Data Quality for Temporal Streams

Tamraparni Dasu, Rong Duan, Divesh Srivastava
AT&T Labs - Research, Bedminster, NJ 07921, USA
{tamr, rongduan, divesh}@research.att.com

Abstract

Temporal data pose unique data quality challenges due to the presence of autocorrelations, trends, seasonality, and gaps in the data. Data streams are a special case of temporal data where velocity, volume and variety present additional layers of complexity in measuring the veracity of the data.

In this paper, we discuss a general, widely applicable framework for data quality measurement of streams in a dynamic environment that takes into account the evolving nature of streams. We classify data quality anomalies using four types of constraints, identify violations that could be potential data glitches, and use statistical distortion as a metric for measuring data quality in a near real-time fashion. We illustrate our framework using commercially available streams of NYSE stock prices consisting of aggregates of prices and trading volumes collected every minute over a one year period from November 2011 to November 2012.

1 Introduction

In today's data driven world, decisions are based on analysis applications that process continuous streams of data. Typically, the data are summarized or characterized by a *statistical signal* that represents the data within some bounds of uncertainty. The statistical signal could be as simple as a mean and standard deviation, or as complex and comprehensive as a joint probability distribution. Analysis applications rely on this signal to make predictions, forecasts and to create reports, dashboards and visualizations. Data quality issues interfere with this signal and distort the data, leading to misleading analysis outcomes and potentially bad decisions.

When there are data quality issues in temporal data, such as gaps (missing data), or dips (incomplete data) and spikes (duplicate data), they impact the underlying, constantly evolving density distributions. Some bins in the histogram will have less mass (missing/incomplete) than normal while other bins will have additional mass (duplicates) as compared to the norm resulting in distorted densities. Abnormally high values (inconsistent data) will show up as outliers in the tails of the distribution. Peculiar to streaming data, late arrivals (untimely or out-of-order data) will manifest as holes in the density at their expected arrival time, and spikes in the density when they actually arrive. Figure 1 illustrates some of these concepts using data streams that measure volumes of data transmitted by two correlated data sources depicted in red and blue. We will discuss it in detail in Section 2.3.

Since data are constantly changing over time (for example, level shift in Figure 1), the nature of data quality issues as well as the extent to which they are present changes too. Therefore the notion of what is acceptable in terms of data quality changes over time as well. In addition, it is important to take seasonality, periodicity

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

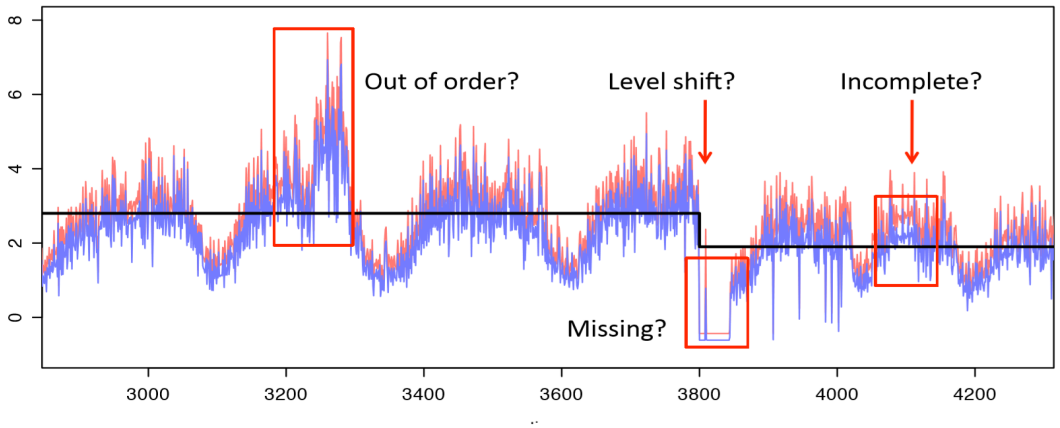

Figure 1: *Two correlated data streams with data quality glitches. Each color represents a different stream. Notice that glitches can be correlated and have complex dependence patterns such as: (a) data that arrives late (timeliness) as indicated by lower volumes followed by higher volumes (out of order), (b) missing values followed by a level shift when a generative process changes, e.g., by dropping a portion of the stream, and (c) incomplete data for the blue stream when the two streams diverge.*

and autocorrelations into account while determining whether a data point is a potential data quality concern. For example, it is certainly the case that trading volumes of stocks are higher at the beginning and end of the day, and very low on the eve of national holidays. It is also critical to consider dependence structure in glitches to prevent overstating of glitches. For example, out-of-order data manifest as missing followed by duplicates. Similarly, in correlated streams, if both streams have the same unusual behavior it might be a genuine feature of the generative process, while divergence between otherwise correlated streams could indicate incompleteness of one of the streams.

In addition to the statistical signal, domain-specific logical constraints need to be embodied as a set of data quality rules to complement the statistical signal. Toward this end, it is important to consult domain experts in formulating these rules as well as in refining them. Otherwise, violations of such constraints may not be data quality issues but more a reflection of incomplete or improperly specified constraints. For example, in the world of telecommunications, call volumes are exponentially higher on certain holidays like Mother's Day. Therefore, any constraint that states that call volumes need to be within a certain range $[Low, High]$ will need to reflect the exceptions.

A variety of logical constraints proposed by the data quality research community can be used to capture domain knowledge specifically in temporal streams, including order dependencies [5], sequential dependencies [4], speed constraints [6] and temporal rules [7]. These can be naturally used within the framework for data quality, discussed next.

2 A Framework for Data Quality in Temporal Streams

We discuss a general, widely applicable framework that addresses data quality concerns unique to temporal data. The methodology applies to numerical data as well as categorical data treated as group-by variables, counts or proportions. The approach is empirically driven and can be applied even in situations where there is no prior

Type	Single Column	Multi-column
Single row	Type 1: Applies to a single attribute and single entity	Type 2: Depends on multiple attributes but only one entity
Multi-row	Type 3: Applies to a single attribute but depends on multiple entities	Type 4: Depends on multiple attributes and multiple entities

Table 1: Types of constraints

domain knowledge. Even if the error rates are high, as long as they are not uniformly spread across the data, the method can isolate concentration of errors in specific subsets of the data.

The framework consists of the following primary stages, stated only briefly here. A detailed discussion follows.

- Formulate or learn rules or constraints that embody the characteristics that data need to meet (within some margin of error or uncertainty) in order to be deemed of a good quality. We call this the *ideal*, D_t^I, at time t.
- Benchmark the observed data D_t against the ideal D_t^I for *unexplained* violations of constraints and flag them as possible *glitches*. Note that violations are not necessarily data quality issues. As demonstrated in [9], many violations have "explanations" that are not captured by extant rules or constraints. These explanations can be used to refine the data quality rule set to prevent them from being flagged in the future.
- The presence of glitches causes the data to deviate from expected or ideal behavior. Capture the extent of deviation using the notion of *statistical distortion* $SD(t)$ introduced in [8]. A high distortion could mean that data D_t might not be suitable for use by downstream applications.
- Using the statistical distortion $SD(t)$ as a *data quality metric*, continuously monitor D_t to identify unusual distortion that falls outside acceptable ranges and alert responsible parties.

We explain in detail in the following sections.

2.1 The Ideal, D_t^I

Data streams are consumed by numerous downstream analysis applications that generate reports, alerts and other information used for decision making. These applications expect the data to meet certain criteria embodied in assumptions like: "Data must contain no missing attributes" (completeness), "a file of size X Megabytes arrives every minute" (timeliness, completeness), "the file size should not exceed Y Megabytes" (presence of duplicates?), "the files should not contain outliers in excess of some threshold, say 1%", or "if the zip code is Z, then the city has to be C". Such constraints describe an ideal D^I of what the data is expected to satisfy.

In the context of databases, multiple data sets could satisfy data quality constraints and match the notion of what is acceptable. And associated with each of these data sets is a data distribution $\mathcal{F}_i$ that captures the statistical properties of that particular data set. Therefore, an ideal D^I could be thought of as a mixture of these distributions $\{\mathcal{F}_i\}$. In the case of temporal data, the ideal itself could be changing with time to accommodate changes in the underlying generative process ("the system will switch from few large files to many smaller files to transmit the same amount of data") or changing needs of applications, and is denoted as D_t^I, where t is time. In the absence of pre-specified formulations of the ideal, it is possible to *learn* D_t^I from historical data.

In this paper we will use the words "ideal" and "expected" interchangeably to describe D_t^I.

2.2 DQ Constraints

As evidenced by the preceding discussion, constraints play an important role in specifying the ideal D_t^I. We classify these data quality constraints by their dependence on *attributes* or features (typically columns in a database table) and *entities* or instances (rows). An entity or row can represent something as simple as an individual identified by a "Customer_ID" and described using attributes like "Age", "Name", and "Phone Number" stored in columns. An entity could also be more complex and hierarchical like a corporation, with branches and sub-branches. Such a classification based only on row-column dependence is conceptually clean and closely relates to the cost of repairs. Logical, representational and statistical constraints all have a place in it and the framework is not dependent on any particular constraint language.

2.2.1 Constraints-Type 1

Type 1 constraints depend on a single row and column, i.e., a single individual value. For example, the constraint "should be an integer" can be validated by examining a single value with no additional reference to other rows or columns. Examples of violations include formatting problems that make the data hard to parse and access. Such representational constraints can be *learned* from existing data using data profiling and data browsing techniques (e.g., see [3] and [1]). Type 1 constraints are inexpensive to validate since they require access to a single cell in the database.

2.2.2 Constraints-Type 2

Type 2 constraints involve other attributes from the same row. For example, in the state of New Jersey sales tax cannot exceed 7% of the price of an item; this can be formulated as a Type 2 constraint, "if State equals NJ, then Sales Tax $<=$ 0.07*Price", which can be naturally expressed as a denial constraint. Note that we need two other columns (State and Price) to validate the value in a single row of the attribute "Sales Tax".

2.2.3 Constraints-Type 3

Next, consider constraints that involve values in a column across multiple rows. These are often based on comparisons between two or more rows (which is the case for many logical constraints) or distributions/aggregates (which is the case for many statistical constraints).

The notion of single-attribute *uniqueness* or *keyness* is a Type 3 constraint. The constraint "Column A is a key" can be specified by simply comparing two rows at a time, and checking that the Column A values are different. Similarly, detection of statistically anomalous values within a single attribute involves validating a Type 3 constraint, e.g., "Data should not lie outside three standard deviations from the mean," where the mean and standard deviation are statistical summaries computed from multiple rows of that column.

2.2.4 Constraints-Type 4

Finally, we define constraints that involve multiple rows as well as multiple columns to be of Type 4. For example, "Attributes A and B should generally trend in the same direction as measured by a correlation of greater than 0.8". Functional dependencies of the form "if two rows have the same SocialSecurityNumber, they must have the same LastName" are also Type 4 constraints. Consistency with an independent, external data source is a Type 4 constraint as well. Identifying multi-attribute *duplicates* has a similar flavor.

Note that the above classification roughly reflects the amount of data that needs to be accessed for validation and changed for repair, capturing the cost as well as the amount of data that could be impacted by potential repairs.

2.3 Identifying Data Glitches

Data glitches are often discovered through data exploration and visualization. Figure 1 shows examples of data glitches in the arrival of two correlated data streams, red and blue. In general, the streams move together. However, between t=3200 and t=3400, there is a drop in the amount of data that arrives, followed by an increase, indicating that data have arrived late causing them to be out of order. At t=3800, there is a short period of missing data followed by consistently lower volumes indicating that the process may have been redesigned to drop some component from transmission. Finally, note the brief period of divergence in the amount of data delivered for the two streams, indicating that the blue stream might be incomplete.

Given how increasingly critical data are to applications and businesses, glitch discovery has become an integral part of the early stages of data management. Data glitches are identified by validating a given instantiation of data for constraint violations. As noted earlier, not all violations of constraints constitute glitches. Some are "suspicious" at first glance but can be explained by experts as legitimate data. For example, in telecommunications, certain holidays have a big impact on network traffic and abnormally high or low volumes are acceptable, and could be explained, e.g., "We expect significantly higher call volumes on Mother's Day."

Domain knowledge in the form of explanations of anomalies can then be used to refine the constraints to reflect the conditions under which the constraint could legitimately be violated, e.g., Mother's Day. Since not all constraint violations are data glitches, it is important to seek glitch explanations to avoid over-treating the data, and embody the explanations as additional constraints to better reflect the ideal, D_t^I. See [9] for details about generating empirical glitch explanations and refining domain knowledge.

2.4 DQ Metric: Statistical Distortion

Analysis applications typically rely on statistical distributions because they drive the signal in the data. Data glitches interfere with this signal, as do indiscriminate attempts to clean the data. In [8], this notion of distortion to the signal in the data caused by cleaning was generalized as *statistical distortion*. Intuitively, statistical distortion is a data quality metric that measures the discrepancy between the ideal that we expect to receive at time t, D_t^I, and the data D_t we actually observe at t, within some acceptable statistical variation. For example, if we expect to receive 10 files every hour, and if we receive 7 files in the current hour, is that too few, or within expected statistical variation?

Formally, given the ideal D_t^I and current data D_t, *statistical distortion* measures the statistical distance $\mathcal{D}$ between them:

$$SD(D_t) = \mathcal{D}(D_t^I, D_t).$$

When the current data D_t is significantly different from the ideal D_t^I, we expect the statistical distortion to be high and vice versa. As a corollary, any good cleaning technique should reduce the statistical distortion and bring the cleaned data within an acceptable distance of the ideal distribution.

Statistical distortion can be captured using many metrics, from simple to complex: it could be a basic difference in proportion of constraint violations between the ideal D_t^I and the given data set D_t, or more data-specific distances like the Mahalanobis Distance (MD), Kullback Leiber Divergence (KLD) or the Earth Mover Distance (EMD). Glitch based distances like the difference in proportions are easy to compute and appropriate for the streaming setting. The difference in actual statistical distributions of the D_t^I and D_t captured by MD, KLD and EMD are computationally more expensive but capture the amount of disparity. These distances are useful for drilling down to the specific attributes or entities that are causing the disparity, and help in identifying the source of the glitches. Application needs and resource constraints determine the choice of distance used.

2.5 Making the Data Usable

Once the data glitches are identified and validated, data are made usable by making changes or repairs to the data to align it with expectation D_t^I, within acceptable limits of statistical distortion. Note that the repairs are often specific to applications. Marketing analysts focus on typical values that represent a group and hence will not tolerate outliers, while network engineers are interested in the rare catastrophic outlier and will not tolerate any missing values. In general, human experts may need to be involved to identify the appropriate repair strategy. In this paper, we will not focus on general repair strategies, but will discuss specifics in the context of our case study below.

3 A Case Study: NYSE Data

We illustrate the concepts discussed in the preceding sections using NYSE stock data, a classic example of temporal streams. We were able to access the raw minute-by-minute aggregates of stock prices and trading volumes. In order to address concerns specific to financial streams, we focus on quality concepts outlined by Thomson Reuters (TR) in their guidelines for cleaning financial data streams [11].

3.1 Description

The case study data consists of "minute bars" from over 2000 stocks traded on the New York Stock Exchange (NYSE) spanning the period from November 2011 to November 2012. The attributes, recorded within every minute interval [DT_t, DT_t + 1 minute) where DT_t is the date-time at time t described below, are:
trading time (YYYY-MM-DD hh:mm) DT_t,
day of the month,
opening price O_t,
highest price observed during the trading minute H_t,
lowest price L_t,
closing price C_t,
and the trading volume V_t.
Not every stock trades every minute. When there is no trade, no record is generated for that stock resulting in an irregular temporal stream. A sample of the data:

```
2011-11-01 09:38,1,21.75,21.78,21.75,21.76,1200
2011-11-01 09:39,1,21.74,21.75,21.73,21.74,1481
2011-11-01 09:40,1,21.75,21.77,21.74,21.76,6675
2011-11-01 09:41,1,21.77,21.81,21.76,21.79,1713
2011-11-01 09:42,1,21.8,21.82,21.78,21.78,2655
```

The Stock ID is embedded in the filename. We consider "StockID + Timestamp" to be a unique key. A separate table provides the mapping from Stock ID to StockTickerName, e.g., StockID=968 and StockTickerName="HAL" for Halliburton.

In addition, we compute derived variables such as the price spread,

$$SP_t = H_t - L_{t'},$$

consecutive changes, delta, in a given attribute, say H_t, the intra-minute high,

$$DL_t = H_t - H_{t'},$$

and the lag, LG_t, between successive trading minutes

$$LG_t = t - t',$$

where t' is the last observed actively traded minute.

We use the NYSE data stream to do two types of data quality checks. The first set of checks monitors the process of *data gathering,* i.e., is the data arriving in an expected fashion? The second set of checks monitors the *data content* for the quality of the contents of data received.

3.2 Glitch Detection

We illustrate using two approaches to glitch detection. Thomson Reuters (TR) proposes a deterministic percent change over previous value i.e they consider D_{t-1}, the previous value, to be the ideal, D_t^I, for the purpose of measuring data quality. In addition to the TR approach, we use the Feed Inspection Tool (FIT) for monitoring streaming data, discussed in detail in [10].

Briefly, FIT computes statistical summaries (mean and higher order moments, quantiles, counts) based on historical data in a sliding window to empirically estimate the ideal D_t^I. The current chunk of observed data D_t is validated against D_t^I. The comparison can be done at various levels of temporal granularity (5 minutes, 15 minutes, every hour) and the summaries can be made fine-grained by partitioning the data into groups such as day-of-week (DoW) and time-of-day (ToD). Note that running a simple outlier detection within each trading day is not very effective. It is well known that trading behavior is dependent on the time of the day. There is heightened activity in the morning when the market catches up with after-hour activity and again at the end of the day when traders try to "flatten their books" by hedging or unwinding risky positions. In addition, important economic numbers are often announced mid-morning causing markets to react. An outlier detection mechanism that does not take the time-of-day effect into account will generate spurious outliers. By limiting the history used in computing expected behavior via an appropriately chosen sliding window, or by deprecating the contribution of data over time using exponential decay models, FIT ensures that the empirically computed ideal evolves with the data and captures changing distributions accordingly. The choice of group by variables ToD and DoW as well as the size n of the sliding window affect the detection of anomalous values by FIT, while the choice of percentage threshold will affect TR's anomalies. We address this aspect in Section 3.5.

3.3 DQ: Data Gathering

We first check for completeness and timeliness by monitoring the number of stocks that report each minute. The reports, known as "minute bars", are delivered as individual files for each stock. The check for file counts by both TR as well as FIT are Type 4 constraints since they use multiple entities (stocks), at multiple times (previous value for TR, and for FIT, values in n rows where n is window size), and multiple attributes for FIT (file counts, and derived attributes ToD and DoW extracted from time stamps).

The left panel of Figure 2 depicts the results of running FIT on the average rate of file arrivals per minute within a given hour, compared to the expected average file rate per minute for that hour based on the sliding window FIT model. We ran FIT with several window sizes and thresholds and did not find any significant change in the results. In the plot, big pink dots represent the morning hours starting at 9:30 AM, gradually changing into smaller blue dots as the day goes by, with the smallest bluest dot for the hour 15:00. We plot only three hours–first, middle and last–to avoid clutter in the plot. The red dots are anomalous values. The size of the red dots allows us to determine which hour of the day they correspond to. According to the plot, there were anomalously low rates of file arrivals per minute during the last hour of July 3, and abnormally high values on Sep 12, to mention just a couple.

The data gathering checks do not require us to open the files, or parse them to examine the content; we just count the number of files, and if needed, their size. This is a fast DQ check and can prevent resources from

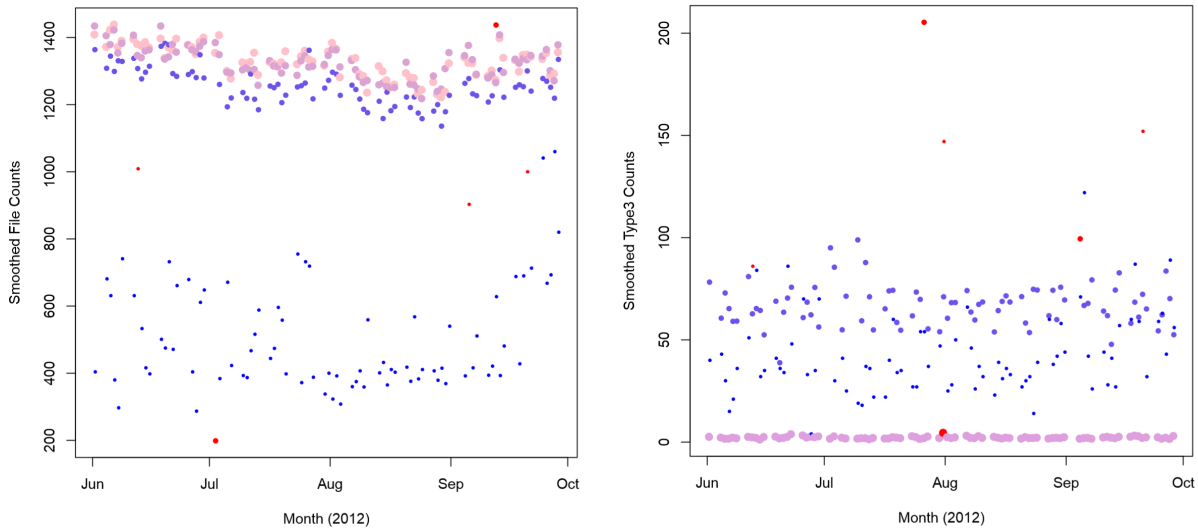

Figure 2: *The panel on left depicts the results of running FIT on the average rate of file arrivals per minute within a given hour. Big pink dots represent the beginning of the trading day at the morning hours gradually changing into smaller blue dots as the day goes by. The red dots are anomalous values. The size of the red dots allows us to determine which hour of the day they correspond to. There were anomalously low rates of file arrivals per minute during the last hour of July 3, and abnormally high values on September 12, to mention a couple. The panel on the right shows anomalous values of statistical distortion as measured by the total of Type 3 glitch counts,* $G^3(t)$.

being wasted on ingesting incomplete files. Figure 3 shows details of two sample anomalous days July 3 and September 12. The box plots in each figure represent the two underlying baseline models to which the days July 3, 2012 and Sep 12, 2012 are compared respectively. The red dots represent the number of stocks that trade each minute within a given hour, for the given day.

On July 3, the trading dropped off at 1 PM, due to an early close of the stock market on the eve of the July 4 holiday. This is an explainable anomaly. On September 12, 2012, there was an unexplained upward trend. The shift here was subtle, hence it took a statistical model like FIT to detect the change. FIT, because it takes into account the day-of-week effect, had already accounted for spikes on Wednesdays caused by Fed announcements. The higher values observed on September 12, 2012 were *outliers* and *inconsistent* with historical data even after taking that fact into account, and could not be explained using a correlated stream of times of Fed announcements.

We decided to investigate further and consult a domain expert. We found that the Fed, for the first time in 2012, had given a definitive extension until 2015 for interest rates, causing stocks to trade more actively as observed by a general shift of red dots toward higher values after 2 PM. (There was an upward trend earlier that day as word leaked out and the markets anticipated the Fed's decision). By seeking additional domain knowledge that is not available freely or in an automated fashion, we were able find an explanation. This is an example where a data anomaly would have been deemed a glitch because the explanation for it is not easily accessed in an automated fashion. Perhaps with sufficient effort, a majority of anomalies might be explained isolating a set of "true" glitches. However, given resource constraints, the process of generating explanations falls short of this clean separation of glitches and explainable anomalies.

Note that glitches captured by monitoring file counts are correlated glitches–movements that affect a large number of stocks.

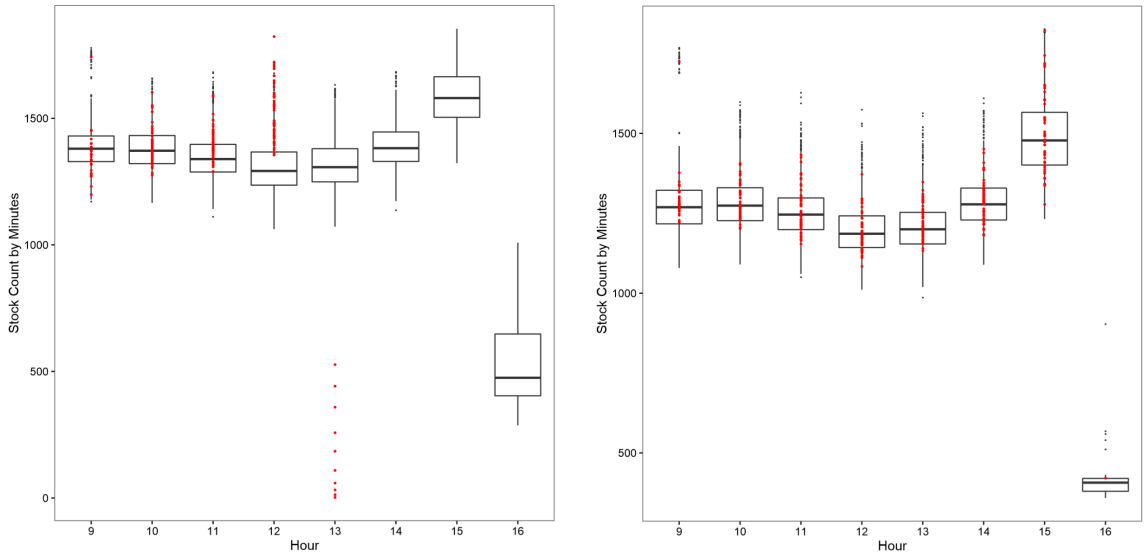

Figure 3: *Two sample anomalous days July 3 and September 12: number of stocks that trade each minute, within each hour. The box plots in each figure represent the two underlying baseline FIT models to which the days July 3, 2012 and September 12, 2012 are compared respectively. The red dots represent the observed values for the given day. While the July 3 dip was obvious, the September 2012 highs were more subtle. Notice the skew of red points toward higher values, particularly pronounced in hours 11,13, 14 and 15. It required an expert Fed watcher to provide an explanation.*

3.4 DQ: Data Content

A vast majority of DQ checks relate to content of the actual data. We consider stocks individually while detecting glitches, but it might be necessary to consider market behavior (correlations among stocks) for explaining or validating glitches, e.g., did all the stocks experience a drop or spike in volumes?

3.4.1 Type 1 Constraints

For each of the NYSE stocks and for each of the trading minutes, we follow the Thomson Reuters guidelines to formulate the following constraints:
(1) no attribute should be missing;
(2) all prices and volumes should be non-negative.

Let violations of these constraints at time t be denoted by $g_1^1(t), g_2^1(t)$, where the superscript denotes glitches of Type 1, and the subscript indexes the glitches as 1 and 2 respectively.

We also aggregate these over all NYSE stocks to compute the total glitches of these types at time t as follows:

$$G_1^1(t) = \sum_{NYSE} g_1^1(t)$$

and

$$G_2^1(t) = \sum_{NYSE} g_2^1(t)$$

respectively, where the summation is over all stocks and t is indexed over the trading minutes. We will use these sums later to compute statistical distortion.

Since the data was obtained from an authoritative source, we could not find any obvious glitches of Type 1. This is not surprising since Type 1 and Type 2 glitches are the easiest to detect (do not require accessing other rows/entities). They are often the ones that are cheaply remedied and hence taken care of by data providers.

3.4.2 Type 2 Constraints

Domain experts often stipulate conditions on inter-relationships between attributes. For example, for a given stock S at time t:
(1) the closing price C_t should be contained in the interval $[L_t, H_t]$ where L_t is the lowest price observed during that minute t, and H_t is the highest.
(2) If the closing price C_t is present, then volume V_t should be present. While this might seem redundant to the first Type 1 constraint mentioned above, it addresses a specific aspect of the process: volume is determined only after the close of the trading minute.
These are examples of Type 2 constraints. Using notation introduced above, let the violations of these two constraints be denoted by $g_1^2(t)$ and $g_2^2(t)$, and the aggregates across all stocks be denoted by

$$G_1^2(t) = \sum_{NYSE} g_1^2(t)$$

and

$$G_2^2(t) = \sum_{NYSE} g_2^2(t)$$

respectively. We could not find Type 2 glitches in this particular data stream that violated the constraints specified above, because, as mentioned earlier, they are cheaply detected and remedied. Since we could not find a Type 1 or Type 2 constraint violation using the TR guidelines, in order to demonstrate a Type 1 constraint, we applied a rule of thumb that states that blue chip stocks need to trade every minute. Since this type of glitch falls out naturally as a Type 3 glitch by learning the number of trades for each stock, we merely demonstrate it here to exemplify a Type 1 constraint violation. There are 390 minutes in a normal trading day, starting at 9:30 AM and ending at 15:59 PM. Sometimes there is an additional 391^{st} minute, a spillover record at the end of the day stamped 16:00 to report the overflow from the previous minute. So we could stipulate a simple Type 1 constraint on the daily totals: "every day should have at least 390 minute files". Figure 4 depicts daily total of trading minutes (corresponds to the number of files received) of the stock HAL. It usually fluctuates between 390 and 391.

There are noticeable violations on the days before Thanksgiving and July 4 respectively. This is expected since the stock exchange closes early on these days. There were less noticeable gaps on May 23, May 25 and June 6 when there were 3, 2 and 20 missing minutes. All these were detected as outlying "lags" by FIT, as a part of Type 3 glitch detection where the model expected the lag between successive trades to be one minute. For a heavily traded stock like HAL, gaps in trading are very unusual. The missing trading minutes on May 23, May 25 and June 6 are deemed data glitches (missing/incomplete data) since we could not find any explanation for them using readily available domain knowledge such as holiday trading patterns.

However, the most interesting missing data detected by FIT, but not by mere counting of total minutes, was on August 13. The total number of trading minutes were 390, a very common and acceptable value, hence not a violation of the Type 1 constraint stipulated above. However, FIT flagged a missing minute. Closer examination showed that the reason FIT generated a lag alarm was because of the missing minute at 13:54 causing an unusual lag of more than a minute in data reporting.

```
2012-08-13 13:52,13,34.76,34.81,34.76,34.81,16704
2012-08-13 13:53,13,34.8,34.8,34.78,34.79,5913
2012-08-13 13:55,13,34.79,34.82,34.79,34.8,14644
```

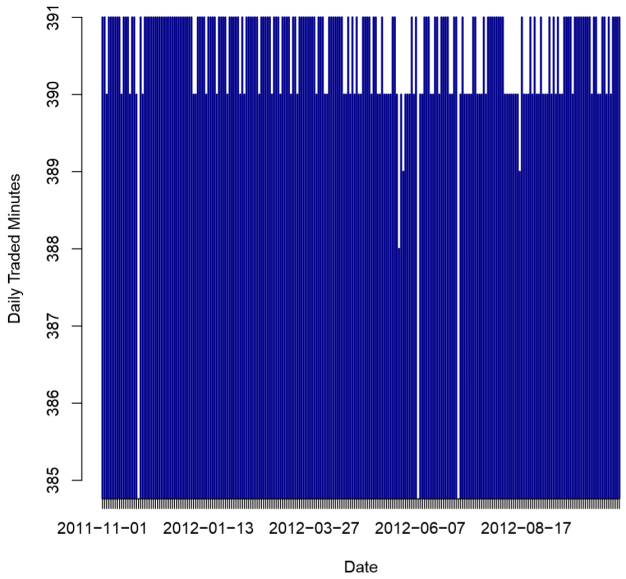

Figure 4: *The total daily trading minutes (corresponds to the number of files received) of the StockID 968 (HAL). It usually fluctuates between 390 and 391. There are noticeable gaps on the days before Thanksgiving and July 4 respectively. This is expected since the stock exchange closes early on these days. There were less noticeable gaps on May 23, May 25 and June 6 when there were 3, 2 and 20 missing minutes. All these were detected as outlying "lags" by FIT (Feed Inspection Tool), as a part of Type 3 glitch detection where the model expected the lag between successive trades to be one minute.*

```
2012-08-13 13:56,13,34.8,34.8,34.78,34.79,9154
```

The total counts remained at 390 due to the presence of the spillover record at 16:00.

```
2012-08-13 15:59,13,35.01,35.04,35,35.03,191139
2012-08-13 16:00,13,35.03,35.03,35.03,35.03,100
```

Thus, FIT provides a mechanism for detecting potentially missing records at the minute level, that could be overlooked by monitoring aggregates. This is an example where a simple Type 1 constraint would have failed, while a Type 3 constraint that takes into account the history of inter-trading lags catches the missing record.

3.4.3 Type 3 Constraints

Next, we validate Type 3 constraints. Because Type 3 and Type 4 constraints capture more interactions and interrelationships in the data, they tend to reveal complex glitches that might not be obvious. We expect the values of the high, low, open and close prices, and volume, to be within certain statistical ranges for each of the stocks. Similarly, derived variables such as the lag between successive trades LG_t, the price spread SP_t, and consecutive changes, e.g., for the high H_t, $DL_t = H_t - H_{t'}$ should all fall within observed statistical ranges.

We compute the expected behavior in two ways. The first, as expressed by [11], as a fixed percentage of the previous value. And second, using the statistical models employed by FIT. These are Type 3 constraints because they use multiple entities but within a single attribute. We articulated eight such constraints, one each for the five variables H_t, L_t, O_t, C_t, V_t and one for each of the derived variables SP_t, DL_t, LG_t. Note that in principle we could have derived variables for each of the original five variables, but for purposes of illustration we focus on

the price spread SP_t, the trading lag LG_t, and the change (delta) DL_t in successive values of the intra-minute high, H_t.

Therefore, there could be eight possible glitches of Type 3,

$$g_i^3(t), i = 1, \ldots, 8,$$

and the aggregates across all stocks and all types are given by:

$$G^3(t) = \sum_i G_i^3(t) = \sum_i \sum_{NYSE} g_i^3(t), i = 1, \ldots, 8.$$

The total of Type 3 glitches, $G^3(t)$, (since Type1 and Type 2 errors are all zero) is a simple measure of *statistical distortion*, the disparity between what we expect and what we actually observe. That is,

$$SD(t) = G^3(t).$$

The panel on the right in Figure 2 shows the results of running FIT on the total of Type 3 glitch counts, $G^3(t)$. Note that the inherent noise in the streams ensures that at any given time there will be a certain number of glitches. We check if the observed number of glitches are *above or below* the statistical variation. It is important to identify too few glitches too because that might indicate a drastic change, e.g., a system has malfunctioned and is sending bogus records, e.g., "price flats" where there is no variation at all. Here, the FIT model takes into account the hour of day. Big pink dots represent the morning hours, gradually changing into smaller blue dots as the day goes by. The red dots are anomalous values. There were several days that featured a high number of glitches. These potentially correspond to global events that affected a large number of stocks, like the Fed decision on September 12.

3.4.4 Type 4 Constraints

Finally, we cross reference with an independent data source to validate our data. We used the daily stock data available on Yahoo to validate the intra-minute high, H_t, on stocks. We formulate the following Type 4 constraint: "The highest price from the daily Yahoo file should be the same as the maximum of all the intra-minute highs in the NYSE minute bar file". The bar chart on the left panel of Figure 5 shows the distribution of the number of stocks that violated that constraint on a given number of days. The tallest blue bar shows that there were around 1100 stocks that matched the constraint almost every day during the observation period of over 250 days. The red bars on the right show stocks that violated that constraint on most days. The panel on the right shows an example stock of each of these types and plots the daily high against the maximum intra-minute high from the minute files. The blue stock is perfectly linear as expected, while the red stock's values are spread out including a very suspicious "price flat" indicative of a spurious interpolation of missing values.

Given the price flats in daily file, we suspect the NYSE data has better quality than the independent source.

3.5 Glitch Prone Stocks

Next, we defined the statistical distortion corresponding to a given stock S to be the proportion of glitches of all types in the entire stream corresponding to that stock. That is,

$$SD(S) = \frac{\sum_i G^i(S)}{N}, i = 1, \ldots, 4$$

where N=the total number of possible glitches. Note the implicit summation over time t. In the simplest case, if there are K data elements (for example, 390 traded minutes on each of 280 days described by 7 attributes would yield 764,400 data elements), and if each of these could have a maximum of C glitches associated with it, then

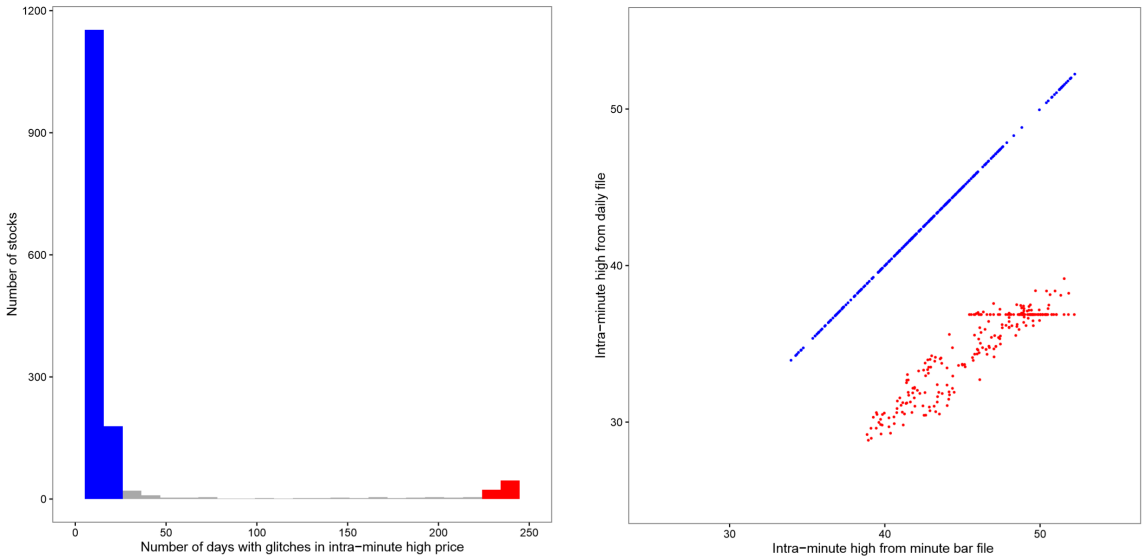

Figure 5: *We validated our data using an independent, external data source and formulating the following Type 4 constraint: "The highest price from the daily Yahoo file should be the same as the maximum of all the intra-minute highs in the NYSE minute bar file". The bar chart in the left panel shows the distribution of the number of stocks that violated a Type 4 constraint on a given number of days on the X-axis. The tallest blue bar shows that there were around 1100 stocks that matched the constraint almost every day during the observation period of over 250 days. The red bars on the right show stocks that violated that constraint on most days. The panel on the right shows an example stock of each of these types and plots the daily high against the maximum intra-minute high from the minute files. The blue stock is perfectly linear as expected, while the red stock's values are spread out including a very suspicious "price flat" indicative of a spurious interpolation of missing values in the external source.*

$N = K * C$. For example, if $C = 2$ (e.g., where the possible glitches are: missing OR duplicate OR outlier OR (duplicate AND outlier)), then $N = 1,528,800$ for the stock S.

We sorted the individual stocks by the statistical distortion during the period of observation. Figure 6, left panel, shows a plot of the glitch proportion (percentage) for individual stocks. Note the elbow in the curve at 3.5%. Based on this empirical observation, it is reasonable to consider the quality of data for stocks with statistical distortion greater than 3.5% to be suspect. The worst stock, with a statistical distortion of almost 7%, is represented by a purple dot at the top, its StockID given by 1881. The "cleanest" stock is shown by a gold dot at the bottom of the curve with a glitch proportion of less than 2%, with a StockID of 968, which we know from our previous studies to be Halliburton. The distributions of glitches in these two stocks are shown in Figure 6, right panel. The box plot for the purple stock 1881 is wider, indicating more glitched data, as well as more glitches in each minute indicating that there were more corrupt attributes per minute as well. Stock 968 on the other hand, when glitchy, had in general fewer glitches per minute, typically just 1.

In Figure 7, we show a typical day for each of these two stocks, for intra-minute high price, H_t. The cyan dots represent Thomson Reuters anomalies, while red dots represent FIT anomalies. For TR, we selected a percentage that yielded a number of anomalies comparable to FIT, since the actual percentage threshold changes from stock to stock. The black solid line represents the ideal D_t^I as computed by the streaming FIT model. The blue line represents observed values. In general, the TR method creates more alarms since it is based only on the previous value and a fixed percentage threshold. The FIT model used here is based on a window of 25 values, by hour, hence more robust but also more sensitive since it is customized to each hour.

Note that 1881 is a penny stock and hence any change is bound to be large on a percentage basis as evidenced

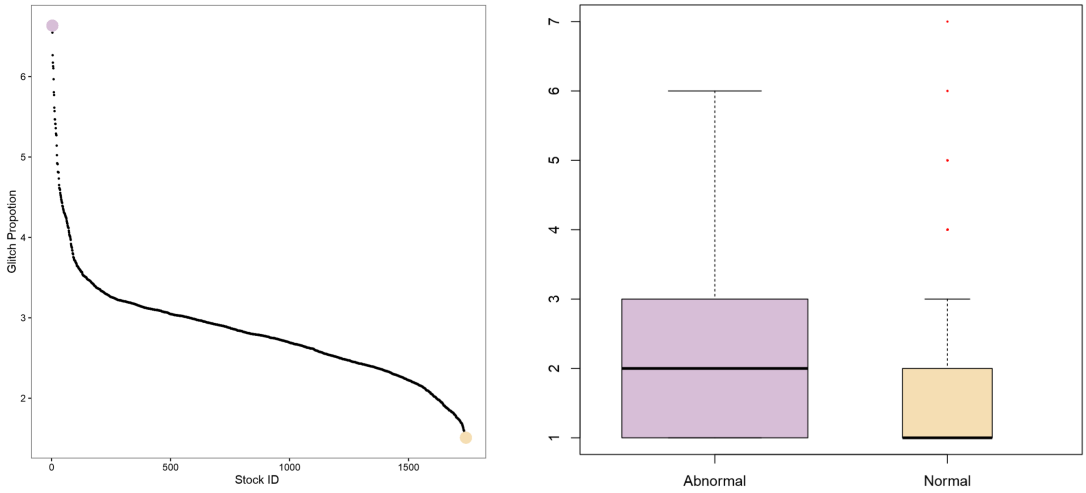

Figure 6: *Individual stocks sorted by the amount of statistical distortion as measured by proportion of glitches during the period of observation. Note the elbow in the curve at 3.5%: stocks with glitch proportion greater than 3.5% are suspect. The worst stock, with a statistical distortion of almost 7%, is represented by a purple dot at the top, Stock ID 1881. The "cleanest" stock is shown by a gold dot at the bottom of the curve with a glitch proportion of less than 2%, StockID 968. The distributions of glitches in these two stocks are shown in the right panel. The box plot for the purple stock 1881 is wider, indicating more glitched data, as well as more glitches in each minute indicating that there were more corrupt attributes per minute as well. Stock 968 on the other hand, when glitchy, had in general fewer glitches per minute, typically just 1.*

by TR's cyan dots. However, FIT alerts less often since it is based on statistical variation and picks up only significant changes. Stock 968 exhibits great volatility in the beginning of the plot. FIT considers this normal for the stock and does not alert while the TR method generates numerous alerts based on percentage change. Toward the middle, there is much less volatility and FIT identifies anomalies in an appropriate fashion even though the variability is smaller in absolute terms as compared to the beginning.

4 Conclusion

In this paper, we presented a framework for monitoring data quality in dynamic temporal streams. We employ four types of constraints in increasing order of data dependence, to define data quality problems or glitches. We use FIT, a statistical stream monitoring tool, to detect the glitches in near real time, capturing instances where current data D_t varies statistically from the ideal D_t^I computed in a data-driven fashion by FIT using historical data. We use *statistical distortion* to measure the disparity between the ideal and observed by tracking the proportion of glitches across stocks and within stocks. Our case study of financial data streams of 2000 NYSE stocks during the period November 2011 to November 2012 revealed interesting data quality phenomena some of which could be explained by our domain expert.

5 Acknowledgments

We would like to thank Kumar Doraiswami for donating a year's worth of NYSE minute bar data, and for being our domain expert, and Simon Urbanek for permitting us to annotate and use the underlying plot in Figure 1.

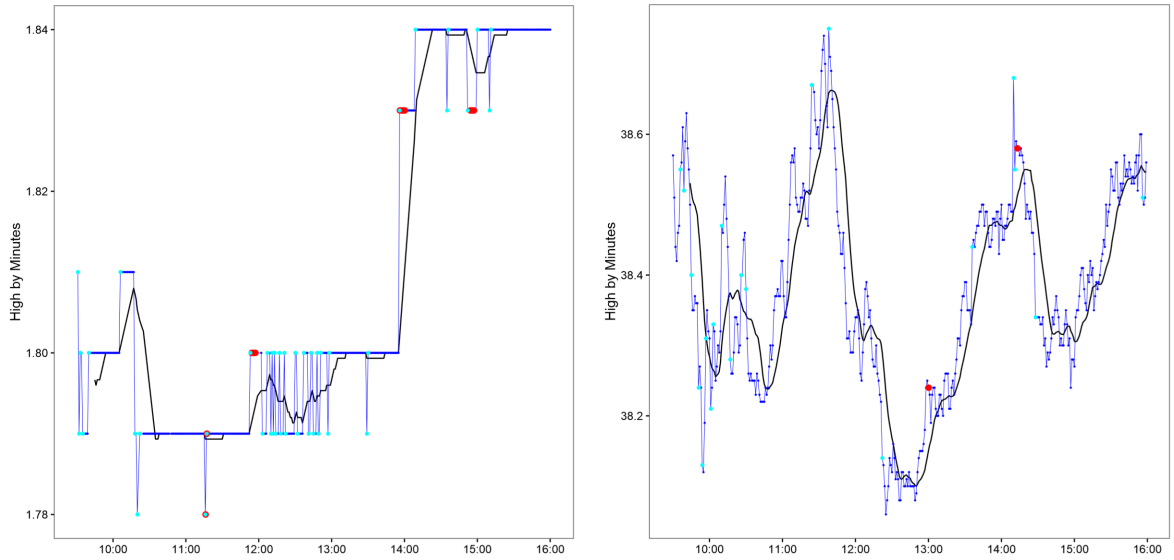

Figure 7: *A typical day for each of the stocks 1881 and 968, for the attribute H_t, the intra-minute high. The cyan dots represent Thomson Reuters anomalies, while red dots represent FIT anomalies. For TR, we selected a percentage that yielded a number of anomalies comparable to FIT, since the actual percentage threshold changes from stock to stock. The black solid line represents the ideal D_t^I as computed by the streaming FIT model. The blue line represents observed values. In general, the TR method generates more alarms since it is based only on the previous value and a fixed percentage threshold. The FIT model used here is based on a window of 25 days, hence more robust but also more sensitive since it is customized to each hour.*

References

[1] Z. Abedjan, L. Golab and F. Naumann, *Profiling relational data: a survey*, VLDB, 2015.

[2] L. Berti-Equille, T. Dasu and D. Srivastava, *Discovery of Complex Glitch Patterns: A Novel Approach to Quantitative Data Cleaning*, ICDE 2011.

[3] T. Johnson and T. Dasu, *Bellman: A Data Quality Browser*, 2001.

[4] L. Golab, H. J. Karloff, F. Korn, A. Saha and D. Srivastava, *Sequential Dependencies*, VLDB, 2009.

[5] S. Ginsburg and R. Hull, *Order Dependency in the Relational Model*, Theor. Comput. Sci., 26, 1983.

[6] S. Song, A. Zhang, J. Wang and P. S. Yu, *SCREEN: Stream Data Cleaning under Speed Constraints*, Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 827–841, 2015.

[7] Z. Abedjan, C. G. Akcora, M. Ouzzani, P. Papotti and M. Stonebraker, *Temporal Rules Discovery for Web Data Cleaning*, PVLDB, 9, 4, 336–347, 2015.

[8] T. Dasu and J. M. Loh, *Statistical Distortion: Consequences of Data Cleaning*, PVLDB, 2012.

[9] T. Dasu, J. M. Loh and D. Srivastava, *Empirical Glitch Explanations*, KDD, 2014.

[10] T. Dasu, V. Shkapenyuk, D. Srivastava, and D. F. Swayne, *FIT: Feed Inspection Tool for Data Streams*, PVLDB, 2015.

[11] Thomson Reuters, *Thomson Reuters Datastream: Data Quality Assurance*, 2010.