Noname manuscript No.
(will be inserted by the editor)

Data Glitches: Monsters in your Data

Tamraparni Dasu

Received: date / Accepted: date

Abstract Data types and data structures are becoming increasingly complex as they keep pace with evolving technologies and applications. The result is an increase in the number and complexity of data quality problems. Data glitches, a common name for data quality problems, can be simple and stand alone, or highly complex with spatial and temporal correlations.

In this chapter, we provide an overview of a comprehensive and measurable data quality process. To begin with, we define and classify complex glitch types, and describe detection and cleaning techniques. We present metrics for assessing data quality, and for choosing cleaning strategies subject to a variety of considerations. The process culminates in a "clean" data set that is acceptable to the end-user. We conclude with an overview of significant literature in this area, and a discussion of opportunities for practice, application and further research.

Keywords Data glitches · Data quality assessment · Statistical cleaning · Correlated errors

1 Introduction

In the era of iPad and Facebook, where both information and entertainment are increasingly electronic, the types and volumes of data being generated are mind-boggling. Scientific and industrial applications, data networks, financial transactions and mobility applications - the common thread that runs through all domains of application is the quality of data. With increasing reliance on automation, the types of issues that plague data have become more frequent

Tamraparni Dasu
180 Park Avenue, Florham Park, NJ 07932, USA
Tel.: +001-973-360-8327
Fax: +001-973-360-8077
E-mail: tamr@research.att..com

and complex. They range from the simple and correctible, to problems that might not be identifiable, let alone remediated. Data quality issues can seriously skew the results of data mining and analysis, with consequences that can potentially cost billions; corporations could make erroneous decisions based on misleading results, machinery could be incorrectly calibrated leading to disastrous failures.

Data issues, if left unaddressed, get entrenched. They propagate rapidly across data systems, contaminating data in an exponential manner. The window of opportunity to clean data is often quite small because the original data might be unavailable later to reconstruct a clean version of the data. For example, if there are errors during the transmission of data, there is only a very short window of time, typically 24 hours, to request re-transmission before the access to the original data is lost forever. Therefore, it is important to detect and treat data quality issues in a timely fashion.

We introduce below a statistical approach to data quality assessment and cleaning. Given that data quality is highly application-specific and domain dependent, we describe a flexible framework within which metrics and solutions can be customized appropriately.

1.1 A Statistical Notion of Data Quality

Let $D = \{d_{ij}\}$ be the data matrix that is available to the user, where $i = 1, \ldots, N$ represents a row that corresponds to a unique entity, and column $j = 1, \ldots, d$ corresponds to an attribute. In reality, D can take more complex forms, but the following discussion applies irrespective of the exact structure of D. In our specification, D has N records or *samples*, and d attributes or *dimensions*.

In general, the data D are generated by a real-world process P such as a data network or financial transaction stream. The user has a certain perception of the data, D^*, derived from either experience or domain knowledge of P. The perception reflects the user's beliefs, assumptions and understanding of the data. For example, a user might expect that a particular column in the data matrix D corresponds to *revenues measured in dollars, and that it follows a log-normal distribution.* Any and all of these could be wrong due to flawed documentation or incorrect model assumptions. The attribute, in reality, could represent *production costs, measured in Euros, transformed to follow a Normal distribution.*

The disparity between the actual data D and the user's perception of the data D^*, is a measure of the quality of data. The farther D and D^* are, poorer the quality of data, less reliable the results, and less usable the data. The closer D and D^* are, the better the quality of data and more reliable the results. The goal of data cleaning or data repair is to bridge the disparity between D and D^*.

However, data cleaning or repair should not be performed indiscriminately. The process of cleaning should preserve the original statistical distributional properties, and not distort or transform them to such an extent the data are no longer representative of the real-world process P. Inference based on excessively transformed data could be meaningless, or worse, misleading.

In an ideal world, we should be able to clean the data and bring D and D^* closer, without distorting it too much. In the real world, it is more often a trade-off between cleaning and distortion, within the user's resource limitations and quality specifications. In this chapter, we provide a brief introduction to these concepts, and explain data quality assessment within the framework of statistical theory.

The rest of the chapter is organized as follows. Section 2 provides an overview of the statistical data quality process, emphasizing its iterative nature. In Section 3, we discuss new types of glitches and complex glitch patterns that can be leveraged to develop efficient context-dependent cleaning strategies. In Section 4, we describe methods for detecting different types of glitches, while Section 5 focuses on assessing the quality of data. Section 6 offers suggestions for cleaning and repairing data. In Section 7, we discuss how to choose a strategy from a multitude of potential strategies, based on the notion of the trade off between cleaning and distortion. A brief literature overview is provided in Section 8. Finally, we present our conclusions in Section 9.

2 Data Cleaning, An Iterative Process

Data cleaning is an iterative process. Typically, a user is presented with a data set D, and would like to clean it to meet cleanliness specifications ("no more than 10% missing values") subject to resource and cost considerations. The process of cleaning could potentially introduce new glitches and make the data dirtier. Therefore, after each cleaning step, the user re-measures the quality and repeats the process until the specifications are satisfied.

A schematic version of the data quality process is shown in Figure 1. It has four broad stages. We give a brief overview below, and describe each stage in detail in the following sections.

Define and Identify: The first step in the data quality process is to have a well-defined notion of what constitutes a data glitch. While there are established definitions for types of glitches (missing, duplicates, inconsistent), it is important to identify those that are relevant to the particular user and context. One user's glitch could be another user's treasured measurement. For example, an extreme observation, known as an outlier or anomaly, could skew one particular user's analysis, because she is interested in computing averages. The outlier would constitute a glitch and should be treated. On the other hand, another user might specifically want those extreme values. He might be interested in rare but high-risk events, like unusual loads on a data network, and would want to preserve them to better understand their occurrence patterns and their impact on network performance.

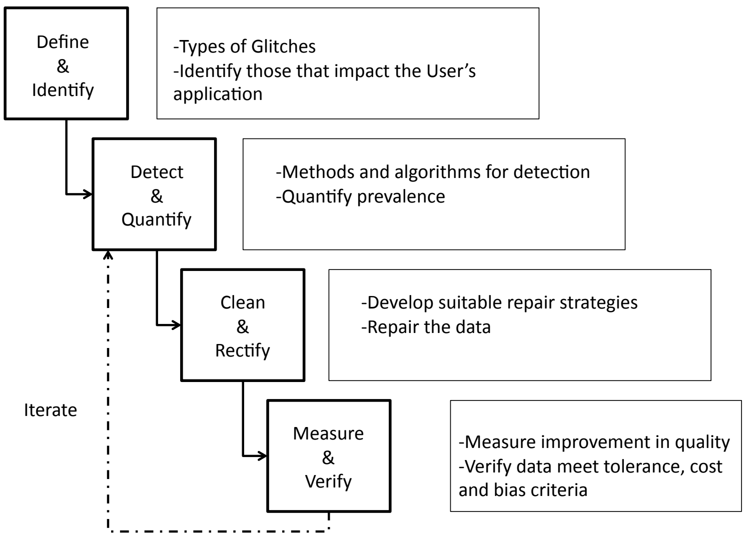
Define & Identify
-Types of Glitches
-Identify those that impact the User's application
Detect & Quantify
-Methods and algorithms for detection
-Quantify prevalence
Clean & Rectify
-Develop suitable repair strategies
-Repair the data
Measure & Verify
-Measure improvement in quality
-Verify data meet tolerance, cost and bias criteria
Iterate

Fig. 1 Overview of Data Quality Process: The process is flexible to allow the user to customize data quality assessment and repair to suit resource and accuracy requirements.

Detect and Quantify: Once the set of relevant glitches is defined, we need methods that can comb through the data and detect the glitches and tag them. *Glitch detector functions* are used to identify glitches and associate with each data cell d_{ij}, a vector of indicators that denote the presence or absence of each type of glitch. The resulting bit vector v_{ij} is called a *glitch signature*. Examples of glitch detector functions range from the simple, like the binary presence or absence of an attribute, to the complex, such as a multi-dimensional outlier detection technique. Glitch signatures can be complex as well, and need not be limited to bit vectors. For instance, in the case of outlier detection, they could potentially include p-values instead of a binary indicator of presence or absence of an outlier.

Once all the glitches are identified and tagged, we can quantify their prevalence and measure the quality of the data using an objective measurement called a *glitch score* and *glitch index*, a weighted average of the glitch signatures. Both glitch signatures and glitch scores can be customized to the needs of the user.

Clean and Rectify: It is not essential that the user should clean all the glitches. Some glitches are more expensive to clean than others, and some have little or no impact on the user's applications. Since for each type of glitch there are numerous options for cleaning, the user selects a suitable cleaning method.

In the absence of prior knowledge, an experimental framework can be used to choose an appropriate strategy from a set of candidate strategies. The methods are then applied to repair or clean the data.

Measure and Verify: The glitch score is recomputed after the cleaning, to see whether it meets the user's specified tolerance limits in terms of cleanliness. In addition, we also quantify the impact of data cleaning on the statistical distribution of D, called *statistical distortion*, measured using a histogram distance . If both the glitch score and statistical distortion are satisfactory, the cleaning process is over and the data are ready to use. Otherwise, the entire process is repeated, subject to user's cost considerations.

We will elaborate on each of the stages of the data quality process in the sections that follow.

3 Complex Data Glitches

There is a vast amount of literature on types of data glitches. We provide a brief summary of select references in Section 8. In general, data glitches are errors in the measurement and recording of data that adversely impact analysis. We begin our discussion with some examples of common data quality problems.

Missing data: Missing data are instances where individual data values, entire records and entire attributes disappear, due to a multitude of reasons: human error; malfunctioning software used for data collection, processing and storing; restrictive data entry interfaces; system errors. Given the volume of data generated in most modern applications, missing data could go unnoticed for long periods of time. If not addressed, missing data could result in serious biases in the analyses.

Inconsistent and faulty data: Inconsistencies arise in the data in many ways. Data are entered incorrectly, often due to manual error, and could be hard to detect. For example, when "09-07-2010" is recorded as "09-07-2001", it is very difficult to tell the entry apart from other legitimate entries, unless we have additional information that flags it as an error. Without a scalable method for validation, there is no way to determine whether the value is correct or not. Given the volume of data, manual checking is not a scalable method.

A variation of this error that is more easily detected is the case where 09/07/2010 is entered as 90/07/2010. We can use the constraint "If month = 07, then the day of the month must lie between 1 and 31" to identify this inconsistency. Constraint satisfaction methods are used widely to automate well-defined inconsistency checks by ETL (Exrtact-Transform-Load) tools.

Anomalies and outliers: Data points that are unexpected, while not necessarily incorrect or inconsistent, are potentially suspicious. For example, in the sequence of numbers "$1, 2, 8, 3, 0, 2, 3, 4, 1, 6, 123, 1, \ldots''$, 123 stands out as an unexpected value. Considerable literature in statistics and computer science has been dedicated to the detection and removal of unexpected data, for example [1] and [4].

Duplicates: Data records that claim to represent the same piece of information but have distinct, non-unique values are considered to be duplicates. For example, in a database where name and address uniquely identify an individual and the attribute of interest is height, the following two non-unique records are duplicates.

```
John Doe, 123 Main Street, Tall
John Doe, 123 Main Street, Short
```

Duplicates are often hard to define, because the notion of uniqueness of an entity varies from context to context within the same framework.

Undocumented data: One of the most egregious data quality problems is the lack of documentation. Without a proper data dictionary, what do the values in the following data record mean?

```
123-444-1212|3.59|2|145
```

If we know that the data pertain to a telephone record, we can guess that the first field corresponds to a telephone number, the second to a dollar amount, and the third to utilization in hours. But it would be a guess, and without documentation, the data are *unusable.*

There are numerous other data quality issues, including a variation in quality across different sections of the data, when data are integrated from sources with different recency. Some sections of the data could contain outdated information. Data integration itself could introduce data quality issues by joining together records that belong to different entities even though they have the same supposedly "unique" join key.

3.1 A Challenging Problem

Detecting and treating data quality problems is challenging due to complexity and application specific nature; it is difficult to to generalize or automate solutions. Other reasons include:

Relevance: The meaning, severity and treatment of data glitches vary from domain to domain. A marketer might be interested in typical values, while an engineer might be interested in outliers. The latter will have a lower tolerance to missing values while the former might not care as long as there is a reasonable sample to estimate typical values. This lack of generality makes it hard to automate data quality solutions across domains.

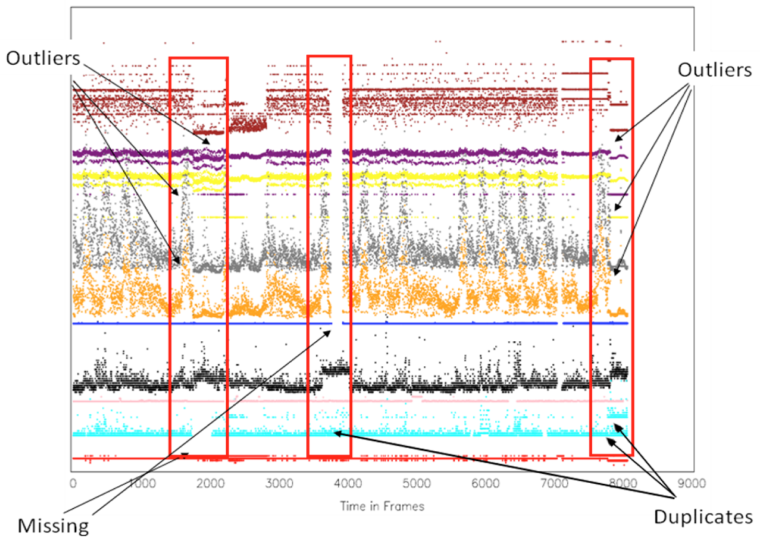

Fig. 2 Complex glitches: In a data stream of IP network data collected over a brief period, multiple glitches occur in complex patterns of static and temporal dependence. The plot shows 10 attributes (differentiated by color) on a transformed scale. Glitches are of multiple types, occurring at multiple times, affecting multiple attributes and multiple records.

Ambiguity: Sometimes, the boundary between good and bad data is not very clear. Different outlier detection methods could specify different thresholds for identifying abnormal values, resulting in potentially conflicting outcomes.

Complex dependence: Different types of data glitches co-occur, or, one type of glitch could mask the occurrence of others. For example, missing values could mask the presence of duplicates or outlying values.

Dynamic: In addition, as data evolve, the extent and type of glitches change over time, making glitch detection models obsolete.

In recent work [3], Berti-Equille *et al.* propose that glitches tend to be complex entities that co-occur in distinct patterns, as opposed to the common assumption of independent and random occurrence of glitches. As an example, consider Figure 2. It shows ten attributes (differentiated by color) related to the performance of a network element in an IP network over a brief period. The attributes are plotted on a transformed scale. Missing values, outliers and duplicates co-occur, sometimes with a short lag in time.

To capture the variety and multiplicity of these interdependent co-occurring glitches, the authors propose the following classification of complex types of glitches. Given a data set $D = \{d_{ij}\}$:

Multi-Type Glitch: A value $d_{ij} \in D$ has a multi-type glitch if there are at least two categories of glitches associated with it.

Concomitant Glitches: Two or more values d_{ij} and $d_{ij'}$ $(j \neq j')$ in D are concomitant glitches if they are both glitches and occur in the same data row i.

Multi-Occurrent Glitch: A multi-occurrent glitch c in the data set D is a glitch whose type is shared by two or more distinct values in the data set.

The paper demonstrates that patterns of dependence and co-occurrence of glitches can be leveraged to develop highly efficient, quantitative cleaning strategies that are specially tailored for the data set. By cleaning co-occurring glitches simultaneously rather than each glitch type independently, the user saves on resources and cycle times.

4 Glitch Detection

Glitch detection, like most aspects of data quality, is dependent on the application and the user. There are many tools in the fields of data mining, database theory and statistics to identify a wide variety of problems in the data. It could be as simple as identifying missing values ("value is not populated"), or as complex as identifying inconsistent values by checking against an exhaustive set of rules specified by domain experts.

In general, glitch detection methods fall into two broad categories: Constraint based methods developed in consultation with experts, or constructed from data properties and functional dependencies that the data must satisfy; and quantitative methods based on statistical and data mining approaches.

In our experience, an effective data quality auditing methodology must have both components. Data quality is highly domain and context dependent, and therefore it is important to incorporate domain knowledge gathered from experts into a set of rules or constraints that the data must satisfy. In addition, given the vast amounts of rapidly accumulating data, statistical techniques are essential to screen the data and identify smaller subsets for further analyses to identify patterns and co-occurrences. Such glitch patterns could aid root cause identification to eliminate future glitches, and guide effective data repair.

4.1 Constraint Satisfaction Methods

Constraint satisfaction procedures enforce compliance with specific constraints. Some constraints can be formulated based on heuristics or consultation with experts. For example, "duration should be non-negative", or "if US ZIP code = 07932 then state=NJ", or, "if wireless service=voice only, then text traffic=0". Any record that violates this rule is deemed inconsistent.

As another example, consider the constraint, “unique identifier should be present”. A unique identifier depends on the application, and is typically used to denote distinct samples, entities or measurements in the data, loosely speaking. For example, a credit card number and a time stamp together would constitute a unique identifier for a data stream of credit card transactions. Any record that does not meet this constraint could be considered missing, because the data measurements that are populated (“items bought, amount paid in dollars”) cannot be attributed to a distinct entity (“credit card number and transaction time”).

Unique identifiers are critical for identifying duplicates. Duplicate detection can be posed as a constraint as well. “If two or more records have the same unique identifier, then the two records should be identical in the remaining attributes”. Otherwise, they constitute duplicates that need to be remediated.

4.2 Statistical Methods

Statistical methods are most commonly employed for detecting inconsistencies and outliers. There is an abundance of outlier detection methods, such as those that employ error bounds, quantile-based methods, and model based methods. Most are univariate, that is, they detect outliers in each attribute separately. Common univariate methods include: $3 - \sigma$ limits of a Gaussian distribution; quantile-based methods such as the 5th and 95th percentiles of a distribution; and outliers with respect to statistical models such as regression based outliers.

Multivariate methods take into account the interdependence between attributes. A common multivariate outlier detection method uses the Hotellings T^2 statistic, the multivariate Gaussian equivalent of the $3 - \sigma$ test. Details of this method can be found in [12].

4.3 Patterns of Glitches

In addition to detecting individual glitches, it is important to detect glitch patterns and glitch dependence. Glitch patterns offer clues to efficient data cleaning strategies that are specific to the application. Figure 3 depicts a data stream generated by a data network, where a data record is scheduled to arrive in every polling interval. However, we noticed that in a large proportion of instances, missing values (dashed red arrows) in a polling interval were immediately followed by duplicates (solid red arrows) in the next polling interval. By consulting domain experts (network engineers), we found that an occasional delay in polling causes this pattern. Therefore, instead of using a blind cleaning strategy of imputing missing values using a mean or a median, we imputed missing values using the closest duplicate (in time) from the next polling interval. Thus, we were able to (1) treat two types of glitches (missing and duplicates) simultaneously, and (2) make good use of valuable data collected at great expense, rather than override it with estimated (mean, median) or synthetic (simulated) values. The paper [3] contains an exhaustive

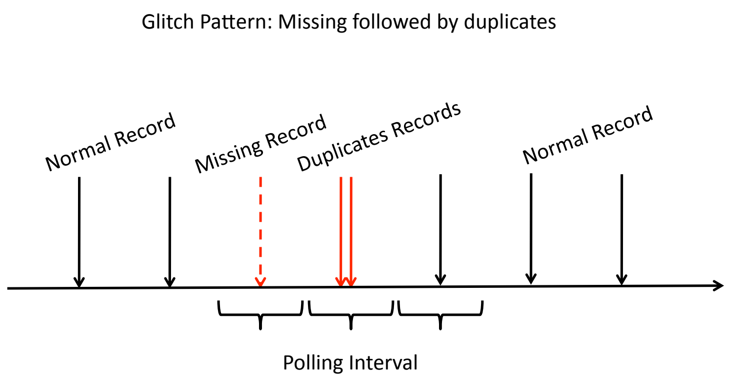

Fig. 3 Pattern of glitches: Missing values (broken red arrow) are followed by duplicates (solid red arrows). By imputing a missing value using the nearest duplicate (in time), we rectified two types of glitches at once, and kept statistical distortion low by using real values rather than statistical estimates like the mean or median.

discussion of how to use glitch patterns and dependence structure to develop cleaning strategies.

5 Quality Assessment

As in most aspects of data quality, data quality assessment is highly context and application dependent. It is difficult to formulate a general solution that will work in all situations. In this section, we outline a high level approach that can be customized to the needs of a specific user. There are three main steps: (i) annotating the glitches, (ii) weighting them according to the user's prioritization, and (iii) combining them into a single comprehensive score that can be used to evaluate quality, and measure the improvement in quality, The user can compare strategies using the glitch score and choose the most suitable one.

5.1 Glitch Signatures

A glitch signature is a way of summarizing the usability of a value in the data matrix. Each data element $d_{ij} \in D$ is mapped to a vector of bits $v_{ijk}, k = 1, \ldots, c$, where each bit corresponds to one of the c glitch types. For example, suppose that we are interested in $c = 3$ types of glitches – missing values, outliers and inconsistencies, in that order. In addition, suppose that the value

d_{ij} has an inconsistency associated with it. The corresponding vector of bits that indicates the presence or absence of each of these glitch types in d_{ij} is given by:

$$v_{ij} = (0, 0, 1). \tag{1}$$

The bit vector v_{ij} is called the *glitch signature* of d_{ij}. While this is a very simple example, see [3] for a more nuanced formulation of glitch signatures.

A value $d_{ij} \in D$ is "clean" if and only if every value in v_{ij} is zero, that is the sum of the bits is 0. Glitch signatures can be considered *augmented* or derived data that enable us to flag glitches, quantify their prevalence, and identify patterns in an analytic, automated fashion.

5.2 Glitch Weighting & Scoring

Glitch weighting allows the user to choose the importance of each type of glitch. Let $\{w_k\}, k = 1, \ldots, c$ be values between 0 and 1 such that $\sum_k w_k = 1$. If the user wishes to eliminate missing values but is not too concerned about outliers, then the weights can be chosen to reflect that. For instance, $w_1 = 0.6, w_2 = 0.1, w_3 = 0.3$ for the three types of glitches of missing, outliers and inconsistencies reflects their relative importance to the user.

The individual glitch score g_{ij} associated with each value in the data matrix is computed as follows:

$$g_{ij} = \sum_k v_{ijk} w_k. \tag{2}$$

For the above example, the glitch score would be:

$$0 * 0.6 + 0 * 0.1 + 1 * 0.3 = 0.3$$

.

5.3 Glitch Index

The overall glitch score $G(D)$ of the data set D, called *Glitch Index*, would simply be:

$$G(D) = \sum_{ij} g_{ij}. \tag{3}$$

Note that we can extend the notion of weighting to individual attributes (columns) and rows (records or samples) by devising additional weighting schemes similar to the one above. A network engineer might consider glitches in CPU usage data of a router to be more important than the latitude and longitude information of the physical location of the router. The latter does not change often and can be inferred easily from prior values.

6 Data Repair

Once the glitches have been tagged and quantified, we are ready to clean the data. Many considerations go into choosing a suitable strategy. The multitude of methods and tools available for this purpose is quite daunting. We give a brief overview below. The ultimate choice depends on the user's requirements.

6.1 Cleaning Methods

Glitch detection methods often include a provision for cleaning the data. A comprehensive overview of data cleaning methods can be found in Chapter 5 of [5]. Additional material can be found in [2]. We describe the salient points below.

Missing Value Imputation and Treatment of Inconsistencies Missing value imputation methods can range from the simple to highly complex, providing a perfect example of the wide spectrum of methods available. The simplest approach consists of replacing missing values in a numeric attribute by a *statistical estimate* such as a mean or a median of the non-missing values of that attribute. Or, we can take into account an attribute's relationship to other attributes by using *regression models* to "predict" the value of the missing attribute as a function of other non-missing attributes. A more exhaustive approach would entail simulating a joint distribution of all the attributes and imputing values drawing from the simulated distribution. Statistical software SAS includes the module PROC MI for missing value imputation.

Occasionally, a default value, such as 9999 or -10000 is used to denote missing values. Usually, these are easy to detect, unless a common value like 0 is used to denote missing values, or, there is no standard representation, resulting in multiple ways of denoting missing values. A data browser like Bellman [6] can be used to identify such non-standard representations.

Non-numeric attributes can be imputed using heuristics and functional dependencies. For example, "if ZIP=07932 and STATE=missing, then impute STATE=New Jersey."

There is a vast amount of literature on ETL (Extract-Transform-Load) cleaning tools in the database community. See the tutorial [2] for details. In the absence of domain specific solutions, missing value imputation methods can often be used to correct inconsistencies.

Outlier Treatment Outliers are data values that are unexpected, based either on likelihood of occurrence, or on other criteria. Whether outliers should be treated, or preserved in the data, depends on the user and application. Because outliers tend to be few in number, the easiest solution is to simply *drop* them from the data set. If that is not an option, a common approach in statistics is *Winsorization*, where outliers are replaced with the nearest non-outlying value of the attribute. For example, if any value beyond the $3 - \sigma$ is considered an outlier, then every outlier is replaced with the closest $3 - \sigma$ limit.

Other approaches may entail treating outliers separately while developing models such as regression.

De-duplication There is extensive literature in the database community on the treatment of duplicate records. See Elmagarmid *et al.* [4] for an overview of duplicate detection in databases. Data repair consists of eliminating the duplicates by means such as retaining a random record, or fusing the duplicates into a single unique record as discussed in [11].

For example, if there are multiple usage records associated with a given router on a network with the same time stamp, we could (1) retain the first (or last, or middle) record, or (2) choose a random record each time, or (3) combine all the records for the router with the same timestamp using averaging techniques.

Pattern driven methods In Section 4.3, we discussed data-driven strategies that are developed using glitch patterns. Data mining methods and analytical techniques are used to identify these patterns. Domain experts examine the patterns and participate in developing cleaning strategies. This is an excellent way to customize solutions to specific problems. Even in the absence of domain experts, the dependence between glitches, and their correlation patterns still offer valuable clues to glitch treatment as discussed in [3]

7 Choosing Data Cleaning Strategies

We have used the phrases data cleaning method and data cleaning strategy interchangeably. However, to be more specific, a strategy is a concatenation of methods. For example, we might start by imputing missing values, then remove duplicates and finally treat outliers, a strategy that is a concatenation of three methods. Even with as few as three types of glitches, different methods of treatment can lead to an exponentially large number of possible strategies as illustrated in Figure 4. Given that the order of treatment has an influence on the outcome, the possibilities are even more overwhelming.

In [7], Dasu and Loh suggest using a three pronged approach to evaluating data cleaning strategies that takes into account: (1) improvement in glitch scores, (2) impact of the strategy on statistical distribution of the original data, and (3) cost. We explain below.

Glitch Index Improvement Suppose that the glitch index of the original data set D is $G(D)$ as described in Section 5. After applying strategy S, we obtain a treated data set D^c, whose glitch index is $G(D^c)$. The glitch improvement in the data set D caused by the strategy S is $\Delta_S(D)$, given by:

$$\Delta_S(D) = G(D^c) - G(D). \tag{4}$$

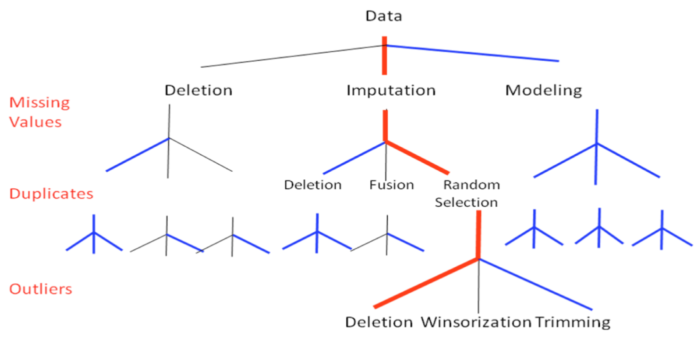

Fig. 4 A simplistic cleaning strategy tree: With a sequential, pre-defined, one at a time approach to treating data glitches, the number of possibilities is exponentially large. In this example, we consider three treatment possibilities for each of three data glitches (missing, duplicates and outliers) and arrive at 27 possible strategies.

The data cleaning process entails applying a sequence of strategies $\{S_i\}$ until the final data set D^F meets the user's specifications. Note that the sequence of $\{\Delta_{S_i}(D)\}$ need not be monotonic. It is possible that a strategy might introduce more glitches, increasing the glitch index. For example, imputing missing values might introduce duplicates and outliers.

Statistical Distortion Dasu and Loh [7] propose the notion of *statistical distortion* to measure the effect of a cleaning strategy on the distributional properties of the original data D. The data are generated by a real-world process (traffic through a network router) P, and by cleaning the data, we are altering data values, and distorting the statistical properties of the original data. When this statistical distortion is excessive, the cleaned data are significantly different from the original data and no longer represent the real-world process P. This could be the biggest data quality problem of all!

Statistical distortion measures the distance $\mathcal{D}(D, D^F)$ of D from D^F. Distance between two data sets can be measured in many ways. Simple approaches include comparing means and other summary statistics of the two data sets. More sophisticated methods entail comparing multivariate distributions of the data sets using histogram distance metrics such as the Earth Mover Distance. See [7] for details.

Cost Considerations Along with glitch improvement and statistical distortion, cost C plays a major role in the choice of cleaning strategies. Every cleaning operation has a cost, whether it is computational or human (expert opinions). The user typically sets an upper bound K on cost, before the data cleaning process is undertaken.

Therefore, a *candidate set of strategies* $\mathcal{A}$ consists of strategies that meet the following criteria:

$$\begin{aligned} C &< K \\ G(D^F) &< r \\ \mathcal{D}(D, D^F) &< d \end{aligned}$$

where C represents the cost, K the maximum cost that the user is willing to incur, and r and d are tolerance bounds for the glitch index and statistical distortion respectively. It is likely that there are multiple strategies that meet these criteria, in which case the user can choose the one that is most desirable among them (e.g. least expensive or least distorting).

7.1 An Experimental Framework

Note that it is often impractical to evaluate strategies by applying them to the data first. It defeats the very purpose of strategy selection. In the absence of other guidelines or expert opinion, strategies can be selected empirically using an experimental framework. The framework entails sampling from the data D, and applying the strategies to multiple samples of data to evaluate them on the three criteria described above. See [7] for details.

8 Relevant Literature

We describe below a set of references that is neither complete nor exhaustive, but is intended to serve as a starting point for further study.

Redman [13] provides an introductory overview of traditional definitions of data quality problems. Dasu and Johnson [5] discuss a comprehensive view of data quality and take a technical approach to the definition and cleaning of data glitches, and include those based on process, constraint violation, and data interpretation.

Barnett and Lewis [1] discuss a statistical approach to outlier detection, while Chandola *et al.* survey the literature on anomaly detection from a computing perspective in [4]. Duplicate records, their definition, detection and removal is reviewed by Elmagarmid *et al.* in [8]. *Bellman*, discussed in [6], is a database browser for exploring glitches and glitch distributions in databases. Golab *et al.* [9] describe the use of functional dependencies to formulate constraints for identifying inconsistencies, and to isolate subsets that match or violate the constraints. See [10] for a comprehensive overview of outlier detection methods.

The iterative nature and importance of measuring data quality after every cleaning pass is discussed by Berti-Equille and Dasu in [2]. The tutorial also provides and exhaustive introduction to recent advances in data quality and data quality mining.

Glitch signatures, patterns of glitches, glitch index and glitch scoring are discussed in detail in [3].

A survey of data cleaning methods can be found in [2].

In [7], the authors propose a three-pronged approach for evaluating data cleaning strategies, including the novel concept of statistical distortion which measures the change in the statistical properties of the data caused by data cleaning. They also discuss an experimental framework for empirically choosing the best cleaning strategy as defined by their three-pronged approach.

9 Conclusion

In this chapter, we have presented a comprehensive data quality process that includes an overview of data glitches, statistical approaches to assessing data quality, and the development and evaluation of data cleaning strategies. This is an active and exciting area of research given the sheer volume and variety of data types that are emerging as a consequence of the confluence of information, entertainment and mobility. Open problems include glitch correlation, new metrics for data quality evaluation, glitch scoring and statistical distortion [7].

References

1. Barnett V., Lewis T., Outliers in statistical data. John Wiley, Chichester, (1994).
2. Berti-Equille, L., Dasu, T. Advances in Data Quality Mining, Tutorial, KDD (2009).
3. Berti-Equille L., Dasu T., Srivastava D., Discovery of complex glitch patterns: A novel approach to quantitative data cleaning, IEEE Conference on Data Engineering, (2011).
4. Chandola V., Banerjee , Kumar V., Anomaly Detection: A Survey. ACM Computing Surveys, (2009).
5. Dasu T., Johnson T., Exploratory data mining and data cleaning. John Wiley, New York, (2003).
6. Dasu, T., Johnson, T., Muthukrishnan, S., Shkapenyuk, V., Mining Database Structure; Or, How to Build a Data Quality Browser, Proc. SIGMOD, (2002).
7. Dasu, T., Loh, J. M., Statistical Distortion: Consequences of data cleaning, PVLDB 5(11):1674-1683 (2012).
8. Elmagarmid A. K., Ipeirotis P. G., Verykios V. S., Duplicate Record Detection A Survey. IEEE Transactions on knowledge and Data Engineering (TKDE) Vol. 19 No. 1, pp. 1-16 (2007).
9. Golab, L., Saha, A., Karloff, H., Srivastava, D., Korn, P., Sequential Dependencies, VLDB (2009).
10. Kriegel, H., Kroger, P., Zimek, A., Outlier Detection Techniques. Tutorial, PAKDD (2009).
11. Liu, X., Dong, X. L., Ooi, B. C., Srivastava, D., Online Data Fusion, PVLDB 4(11): 932-943 (2011).
12. Rao, C. R., Linear statistical inference and its applications, John Wiley and Sons, (1973).
13. Redman T., Data quality for the information age. Artech House, USA, (1997).