Tabular Data Anomaly Patterns

Dina Sukhobok, Nikolay Nikolov, and Dumitru Roman
SINTEF
Forskningsveien 1a, 0373 Oslo, Norway
Email:{fristname.lastname}@sintef.no

***Abstract*—One essential and challenging task in data science is data cleaning – the process of identifying and eliminating data anomalies. Different data types, data domains, data acquisition methods, and final purposes of data cleaning have resulted in different approaches in defining data anomalies in the literature. This paper proposes and describes a set of basic data anomalies in the form of anomaly patterns commonly encountered in tabular data, independently of the data domain, data acquisition technique, or the purpose of data cleaning. This set of anomalies can serve as a valuable basis for developing and enhancing software products that provide general-purpose data cleaning facilities and can provide a basis for comparing different tools aimed to support tabular data cleaning capabilities. Furthermore, this paper introduces a set of corresponding data operations suitable for addressing the identified anomaly patterns and introduces Grafterizer – a software framework that implements those data operations.**

***Index Terms*—data anomalies, data cleaning, tabular data, data cleaning and transformation operations**

1. Introduction

The research literature in the field of data science very often assumes that the input data that serves various data analytics processes is already in a clean and correct state for data consumption [1]. However, an essential part of time and effort of data workers is concentrated on cleaning data of problems in their values or structure. Recent surveys show that the time spent on data cleaning constitutes up to 60% of data scientists' tasks [2].

According to [3], data is a set of discrete objective facts describing a set of items (events, objects). In other words, data are the quantitative or qualitative values for the attributes or characteristics of an item. The most commonly used classification of data distinguishes three types of data, depending on how the data are organized:

- *Structured data* are represented mostly by tables in relational databases, and are characterized by strict adherence of data to the associated schema.
- *Semi-structured data* may have some certain structure, but are not organized as strictly as structured data. Examples of semi-structured data are simple tabular formats (e.g., CSV), XML and JSON documents. Data stored in NoSQL databases are also considered as semi-structured data.
- *Unstructured data* don't have any structural organization. Typical examples are text documents or multimedia content [4].

Some authors propose narrower definition of data. In [5], the authors distinguish between data and information, where information is defined as a broader concept that includes all the organizational types mentioned above, and data refers only to structured data in a database. In this way data quality and information quality differ significantly in their scope.

Semi-structured data, particularly in tabular format, are well-known and widely used by data scientists. For example, an analysis of Open Data portals shows that most of the data openly available on the Web are in tabular format [6]. In this paper we investigate data cleaning as a way to increase the quality of tabular data.

When a dataset does not satisfy a given data quality criteria, it means that it contains *data anomalies*. In order to provide higher data quality, data anomalies have be first detected and then addressed. The state of the art research literature includes several taxonomies of data anomalies [7], [8], [9], [10], [11], [12], [13], [14], however, there is currently no universal and agreed upon definition of data anomalies. One of the reasons for this is the large number of possible data quality violations due to the variety of data types and domains. Thus, [11] distinguishes between data cleaning tasks for quantitative data, categorical data, postal addresses and identifiers. Several taxonomies are based on sources of data anomalies [7], [12]. Some authors investigate data anomalies in the context of data integration [8], [15], whereas other authors describe only anomalies in single-source data. Our strategy in this paper in defining a set of data anomalies is more practical. We aim to provide an insight into basic data anomalies independent of the data domain, data acquisition technique, or the purpose of data cleaning. Such a set can serve as a central reference for basic data anomalies, can be used for developing and enhancing software products that provide general-purpose data cleaning facilities and can make it easier to compare different tools incorporating data cleaning capabilities.

To compile the set we make use of the notion of *pattern*. A pattern can be defined as "the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts" [16]. The data anomaly patterns collected and

978-1-5386-0960-6/17 $31.00 © 2017 IEEE
DOI 10.1109/Innovate-Data.2017.10

categorized in this paper are defined through the scope of the anomaly, conditions under which the anomaly can occur, a description of the problem associated with each data anomaly, and example dataset containing the anomaly. Furthermore, based on the identified patterns, we propose a set of data cleaning and transformation operations suitable for addressing the identified data anomalies and introduce Grafterizer - an interactive tool and reference implementation for the proposed data cleaning and transformation operations.

The main contributions of this paper include:

- Collecting, categorizing, and documenting basic tabular data anomalies as patterns, to serve as a reference resource for tabular data anomalies;
- A set of corresponding tabular data cleaning and transformation operations to address tabular data anomalies, to serve as a reference resource for tabular data cleaning operations; and
- A prototype implementation of tabular data cleaning and transformation operations, to demonstrate the practical feasibility of the proposed tabular data cleaning and transformation operations.

The rest of this paper is structured as follows. Section 2 describes data cleaning as a process. Section 3 describes the tabular data anomaly patterns. Section 4 introduces the data cleaning and transformation operations as a way to address the basic tabular anomalies and presents Grafterizer — our reference implementation for the proposed data cleaning and transformation operations. Finally, Section 5 presents relevant related works on data cleaning and transformation, and Section 6 summarizes the paper.

2. Data Cleaning and Transformation as a Process

Comprehensive data cleaning consists of four general phases [10] and is iterative by nature. The phases of data cleaning are:

1) *Data auditing* aimed to detect data anomalies;
2) *Definition of a transformation workflow* aimed to suggest a way to remove detected data anomalies;
3) *Execution of a transformation workflow* aimed to apply suggested transformation to data;
4) *Verification of the executed transformation* aimed to evaluate the data cleaning results.

This process is cyclic and has to be repeated until data quality satisfies desired criteria.

The first phase in a typical data cleaning process is data auditing. During this phase, the data anomalies in a dataset are detected. Some of the existing software products for data cleaning have data anomalies detection capabilities. Nevertheless, user participation cannot be completely excluded and human judgment is crucially important in the process of evaluating identified data anomalies and for choosing an appropriate method to address them [17]. When automated data auditing is not supported by the data cleaning system or it cannot help with detecting data anomalies, data workers need to inspect the data manually. In this scenario, an important role is played by appropriate data ordering [18]. Changing a way of data ordering can make it easier to scan data values in order to identify anomalies. The output of the first phase of data cleaning process is a list of data anomalies residing in the audited dataset.

During the second phase—the definition of a transformation workflow—the data worker specifies data cleaning and transformation operations to be performed on the data in order to eliminate data anomalies, enrich the data, or transform it into a form more suitable for further audit. The result of the definition of a transformation workflow phase is a sequence of data cleaning and transformation operations.

The third phase, the transformation workflow execution, implies the execution of the constructed workflow. The output of this phase is data transformed in accordance with the data cleaning and transformation operations specified in the previous step.

The last phase of data cleaning and transformation cycle is verification of executed transformation. During this phase, the data worker verifies whether the anomalies detected during the first phase have been removed. Although some authors such as [19], [20] do not distinguish verification as a separate step of data cleaning and transformation, this phase is very important since it gives an evaluation of the performed transformation and determines a plan for further action. After the verification of the performed transformation, the transformed data may be suitable for further usage and the cleaning and transformation cycle is completed. In other cases, data may still contain anomalies or require further enrichment, and the cycle of data cleaning and transformation starts again from the phase of data auditing.

Thus, phase one (data auditing) requires knowledge of what data anomalies can affect data quality in the particular data cleaning task. Phase two (definition of the transformation workflow) requires knowledge about data cleaning and transformation operations. Phase four (the verification of the transformation workflow) requires the data worker to evaluate the effect of data cleaning and transformation operations on the data anomalies. Consequently, all the phases of data cleaning that expect data worker's involvement make use of the set of data anomalies and the set of corresponding data cleaning and transformation operations.

3. Tabular Data Anomaly Patterns

Despite a great number of publications on data anomalies, there is currently no basic set of data anomalies that is being used as a central reference, for example to compare different tools incorporating data cleaning capabilities. The taxonomies of data anomalies described in literature differ in terminology, coverage and ways of mapping data anomalies to the affected data quality dimensions [14], [21]. Different authors use also various names for data anomalies, such as data quality problems [8], discrepancies [19], data distortions [7], data errors [20], data defects [14], or just problems with data.

TABLE 1: List of data anomaly patterns for tabular data

N	Scope	Anomaly pattern	Description
1	Cell values	Illegal values	Values outside of domain range
2		Inconsistent values	Syntactically correct but contradicting with other attribute values
3		Missing values	Column values are not present
4	Column headers	Column headers containing attribute values	Column headers are attribute values themselves, not attribute names
5		Incorrect column headers	Column headers are inconsistent with the attribute they hold
6	Column headers and cell values	Columns not related to data model	Column headers are inconsistent with the attribute they hold
7		Multiple values stored in one column	Several attribute values of the data model are stored in one column
8		Single value split across multiple columns	One attribute value of the data model stored across several columns
9	Rows	Rows not related to data model	Records in the source dataset describe unrelated entities
10		Duplicate rows	The same entity (having the same primary key according to the data model) is described more than once in the dataset

Due to the inevitable diversity in source dataset quality levels and the diversity of the purposes of data cleaning, identifying and agreeing upon all data anomalies is unrealistic. However, it is possible to identify patterns of the most common data anomalies.

In order to obtain the list of possible data anomalies, we investigated the research literature on data quality, statistics, and data cleaning, including literature containing interview results with data workers [22] and literature using pure logical reasoning to describe data quality issues [7], [8], [9], [10], [11], [12]. We narrowed the scope of our investigation to anomalies related to single-source tabular data that are the most commonly encountered in tabular data. The most common anomalies were identified during doing practical work on data cleaning. In particular, the reference implementation introduced in this paper (see Section 4.2) was used for data cleaning and transformation in various data-driven large-scale projects, e.g. proDataMarket [23], SmartOpenData [24], DaPaaS [25], amongst others.

The proposed set of data anomalies is classified based on the scope of a data anomaly in a tabular dataset. Tabular datasets are composed of *rows* and *columns* [9]. Columns in tabular data are almost always labeled with *column headers*. The values on the intersection of rows and columns are **cell values**. According to the proposed classification, data anomalies can occur in cell values (attribute values for particular records), column headers (attribute names), in cell values and column headers at the same time or in rows (entire records).

Data tables are intended to represent some part of the real world and each row in a table should be mapped to objects of a real world. Therefore, any high-quality tabular dataset should follow these rules:

1) Each row represents an **entity**, which can be, for example, a person, place, physical object or an event. Entities have a unique existence in the real world.
2) Each column header represents an entity **attribute**.
3) Each column value represents a **value** of the corresponding attribute of an entity.
4) Each table represents a **collection** of entities.
5) All entities in a collection have the same **entity type**.

Deviations from these rules can indicate a data anomaly.

A summary of resulting data anomalies (identified with the help of above mentioned analysis and practical work) is provided in Table 1.

Data anomalies may concern either just data itself or data with respect to a given data model, which is closely connected with further data consumption. Thus, anomalies 6 - 10 in Table 1 concern data with respect to the data model.

The sources of problems with the data may differ. The most common reason of erroneous data is human errors during the manual production of the data and data acquisition as merging from multiple sources [15], [26]. Another source for data quality issues is data schema evolution over time, which can cause misinterpretation of new entity types or attributes. Finally, automated data generation, such as information derived from sensors, carries its own issues, such as errors due to the inferences or wrong calibration [18].

In this paper we focus on the effect of data anomalies on data quality rather then their origin, and the identified set of data anomalies is independent of the anomaly source.

In the following, we provide details on each pattern.

Anomaly pattern #1: Illegal values

- ***Scope:*** Column cells.
- ***Conditions:*** The attributes in a dataset have certain constraints.
- ***Description:*** Values in column cells are outside of domain range. Domain range here should be understood as the set of possible values for the attribute (interval, set of distinct values, pattern). The anomaly occurs when the cell values violates the constraints of the attribute domain. The most common examples are mismatch of the attribute data type and misspellings (as values outside the dictionary domain).
- ***Examples:*** See Tables 2 and 3.

TABLE 2: Illegal date value

Order ID	Date of order
17004	28.02.2016
17005	30.02.2016
17006	01.03.2016

TABLE 3: Illegal integer value

Department	Number of employees
Purchasing	7
Sales	15
IT	6.5

Anomaly pattern #2: Inconsistent values

- ***Scope:*** Cell values.
- ***Conditions:*** A functional dependency between attributes in dataset exists.
- ***Description:*** Syntactically correct but contradicting with other attribute values. This anomaly takes place when the cell values violate functional dependencies on other attributes within a record.
- ***Example:*** See Table 4.

TABLE 4: Inconsistent values

Municipality	Occupational class	Employed men	Employed women	Total
Breda	Communications	482	49	531
Breda	Credit and banking	35	1	37
Breda	Insurance	32	2	34

Anomaly pattern #3: Missing values

- ***Scope:*** Cell values.
- ***Conditions:*** The ”not-null” constraint for the attributes in dataset exists.
- ***Description:*** Cell values are not present.
- ***Example:*** See table 5.

TABLE 5: Missing values

Municipality	Occupational class	Employed men	Employed women	Total
Breda	Communications	482	49	531
Breda	Credit and banking	37		37
Breda	Insurance	32	2	34

Anomaly pattern #4: Column headers containing attribute values

- ***Scope:*** Column headers.
- ***Conditions:*** Several column headers in a dataset characterize the same data attribute.
- ***Description:*** Column headers are the attribute values themselves, not the names of the attributes.
- ***Example:*** See Table 6.

Anomaly pattern #5: Incorrect column headers

- ***Scope:*** Column headers.
- ***Conditions:*** *No special conditions.*
- ***Description:*** Column headers are inconsistent with the actual attribute they hold.
- ***Scope:*** Column headers.

TABLE 6: Column headers containing attribute values

Municipality	Occupational class	2011	2012	2013
Halden	Service and sales workers	3432	3367	3335
Halden	Skilled agricultural, forestry and fishery workers	210	228	213
Halden	Plant and machine operators and assemblers	1744	1700	1694

- ***Example:*** See Table 7.

TABLE 7: Incorrect column headers

Occupational class	Municipality	Employed men	Employed women	Total
Breda	Communications	482	49	531
Breda	Credit and banking	35	2	37
Breda	Insurance	32	2	34

Anomaly pattern #6: Columns not related to data model

- ***Scope:*** Column headers and cell values.
- ***Conditions:*** The dataset should confirm to a certain data model.
- ***Description:*** Dataset describes attributes not relevant in scope of the given data model.
- ***Example:*** See Table 8.
- ***Data model:*** See Table 9.

TABLE 8: Columns not related to data model

Name	Social security number	Address	Hobbies
Alice Smith	1234567	New York, Harrison Street, 507	Skiing
Bob Johnson	2345678	Richmond, Main Street, 17	Chess

TABLE 9: Data model example

Customer
Name Social security number Address

Anomaly pattern #7: Rows not related to data model

- ***Scope:*** Rows.
- ***Conditions:*** The dataset should confirm to a certain data model, where certain inclusion constraints exist.
- ***Description:*** The records, contained in the source dataset describe entities not related to the data model. This anomaly takes place when the records in a dataset violate the constraints of the record inclusion.
- ***Example:*** See Table 10.
- ***Data model:*** See Table 11.

TABLE 10: Rows not related to the data model

Owner name	Owner Id number	Property Id number	Owned from	Owned until
Alice Smith	1234567	124931510	03.05.2004	03.05.2024
Bob Johnson	2345678	124931511	10.10.2003	01.01.2017
MyCompany, Inc.	971032081	124931526	21.01.2008	01.01.2018

TABLE 11: Data model example

Ownership [Inclusion constraint: physical persons]
Owner name
Owner Id number
Property Id number
Owned from
Owned until

Anomaly pattern #8: Multiple values stored in one column

- ***Scope:*** Column headers and cell values.
- ***Conditions:*** The dataset should confirm to a certain data model.
- ***Description:*** The values, described by several attributes of the data model are stored in one column.
- ***Example:*** See Table 12.

TABLE 12: Multiple values stored in one column

Name	Social security number	Address
Alice Smith	1234567	New York, Harrison Street, 507
Bob Johnson	2345678	Richmond, Main Street, 17

- ***Data model:*** See Table 13.

TABLE 13: Data model example

Customer
Name
Social security number
City
Street address

Anomaly pattern #9: Single value split across multiple columns

- ***Scope:*** Column headers and cell values.
- ***Conditions:*** The dataset should confirm to a certain data model.
- ***Description:*** The values, described by one attribute of the data model are stored across several column.
- ***Example:*** See Table 14.

TABLE 14: Single value split across multiple columns

Name	Social security number	City	Street address
Alice Smith	1234567	New York	Harrison Street, 507
Bob Johnson	2345678	Richmond	Main Street, 17

- ***Data model:*** See Table 15.

TABLE 15: Data model example

Customer
Name
Social security number
Full address

Anomaly pattern #10: Duplicate rows

- ***Scope:*** Rows.
- ***Conditions:*** The dataset should confirm to a certain data model where a primary key (one-attribute or composite) is assigned.
- ***Description:*** The same entity (having the same primary key according to the data model) is described more than once in the dataset.
- ***Example:*** See Table 16.

TABLE 16: Duplicate rows

Name	Social security number	City	Street address
Alice Smith	1234567	New York	Harrison Street, 507
Bob Johnson	2345678	Richmond	Main Street, 17
Mary Williams	2345678	New York	East 52nd Street, 55

- ***Data model:*** See Table 17.

TABLE 17: Data model example

Customer	
PK	**Social security number**
	Name
	City
	Address

4. Addressing Data Anomalies

The list of tabular data anomaly patterns presented in Section 3 can be used as a basis for developing methods and practical solutions for data cleaning. As discussed in Section 2, in order to eliminate data anomalies, the data cleaning and transformation operations should be executed.

We define the actions needed to detect or remove data anomalies as *data cleaning operations*. Very often, preparing data for further usage includes both identifying and removing data anomalies, and data enrichment with transformed or computed values to facilitate further consumption. Therefore in this paper we introduce *data cleaning and transformation operations* rather than just data cleaning operations.

In this section, we propose a set of data cleaning and transformation operations suitable to address tabular data anomaly patterns introduced above. In addition, we also introduce a reference implementation of data cleaning and transformation operations.

4.1. Data Cleaning and Transformation Operations

In this paper we consider data cleaning and transformation operations as atomic units of work performed on a dataset to convert the data to the more usable form.

The data cleaning and transformation operations can affect data in one of following ways:

- *Data reordering* operations do not modify cell values, number of entities or their attributes. Instead, they are used to discover data anomalies in data auditing phase.
- *Data editing* operations modify cell values and are used to resolve data anomalies.
- *Data extraction* operations do not modify cell values, but change the number of entities or their attributes and are often used to resolve data anomalies.
- *Data enrichment* operations enrich the data with new values (e.g., compute values, provide summaries).

One of the difficulties of performing data cleaning with the set of pre-defined data cleaning operators is that the more data quality problems the input dataset has, the more difficult is to clean data with the limited by the set of data cleaning and transformation operations [27]. Therefore, one of the key requirements to the set of data cleaning and transformation operations is to make them adjustable to the variety of data quality problems. Data cleaning and transformation operations can be used for detecting data anomalies, eliminating them or enriching the data. The relationship between data cleaning and transformation operations and their function is given in Table 18.

The requirements to the data cleaning and transformation operations are as follows:

1) **Completeness**: the set of operators should be complete and address all identified tabular data anomalies;
2) **Clear logical specification**: data cleaning operators are easy to understand and use;
3) **Independence of implementation**: data cleaning operators are independent of their technical implementation;
4) **Flexibility**: operators have adjustable parameters, thus making it possible to extend the number of data quality problems that can be resolved with the help of each operator. As a special case of such parameter, operators may make use of user-defined utility functions, containing data-dependent treatment of data quality problems.

The proposed operations are parameterized dataset-to-dataset functions. One special case is a ”Derive column” operation. This operation takes as a parameter a value(array)-to-value function and provides a great degree of flexibility to the transformations that can be performed on a dataset.

An overview of the resulting set of data cleaning operators is provided in Table 19.

4.2. Reference Implementation: Grafterizer

The resulted set of data cleaning and transformation operations was implemented in Grafterizer [28], [36] – a web-based framework for data cleaning and transformations, that is a part of DataGraft[1] [29], [30] platform for data transformation and publishing. This section describes both the library for data cleaning and transformation operations (targeting software developers to facilitate data cleaning tasks in their applications) and the graphical user interface of Grafterizer that facilitates interactive data cleaning for data workers.

[1]DataGraft is accessible at https://datagraft.net.

TABLE 18: The relationship between data anomaly patterns and data cleaning operations

Operation/Anomaly pattern	Illegal values	Inconsistent column values	Missing values	Column headers containing attribute values	Incorrect column headers	Columns not related to data model	Rows not related to data model	Multiple values stored in one column	Single value split across multiple columns	Duplicate rows	**Anomaly detection**	**Dataset enrichment**
Add rows												●
Take rows							●					
Delete rows							●					
Shift row											●	
Filter							●				●	
Deduplicate										●		
Sort											●	
Reshape (Melt)				●								
Reshape (Cast)												●
Group and Aggregate												●
Take columns						●						
Add columns												●
Derive columns	●	●	●									●
Shift columns											●	
Merge columns									●			
Split columns								●				
Rename columns					●							

TABLE 19: List of data cleaning and transformation operations

Effect	Operation	Description
Data reordering	Shift Row	Change rows order number
	Shift Column	Change columns order number
	Sort	Sort dataset by given column name(s) in given order
Data editing	Reshape (Melt)	Move columns to rows
	Merge Columns	Merge several columns in one using specified separator between them
	Split Columns	Split specified column into multiple by separator
	Rename Columns	Change column headers
Data extraction	Take Rows	Take a subset of rows by their order numbers
	Delete Rows	Delete a subset of rows by their order numbers
	Filter	Take a subset of rows by value or condition
	Deduplicate	Remove duplicate rows based on the specified simple or composite primary key
	Take subset of columns	Take a subset of columns by specifying either names or indexes of columns to extract or names or indexes of columns to exclude
Data enrichment	Add new row	Add new row to dataset
	Reshape (Cast)	Move rows to columns
	Group and Aggregate	Group dataset by given column (set of columns) and create specified aggregations on other columns
	Add column	Add new column with manually specified value
	Derive column	Add new column with calculated value

Data Cleaning and Transformation Operations Library. The basis for tabular transformations in Grafterizer is Grafter[2] – a powerful software library and DSL for producing linked data graphs from tabular data, which provides extensive support for tabular-to-tabular data conversions and powerful ETL data transformations, suitable for handling large datasets.

The Grafter suite of tools is implemented in Clojure[3] – a functional programming language and Lisp dialect that runs on the Java Virtual Machine (JVM). The use of the JVM allows Clojure to have access to the numerous libraries, available for JVM-based languages. Using a functional programming language, such as Clojure, also provides benefits in terms of data processing including:

- Functional programs typically operate on *immutable data structures*. Since the data structures cannot be modified, they can be shared without the need to ensure concurrency, which allows more efficient memory use and therefore allows to process bigger amounts of data.
- Most of functional languages, including Clojure, support *lazy evaluation*. Lazy evaluation implies deferring the computation of values until they are needed, which helps to avoid unnecessary computations and allows to use infinite data structures.
- Functional languages use *higher-order functions*. This means ability to process code as data and improves program modularity.

The tabular data transformation process in Grafter is realized through a pipeline abstraction, i.e., each step of a transformation is defined as a **pipe** - a function that performs simple data conversion on its input and produces an output. These functions are composed together in a pipeline, whereby the output of each pipe serves as input to the next.

Pipeline functions may be combined arbitrarily, whereby each combination produces another pipeline. Each of these pipes is a pure Clojure function from a *Dataset* to a *Dataset*, where *Dataset* is a data structure used to handle data in Grafter. The example can be seen in Listing 1:

```
1 -> (read-dataset data-file)
2    (drop-rows 1)
3    (make-dataset move-first-row-to-header)
4    (mapc {:gender {"f" (s "female")
5                    "m" (s "male")}}))
```

Listing 1: Example of Grafter's pipeline

Line (1) reads the tabular data and produces a *Dataset*, and, using the thread-first macro, denoted by ”->”, takes this initial value as its first argument, and threads it through the following expressions, thus constituting a pipeline. Each following line from the example above is a function call that takes a *Dataset* and, optionally, other parameters as an input, and returns a modified *Dataset*.

The functions' uniformity in terms of input and output makes it very intuitive for users to use pipelines for data manipulation.

The Grafter library provides a set of useful functions for processing tabular data[4], to large extent overlapping the data cleaning and transformation operations described in this paper. However, some of the operations are not available or were implemented with the different specification from the one identified in Table 19. For this reason we developed a subset of routines as an extension to Grafter. These routines are accessible at Clojars[5] – a public repository for open-source Clojure libraries.

[2]http://grafter.org
[3]https://clojure.org

[4]http://api.grafter.org/docs/0.6.0/grafter.tabular.html
[5]https://clojars.org/grafterizer/tabular_functions

The Grafterizer Graphical User Interface. Grafterizer provides an interactive user interface with end-to-end support for data cleaning and transformation based on the Grafter library. The resulting framework supports building data transformations in an easy and interactive way. Tabular-to-tabular transformations use data cleaning and transformation operations described in Section 4.2 and are specified in a visual presentation of a pipeline, clearly separating the order of data cleaning and transformation operations, used to process the data (Figure 1). The basic functionalities are:

- *Defining and editing data cleaning and transformation operations* – data cleaning and transformation operations can be easily added, edited, reordered or removed. All operations are defined with editable parameters (See Figure 2).
- *Live preview* – the framework interactively displays the transformed dataset, thus supporting a real-time evaluation of a defined transformation workflow. Live preview also supports error reporting by notifying users about errors during transformation execution in a pop-up window.
- *Sharing and reusing the transformation workflows* – the feature is supported by the integration with the DataGraft platform. Transformations created with the help of Grafterizer are stored as a sequence of operations on data and can be reused and copied.

The Grafterizer framework was successfully used to perform data cleaning and transformation in different domains. Notable examples include datasets from the biodiversity and environment protection domain in the SmartOpenData[6] project [24], infrastructure components and natural hazards domain in the InfraRisk [7] project [31] and property related datasets from the proDataMarket[8] project [23].

5. Discussions and Related Work

The problem of data cleaning is well-known and is covered by a number of publications in the state-of-the-art research literature. In particular, many publications study the data anomalies taxonomies [7], [8], [9], [10], [11], [12], [13], [14]. The set of data anomalies discussed in this paper covers tabular data anomalies and represents a superset of tabular data anomalies described in data cleaning literature. Furthermore, we categorize and document the data anomalies as patters, and use the patterns to propose a set of operations to address the anomalies, which we then implemented in a prototype.

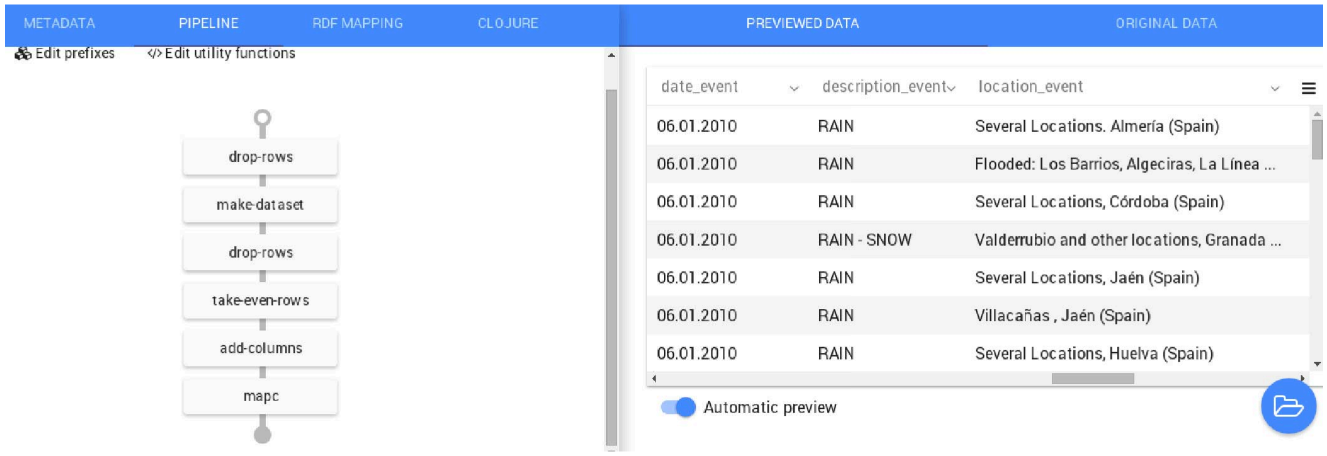
At the same time, a number of practical solutions were developed in order to facilitate data cleaning. Currently available software products for data cleaning and transformation for tabular data can be divided into several groups as follows:

- *Spreadsheet software* still remains ubiquitous among data workers [32]. This is mainly because spreadsheets are a well-known abstraction to most data workers, have a simple intuitive interface, and require no advanced technical skills for their usage. Examples of spreadsheet tools that can be used for tabular data cleaning are *Libre Office Calc*[9], *Microsoft Excel*[10], *Google Spreadsheet*[11] and many others. However, despite their simplicity and interactive design, spreadsheet software products have a number of limitations and disadvantages. Firstly, working with spreadsheets is error-prone. Perhaps, the one most well-known error made during spreadsheet data transformation occurred in Reinhart and Rogoff's austerity-justifying paper [33]. Two Harvard economists published a highly influential piece of work, which contained a wrong conclusion due to an erroneous Excel spreadsheet formula. Transformation workflow definition errors in spreadsheets are rather difficult to identify – data and transformation code are mixed together, significantly hindering the process of code review. Furthermore, conventional spreadsheets are typically limited in functionality and so are incapable of coping with the most sophisticated data quality problems. One more substantial disadvantage of spreadsheet tools is that they are not suitable for processing large amounts of data.
- *Command line interface (CLI) tools* are typically reliable, provide a broad set of functionalities, give the ability to automate data cleaning and conversion, and allow to make this task repeatable. Examples of command-line tools are *csvkit*[12], *CSVfix*[13] and others. Although the tools from this group provide good functional coverage for data cleaning and are able to handle large volumes of input data, they suffer from the lack of convenient user interface.
- *Programming languages and libraries for statistical data analysis* include, for example, *Agate*[14] Python library for data analysis, *R*[15] programming language for statistical computing and the data manipulation tools based on this language, e.g., *dplyr*[16] and *tidyr*[17]. The disadvantage of the tools from this group is that they require users to have considerable knowledge in programming.
- *Complex systems designed to be used for interactive data cleaning and transformation in ETL process*. Examples of relevant complex systems supporting data cleaning as a part of ETL process include *Pentaho Data Integration*[18], *Trifacta Wrangler*[19] and *OpenRefine*[20]. These systems are designed specifically to support an ETL process and offer a number of useful data manipulation functionalities.

Several factors should be considered when choosing the right tool for data cleaning. The general-purpose functional-

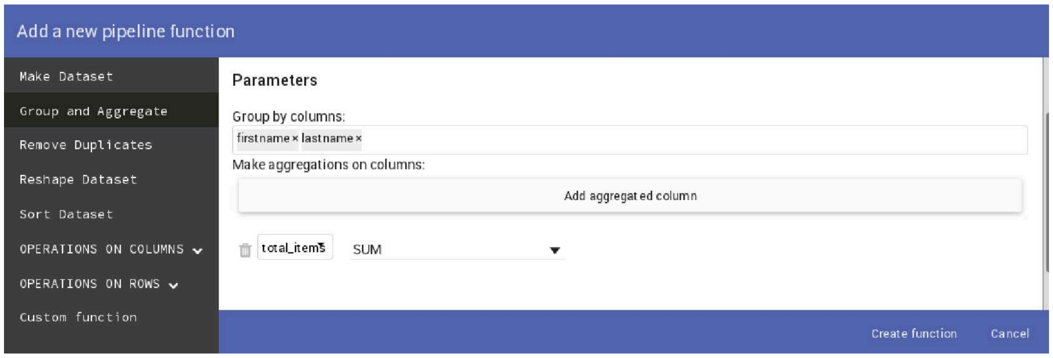
[6]http://www.smartopendata.eu
[7]http://www.infrarisk-fp7.eu
[8]http://blog.prodatamarket.eu

[9]http://www.libreoffice.org/discover/calc
[10]https://products.office.com/en/excel
[11]https://www.google.com/sheets/about
[12]https://csvkit.readthedocs.io/en/1.0.1
[13]http://neilb.bitbucket.org/csvfix/manual/csvfix16/csvfix.html
[14]https://agate.readthedocs.org/en/1.3.1
[15]https://www.r-project.org/about.html
[16]https://cran.r-project.org/web/packages/dplyr/index.html
[17]https://blog.rstudio.org/2014/07/22/introducing-tidyr
[18]http://community.pentaho.com/projects/data-integration
[19]https://www.trifacta.com/products/wrangler
[20]http://openrefine.org



date_event	description_event	location_event
06.01.2010	RAIN	Several Locations. Almería (Spain)
06.01.2010	RAIN	Flooded: Los Barrios, Algeciras, La Línea ...
06.01.2010	RAIN	Several Locations, Córdoba (Spain)
06.01.2010	RAIN - SNOW	Valderrubio and other locations, Granada ...
06.01.2010	RAIN	Several Locations, Jaén (Spain)
06.01.2010	RAIN	Villacañas , Jaén (Spain)
06.01.2010	RAIN	Several Locations, Huelva (Spain)

Figure 1: Overview of Grafterizer's interface



Group by columns: firstname × lastname ×

Make aggregations on columns:

total_items SUM

Figure 2: Overview of Grafterizer's pipeline function

ity of the tool can be assessed by testing whether the tool can be used to eliminate the basic data anomalies. The flexibility of the tool defines how easy is it to eliminate more specific data anomalies and perform complex data transformations. Programming languages and systems providing a possibility to use custom programming code provide the most flexibility, whereas command-line interface tools, spreadsheets and some ETL tools are less flexible. Another factor that plays an important role in choosing the tool for data cleaning is ease of use. Given the fact that most of the data workers are domain experts rather than IT-specialists [22], data cleaning tools should be easy to use and understand. The ease of use is inherent in spreadsheets and GUI ETL tools, while programming languages require users to have considerable knowledge in programming. The Grafterizer framework described in Section 4.2 was designed to provide flexibility of programming languages and at the same time keep ease of use. An evaluation of the usability and ease of use has been performed in [35].

6. Summary and Outlook

A simple analysis of a typical data cleaning and transformation process indicates that most of the phases of such a process require knowledge about data anomalies, and mechanisms to address the anomalies. Thereby, this paper focuses on tabular data anomalies. Although some of the basic data anomalies appearing in tabular data are known in the literature, no unified and generally agreed upon approach in defining tabular data anomalies exists, making it difficult to compare capabilities of existing data cleaning solutions and identify possible extensions to make them more comprehensive. In this paper we collected, categorized, and documented basic tabular data anomalies as patterns in a uniform way, independent of the data domain, the way of data acquisition or the purpose of data cleaning, with the aim of creating a reference resource for tabular data anomalies. The proposed set of data anomalies patterns can be used as a reference resource for tabular data anomalies and can be of a great value for developing and enhancing tools aimed to support general-purpose tabular data cleaning capabilities. As a plan of future work we plan to do a thorough comparison of software products for tabular data cleaning and transformation based on the data anomaly patterns.

Furthermore, we proposed a set of corresponding data cleaning and transformation operations suitable for addressing tabular data anomalies, which can be used as a reference resource for tabular data cleaning operations. The practical feasibility of the the proposed data cleaning operations was shown by the Grafterizer prototype — a data cleaning framework implementing the proposed data cleaning and transformation operations. As part of future work, we plan to experiment with various Big Data back-ends for Grafterizer to create a scalable solution for data cleaning. Furthermore, to improve the user experience and usability of Grafterizer, we plan to integrate aspects of visual data profiling and predictive interactions as part of the process of data cleaning and transformation, which have been demonstrated and evaluated in [35]. Finally, we plan to introduce spreadsheet-like interactivity by allowing for direct manipulation of data in the tabular preview.

Acknowledgements. This work is partly funded by the EC H2020 projects proDataMarket (Grant number: 644497), euBusinessGraph (Grant number: 732003), and EW-Shopp (Grant number: 732590).

References

[1] de Jonge, Edwin, and Mark van der Loo. "An introduction to data cleaning with R." Statistics Netherlands, The Hauge (2013).

[2] CrowdFlower 2016 Data Science Report. Available at http://visit.crowdflower.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf.

[3] Davenport, Thomas H., and Laurence Prusak. Working knowledge: How organizations manage what they know. Harvard Business Press, 1998.

[4] Elmasri, Ramez. Fundamentals of database systems. Pearson Education India, 2008.

[5] Batini, Carlo, and Monica Scannapieco. Data and Information Quality: Dimensions, Principles and Techniques. Springer, 2016.

[6] Umbrich, Jürgen, Sebastian Neumaier, and Axel Polleres. "Quality Assessment and Evolution of Open Data Portals." Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on. IEEE, 2015.

[7] De Veaux, Richard D., and David J. Hand. "How to lie with bad data." Statistical Science (2005): 231-238.

[8] Rahm, Erhard, and Hong Hai Do. "Data cleaning: Problems and current approaches." IEEE Data Eng. Bull. 23.4 (2000): 3-13.

[9] Wickham, Hadley. "Tidy data." Journal of Statistical Software 59.10 (2014): 1-23. Available at http://www.jstatsoft.org/v59/i10/.

[10] Müller, Heiko, and Johann-Christph Freytag. Problems, methods, and challenges in comprehensive data cleansing. Professoren des Inst. Fr Informatik, 2005.

[11] Hellerstein, Joseph M. "Quantitative data cleaning for large databases." United Nations Economic Commission for Europe (UNECE) (2008).

[12] Kim, Won, et al. "A taxonomy of dirty data." Data mining and knowledge discovery 7.1 (2003): 81-99.

[13] Abedjan, Ziawasch, et al. "Detecting Data Errors: Where are we and what needs to be done?." Proceedings of the VLDB Endowment 9.12 (2016): 993-1004.

[14] Josko, João Marcelo Borovina, Marcio Katsumi Oikawa, and João Eduardo Ferreira. "A formal taxonomy to improve data defect description." International Conference on Database Systems for Advanced Applications. Springer International Publishing, 2016.

[15] Chaudhuri, Surajit, and Umeshwar Dayal. "An overview of data warehousing and OLAP technology." ACM Sigmod record 26.1 (1997): 65-74.

[16] Riehle, Dirk, and Heinz Züllighoven. "Understanding and using patterns in software development." TAPOS 2.1 (1996): 3-13.

[17] Kandel, Sean, et al. "Profiler: Integrated statistical analysis and visualization for data quality assessment." Proceedings of the International Working Conference on Advanced Visual Interfaces. ACM, 2012.

[18] Kandel, Sean, et al. "Research directions in data wrangling: Visualizations and transformations for usable and credible data." Information Visualization 10.4 (2011): 271-288.

[19] Raman, Vijayshankar, and Joseph M. Hellerstein. "Potter's wheel: An interactive data cleaning system." VLDB. Vol. 1. 2001.

[20] Maletic, Jonathan I., and Andrian Marcus. "Data Cleansing: Beyond Integrity Analysis." Iq. 2000.

[21] Laranjeiro, Nuno, Seyma Nur Soydemir, and Jorge Bernardino. "A survey on data quality: classifying poor data." Dependable Computing (PRDC), 2015 IEEE 21st Pacific Rim International Symposium on. IEEE, 2015.

[22] Kandel, Sean. Interactive Systems for Data Transformation and Assessment. Diss. Stanford University, 2013.

[23] Simov, Alexander, et al. Metadata and Property-relatedLinked Data. ProDataMarket Deliverable D1.2. August, 2016.

[24] Tarasova, Tatiana, et al. Harmonization of data to SmartOpenData model. Final iteration. SmartOpenData Deliverable D3.5::Public. August, 2015. Available at http://www.smartopendata.eu/sites/default/files/SmartOpenData_D3.5_Final%20Data%20Harmonisation.pdf.

[25] Berlocher, Ivan, Seonho Kim, and Tony Lee. Use case implementatio, v1. DaPaaS Deliverable D5.2. October, 2014. Available at http://bit.ly/1Ib5uzJ.

[26] Hernández, Mauricio A., and Salvatore J. Stolfo. "Real-world data is dirty: Data cleansing and the merge/purge problem." Data mining and knowledge discovery 2.1 (1998): 9-37.

[27] Galhardas, Helena, et al. Declarative data cleaning: Language, model. and Algorithms. Technical report, INRIA, 2001.

[28] Sukhobok, Dina, et al. "Tabular Data Cleaning and Linked Data Generation with Grafterizer." International Semantic Web Conference. Springer International Publishing, 2016.

[29] Roman, Dumitru, et al. "DataGraft: Simplifying open data publishing." International Semantic Web Conference. Springer International Publishing, 2016.

[30] Roman, Dumitru, et al. "DataGraft: One-stop-shop for open data management." Semantic Web Preprint (2016): 1-19.

[31] Roman, Dumitru, et al. GIS Knowledge Base. InfraRisk Deliverable D2.2. March, 2016.

[32] David Stodder. Improving Data Preparation for Business Analytics. TDWI best practices report. 2016. Available at http://www.paxata.com/wp-content/uploads/tdwi-bpreport-data-prep.pdf.

[33] Rogoff, Kenneth, and Carmen Reinhart. "Growth in a Time of Debt." American Economic Review 100.2 (2010): 573-8.

[34] CAKIR, Goknur Seyma. "Development of a condition based maintenance decision model by data mining." School of Industrial Engineering, TUE (2011).

[35] von Zernichow, Bjørn Marius. Usability of Visual Data Profiling in Data Cleaning and Transformation. Master's thesis. University of Oslo, 2017.

[36] Sukhobok, Dina. Tabular Data Cleaning and Linked Data Generation with Grafterizer. Master's thesis. University of Oslo, 2016. Available at https://www.duo.uio.no/bitstream/handle/10852/51623/masterthesis.pdf?sequence=5&isAllowed=y.