on the RHS, filter RFDs with threshold 0 on another attribute, and order dependencies by the RHS threshold.

T2 *Comparing:* Visualize RFDs resulting from different executions over Twitter data streams, and highlight (by using annotation visual components) at least two commonalities/differences on both the analyzed trends.

[4] `https://archive.ics.uci.edu/ml/datasets.php`

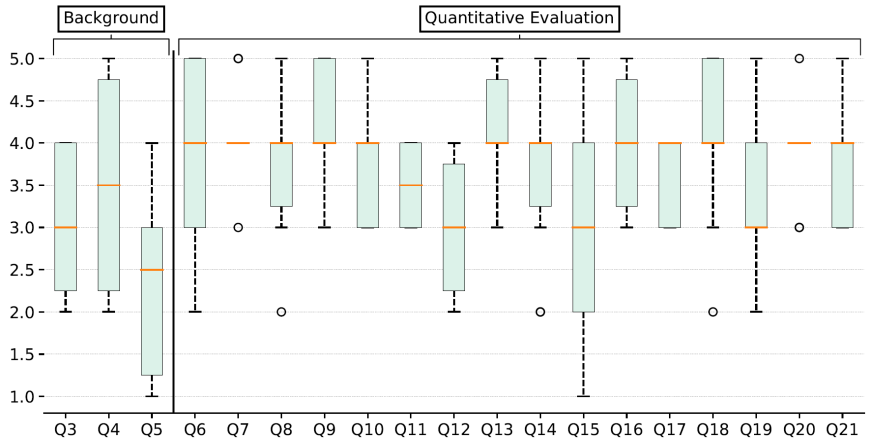

Figure 8: Boxplot showing distribution of user answers to the quantitative questions.

T3 *Correlating:* Analyze how the correlation between two attributes changes in the last 30 minutes, by also setting a RHS threshold upper bound to 3.

T4 *Exploring:* Consider two sets of discovered RFDs over Twitter data streams, find RFDs with a specific RHS attribute, occurring both sets, and graphically show common RFDs.

In particular, to rule out learning and tiring effects that otherwise may compound the execution of later tasks, we carried out a random assignment of tasks for each participant. After completing the assigned tasks, we retrieved feedbacks from participants by means of interviews. In fact, participants were requested to fill the questionnaire of Table 1 (Q6-Q25) to highlight the strengths and weaknesses of STRADYVAR. More specifically, the quantitative questions from Q6 to Q21 have been measured through a Likert scale ranging from 1 (Strongly disagree) to 5 (Strongly agree).

5.2. Results and Discussion

Figure 8 depicts the box plots derived from the answers of participants. A boxplot shows the median (horizontal lines), the interquartile ranges (boxes), the largest and the smallest observations (whiskers). Concerning the quantitative questions from Q6 to Q21 in the questionnaire, STRADYVAR obtained the general agreement of participants, while evaluating its usability and effec-

tiveness. In particular, according to answers for Q9, the analysis of attribute correlations has been considered the most simple task. Moreover, according to answers to Q18, the capability of analyzing differences among discovery results has been positively evaluated. STRADYVAR has turned out to be pleasant and informative according to answers to Q13 and Q14. However, different opinions have been provided on the algorithm execution transparency (see answers to Q15). The answers to Q15 appeared to be strongly influenced by the different outcomes expected from STRADYVAR by participants, probably due to their working/studying environments.

The open questions in the final questionnaire (see Q21-Q25 in Table 1) aimed at highlighting the strengths and weakness of STRADYVAR. In general, they revealed that some participants remarked the necessity to better integrate the different visual interfaces and features, in order to enable the analysis to be performed. Conversely, they welcomed the STRADYVAR's effectiveness in working with discovery results even when they change over time. Moreover, they positively judged the intuitiveness and the interaction capabilities of the STRADYVAR's visual components.

6. Conclusions and Future Work

Continuous profiling permits to consider extremely complex scenarios, where data are dynamically produced. In particular, each modification to data might produce the evolution of many FDs and/or RFDs. In some cases, due to both the high-frequency of updates for both FDs and RFDs, it is unthinkable to analyze the minimal FDs and RFDs for each time instant. Thus, to facilitate the interpretation of discovered RFDs over time, we have proposed the tool STRADYVAR, which relies on several visual components to let users actively analyze and explore the evolution of holding RFDs discovered through incremental discovery algorithms. Based on the results of the user study, in the future, we would like to integrate the several STRADYVAR's functionalities in order to enable users to simultaneously perform different types of analysis. Moreover, it would be useful

to enhance monitoring operations by allowing analysis over multiple executions of different algorithms.

References

[1] F. Naumann, Data profiling revisited, ACM SIGMOD Record 42 (4) (2014) 40–49.

[2] L. Caruccio, V. Deufemia, G. Polese, Relaxed functional dependencies – A survey of approaches, IEEE Transactions of Knowledge and Data Engineering 28 (1) (2016) 147–165.

[3] Z. Wei, S. Link, Discovery and ranking of functional dependencies, in: Proc. of IEEE International Conference on Data Engineering, ICDE '19, 2019, pp. 1526–1537.

[4] S. Kruse, F. Naumann, Efficient discovery of approximate dependencies, Proc. of VLDB Endowment 11 (7) (2018) 759–772.

[5] S. Song, L. Chen, Efficient discovery of similarity constraints for matching dependencies, Data & Knowledge Engineering 87 (2013) 146–166.

[6] L. Caruccio, V. Deufemia, G. Polese, Mining relaxed functional dependencies from data, Data Mining and Knowledge Discovery 34 (2) (2020) 443–477.

[7] P. Schirmer, T. Papenbrock, S. Kruse, D. Hempfing, T. Meyer, D. Neuschäfer-Rube, F. Naumann, DynFD: Functional dependency discovery in dynamic datasets, in: Proc. of International Conference on Extending Database Technology, EDBT '19, 2019, pp. 253–264.

[8] L. Caruccio, S. Cirillo, V. Deufemia, G. Polese, Incremental discovery of functional dependencies with a bit-vector algorithm, in: Proc. of Italian Symposium on Advanced Database Systems, SEBD '19, 2019, pp. 1–12.

[9] T. Papenbrock, T. Bergmann, M. Finke, J. Zwiener, F. Naumann, Data profiling with Metanome, Proc. of VLDB Endowment 8 (12) (2015) 1860–1863.

[10] S. Kruse, D. Hahn, M. Walter, F. Naumann, Metacrate: Organize and analyze millions of data profiles, in: Proc. of ACM Conference on Information and Knowledge Management, CIKM '17, 2017, pp. 2483–2486.

[11] L. Caruccio, V. Deufemia, G. Polese, Visualization of (multimedia) dependencies from big data, Multimedia Tools and Applications 78 (23) (2019) 33151–33167.

[12] M. F. De Oliveira, H. Levkowitz, From visual data exploration to visual data mining: a survey, IEEE Transactions on Visualization and Computer Graphics 9 (3) (2003) 378–394.

[13] Y. A. Sekhavat, O. Hoeber, Visualizing association rules using linked matrix, graph, and detail views, International Journal of Intelligence Science 3 (2013) 34–49.

[14] M. Hahsler, arulesViz: Interactive visualization of association rules with R, The R Journal 9 (2) (2017) 163.

[15] W. Chen, C. Xie, P. Shang, Q. Peng, Visual analysis of user-driven association rule mining, Journal of Visual Language and Computing 42 (2017) 76–85.

[16] K. Techapichetvanich, A. Datta, VisAR: A new technique for visualizing mined association rules, in: Proc. of International Conference on Advanced Data Mining and Applications, ADMA '05, Springer, 2005, pp. 88–95.

[17] C.-W. Cheng, Y. Sha, M. D. Wang, Intervisar: An interactive visualization for association rule search, in: Proc. of ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB '16, 2016, pp. 175–184.

[18] H.-H. Ong, K.-L. Ong, W.-K. Ng, E. P. Lim, CrystalClear: Active visualization of association rules, in: Proc. Workshop on Active Mining, 2002, pp. 1–6.

[19] T. Bleifuß, L. Bornemann, T. Johnson, D. V. Kalashnikov, F. Naumann, D. Srivastava, Exploring change: A new dimension of data analytics, Proc. of VLDB Endowment 12 (2) (2018) 85–98.

[20] D. A. Keim, J. Schneidewind, M. Sips, Circleview: A new approach for visualizing time-related multidimensional data sets, in: Proc. of Working Conference on Advanced Visual Interfaces, AVI '04, 2004, pp. 179–182.

[21] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, R. C. Miller, Processing and visualizing the data in tweets, ACM SIGMOD Record 40 (4) (2012) 21–27.

[22] V. Segura, S. D. Barbosa, Historyviewer: Instrumenting a visual analytics application to support revisiting a session of interactive data analysis, Proc. of the ACM on Human-Computer Interaction (2017) 11.

[23] A. K. Elmagarmid, P. G. Ipeirotis, V. S. Verykios, Duplicate record detection: A survey, IEEE Transactions on Knowledge and Data Engineering 19 (1) (2007) 1–16.

[24] J. Kivinen, H. Mannila, Approximate inference of functional dependencies from relations, Theoretical Computer Science 149 (1) (1995) 129–149.

[25] L. Caruccio, V. Deufemia, G. Polese, Discovering relaxed functional dependencies based on multi-attribute dominance, IEEE Transactions on Knowledge and Data Engineering (2020) To appear.