FIT to Monitor Feed Quality

Tamraparni Dasu
AT&T Labs–Research
tamr@research.att.com

Vladislav Shkapenyuk
AT&T Labs–Research
vshkap@research.att.com

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

Deborah F. Swayne
AT&T Labs–Research
dfs@research.att.com

ABSTRACT

While there has been significant focus on collecting and managing data feeds, it is only now that attention is turning to their quality. In this paper, we propose a principled approach to online data quality monitoring in a dynamic feed environment. Our goal is to alert quickly when feed behavior deviates from expectations.

We make contributions in two distinct directions. First, we propose novel enhancements to permit a publish-subscribe approach to incorporate data quality modules into the DFMS architecture. Second, we propose novel temporal extensions to standard statistical techniques to adapt them to online feed monitoring for outlier detection and alert generation at *multiple scales* along three dimensions: aggregation at multiple time intervals to detect at varying levels of sensitivity; multiple lengths of data history for varying the speed at which models adapt to change; and multiple levels of monitoring delay to address lagged data arrival.

FIT, or Feed Inspection Tool, is the result of a successful implementation of our approach. We present several case studies outlining the effective deployment of FIT in real applications along with user testimonials.

1. INTRODUCTION

Data are being collected and analyzed today at an unprecedented scale, and organizations routinely make important decisions based on data stored in their databases. However, with the huge volume of generated data, the fast velocity of arriving data, and the large variety of heterogeneous data, the veracity (or quality) of data in databases is far from perfect [20, 14]. Data errors (or glitches) in many domains, such as medicine, finance, law enforcement, and telecommunications, can have severe consequences. This highlights the pressing need to develop data quality management systems to effectively detect and correct glitches in the data, thereby adding veracity and value to business processes.

Data errors can arise throughout the data lifecycle, from data entry, through storage, data integration, analysis, and decision making [10]. Much of the data quality effort in the database research community has focused on detecting and correcting errors in data once the data has been collected in a database [5, 14], or during the data integration process [13]. While commercial tools provide capabilities to perform record-level data quality checks and data cleansing during a batch ETL (extract, transform, load) process (see, e.g., [2, 1, 4, 3]), there has been little attention paid to data quality in the context of continuous feeds that are so critical in today's data management systems. This is surprising since data entry time offers the first opportunity to detect and correct errors [8]. We address this problem in our paper, describe principled techniques for online data quality monitoring in a dynamic feed environment, and present case studies based on the deployment of FIT (Feed Inspection Tool) in real applications.

Data feed management systems (DFMSs) have recently emerged to provide reliable, continuous data delivery to databases and data intensive applications that need to perform real-time correlation and analysis [21, 17]. In particular, in prior work we have presented the Bistro DFMS [21], which is currently deployed at AT&T Labs and is responsible for the real-time delivery of over 100 different raw feeds, distributing data to several large-scale stream warehouses. Bistro uses a publish-subscribe architecture to efficiently process incoming data (real-time streams, periodic and ad hoc data) from a large number of data publishers, identify logical data feeds (based on a flexible specification language), and reliably distribute these feeds to remote subscribers. FIT naturally fits into this DFMS architecture, both as a subscriber of data and metadata feeds, and as a publisher of learned statistical models and identified outliers. However, the FIT system architecture is DFMS-agnostic and can be used in conjunction with any DFMS as long as it implements a basic publish-subscribe interface.

FIT performs continuous, *passive* monitoring of the feeds, so as to not introduce any delays in real-time applications that correlate and analyze the data. However, early detection of errors by FIT enables data administrators to quickly remedy any problems with the incoming feeds, and inform data analysts of any potential issues with the newly arrived data. FIT's online feed monitoring can naturally detect errors from two distinct perspectives: (i) errors in the data feed processes (e.g., missing or delayed delivery of files in a feed), and (ii) significant changes in distributions in the data records present in the feeds (e.g., erroneously switching from packets/second to bytes/second in a measurement feed). The former is achieved by continuously analyzing the

Proceedings of the VLDB Endowment, Vol. 8, No. 12
Copyright 2015 VLDB Endowment 2150-8097/15/08.

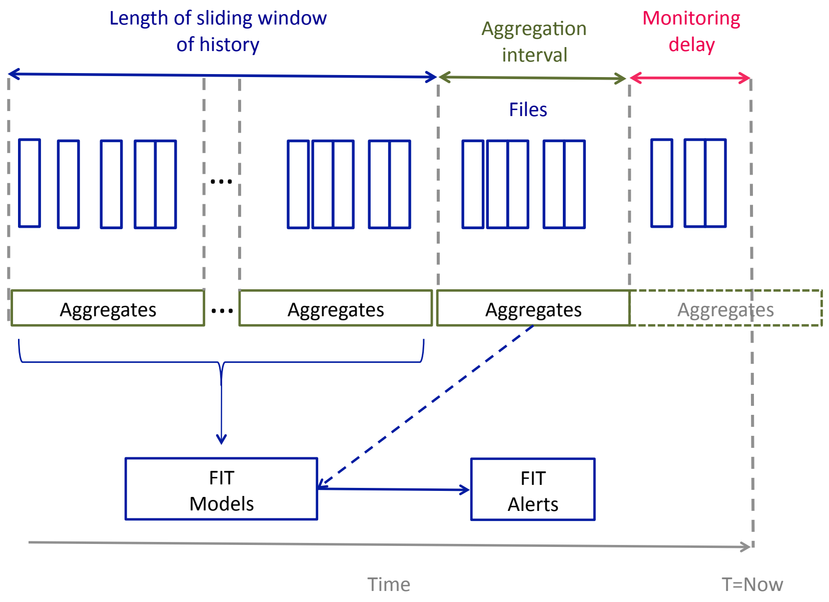


Figure 1: Monitoring feeds at multiple temporal scales enables us to control sensitivity (aggregation interval), adaptability (amount of history) and stability (monitoring delay) of the feed monitoring process.

DFMS metadata feed, while the latter is achieved by continuously analyzing the contents of the data feeds.

For reasons of scalability and interpretability, FIT builds simple, non-parametric statistical models over the most recently seen data (identified by a sliding window) to predict future trends and identify outliers as significant deviations from the predictions. To ensure statistical robustness, these models are built over time-interval aggregated data rather than point-wise data. While useful, these standard outlier detection techniques from the statistics community needed to be made considerably more flexible to account for the variability in data feeds during normal operation, so as to avoid raising unnecessary alerts and incorporate user provided feedback on the raised alerts. For this purpose, FIT detects outliers at *multi-scale*, in three different dimensions, as illustrated in Figure 1.

The first dimension is the *aggregation time interval*, which determines the granularity at which errors can be detected. A short time interval allows for detection of fine-granularity errors, but introduces considerable noise (i.e., variance) into the process. A long aggregation time interval allows for robust predictions, but may mask compensating errors (e.g., fewer files in one time unit and more files in the next time unit, within the same time interval). Detecting outliers using multiple aggregation time intervals enables FIT to effectively deal with this issue.

The second dimension is the *sliding window length*, which determines the extent of history used to build the predictive model. A long window would not allow FIT to quickly identify new errors, while a short window could lead to normal fluctuations being detected as outliers. Detecting outliers using multiple sliding window lengths enables FIT to effectively deal with this issue.

The third dimension is the *monitoring time delay*, which is used to address lagged data arrival. A short monitoring time delay allows FIT to quickly compare the model prediction with the (aggregated) observation, but does not account for normal variability in feed delivery schedules. A long monitoring time delay ensures that late feed arrivals are accounted for, but may sometimes be too late for an administrator to take remedial actions. Detecting outliers using multiple monitoring time delays enables FIT to effectively deal with this issue.

Enabling FIT to effectively monitor multiple feeds continuously and detect outliers at multiple scales necessitates the sampling of data feeds, especially voluminous, high velocity feeds. While traditional sampling is performed at the record level, this cannot be efficiently done on data feeds, since it would require parsing the content of *all* the files in the feed into records to extract the sampled records. For efficiency, FIT samples files from a data feed, then parses and analyzes all the records in the sampled files. We empirically show that this procedure provides similar robustness to record level sampling in practice. To support this functionality, we enhanced the DFMS to be able to efficiently create derived feeds with sampled files, based on file level metadata.

The rest of the paper is organized as follows. Section 2 presents related work. The infrastructure for monitoring feed quality is described in Section 3. Section 4 discusses the architecture and novel statistical modules in FIT. Sections 5, 6 and 7 present a variety of case studies, based on the deployment of FIT in real applications, to demonstrate the flexibility and effectiveness of the FIT approach to online data quality monitoring of dynamic feeds. Note that no personally identifiable information (PII) was gathered or used in conducting these studies. To the extent any data was analyzed, it was anonymous and/or aggregated data. Finally, Section 8 concludes with a summary of the findings discussed in the paper.

2. RELATED WORK

Data feed management systems (DFMSs) have recently emerged as a way for organizations to manage reliable, real-time distribution of high-volume data feeds to interested applications for correlation and analysis [21, 17]. The AsterixDB Big Data Management System (BDMS) [17] contains a feed management subsystem responsible for data ingestion. The focus of the AsterixDB feed manager is on scalable data ingestion using partitioned parallelism and fault-tolerance in the presence of input failures. The Bistro DFMS [21] developed at AT&T Labs-Research focuses on several aspects of feed management, such as efficient processing of source data feeds, mapping of source data feeds to logical consumer feeds using a flexible specification language, and efficient real-time file delivery to intermittently connected remote subscribers. Bistro also provides limited support for detecting feed changes, new feeds, dropped feeds and lost data in already defined feeds. FIT greatly expands on these capabilities by moving the data quality analysis into a separate system.

Data quality has been an active area of investigation from different perspectives, including definition and detection of data glitches, metrics for measuring them, and methods for cleaning. The database community is typically focused on semantic and domain specific integrity expressed in the form of logical constraints, whereas the statistical community has developed a body of work that focuses on capturing the statistical properties of the data and investigating departures from expected statistical distributions.

Recent work, such as [15], relies on constraint specification to identify potentially problematic data, and [16] investigates subsets or "pattern tableaux" that meet or fail constraints [16]. Statistical approaches to data quality, particularly metrics, include the masking index [6] and statistical distortion [11]. Data cleaning and repairs have also been the focus, e.g., in [23] which addresses continuous cleaning of dynamic data by focusing on data semantics, integrity constraints, and the history of repairs simultaneously. Recently, there has been an interest in going beyond detection and repairs, to explain glitches in order to reduce the amount of cleaning-induced distortion in the data [12]. Finally, statistical approaches to data quality have frequently been adapted from process control methods. While we use a relatively simple approach in our models, there has been considerable work in the area of nonparametric methods for process control, summarized nicely in [7].

Many commercial data integration systems such Informatica PowerCenter [2], IBM InfoSphere DataStage [1], SAP BusinessObjects Data Services [4] and Oracle Data Integrator [3] include ETL (Extract, Transform, Load) tools with capabilities to perform record-level data quality checks and data cleansing on data that is about to enter an integrated data warehouse. These tools work in a batch mode for periodic refreshes of data warehouses and do not include support for performing ETL operations in a streaming fashion. In comparison, our focus with FIT is on monitoring of highly dynamic continuous data feeds and detecting errors before they have a chance to propagate to data warehouses or other applications.

The importance of early detection and correction of data errors has long been recognized by the research and industrial communities with a lot of effort put into design of user-interface tools that minimize the possibility of invalid data entry. Examples include defining integrity constraints on individual field values and rejecting data forms that violate these constraints; restricting invalid data entry through usage of check-boxes, radio-boxes and combo-boxes; and auto-completion of field values using a variety of prediction methods. The Usher system [8] uses a probabilistic model of previous form submissions to optimize form design, dynamically adapt data entry and perform after-entry verification to increase the quality of user-entered data. The FIT system shares the focus of early error detection but is primarily concerned with data quality issues in software-generated continuous feeds. Additionally, we do not focus on record-level quality checks but rather on detecting significant changes and anomalies in data distributions in data feeds.

3. FEED QUALITY INFRASTRUCTURE

Data Feed Management Systems (DFMSs) provide a scalable platform for the reliable delivery of continuous data feeds to a variety of data intensive applications, such as databases and streaming data warehouses. In prior work, we described the architecture of the Bistro DFMS [21], deployed at AT&T Labs, that is responsible for the real-time delivery of about 100 different feeds. Since then, we have deployed a second DFMS called the Data Router, which differs from Bistro in its low-level system architecture but which shares a very similar publish-subscribe interface.

FIT can be used with either of these systems, because we chose to design an independent feed quality monitoring tool rather than embedding it within either DFMS. This passive approach has a number of advantages:

- FIT can work with various DFMSs as long as they implement the basic publish-subscribe interface.
- FIT does not require changes in the DFMS to support feed quality monitoring, which simplifies its operational deployment.
- FIT does not introduce processing delays into real-time feed delivery, even if the data quality analysis is computationally expensive.

In this section we describe the high-level architecture of these DFMSs and show how the FIT system slots naturally into that architecture.

3.1 DFMS Architecture

At a very high level, a Data Feed Management System is a publish-subscribe system responsible for routing source data streams generated by feed producers to distributed feed subscribers. Publishers deliver source data feeds as a stream of raw files. Many of these data feeds involve multiple sub-feeds, are highly aggregate and complex, and are not directly usable by subscribers in their source form. In those cases, the DFMS administrator uses a flexible feed definition language to disaggregate the source feeds into their constituent files and create logical consumer feeds that better match the needs of subscribers.

In Figure 2 we show a high-level picture of how data feeds flow through the DFMS. Publishers deliver data feed files to the DFMS with each file explicitly labeled as belonging to one source feed. Upon the receipt of a new file, the DFMS File Classifier matches it to one of the logical consumer feeds and stages it for delivery to all interested subscribers. Files

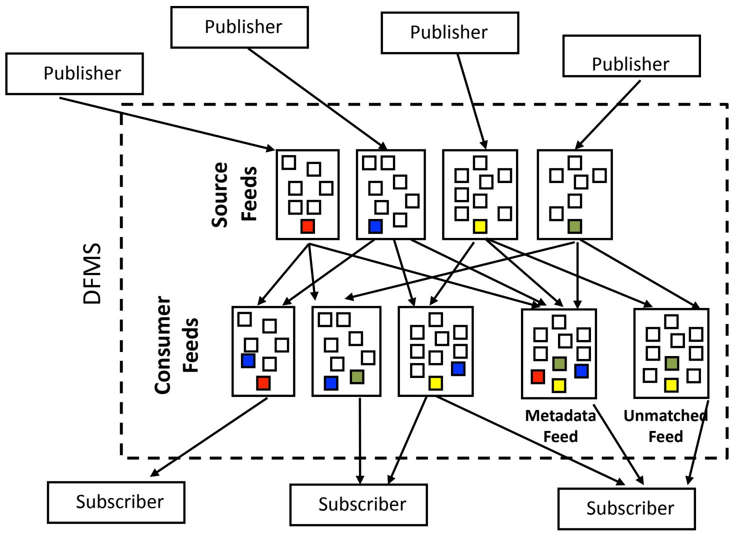


Figure 2: The DQ feed manager may reorganize incoming files in the process of moving them from a source feed to a consumer feed. The nested boxes represent files within feeds.

that do not match any of the defined logical consumer feeds are placed in a special Unmatched Feed that can also be subscribed to. Traditional subscribers are typically not interested in the Unmatched Feed, but FIT makes use of it to identify feed anomalies as described in Section 6.

The interpretation of data feeds can be very difficult in the absence of file metadata such as file formats, data schemas, timestamps, etc., and many feeds are poorly documented despite the general trend towards self-documenting data formats. Bistro and the Data Router support file metadata as a first class citizen and allow cooperating producers to attach metadata to all the files posted to source data feeds. Even when metadata has not been attached, the name of the file often contains useful information, and we have implemented an automated extraction mechanism that extracts file name metadata. Both explicit and automatically extracted metadata are posted into the Metadata Feed which can be consumed both by regular subscribers, to add to their understanding of the incoming files, and by data quality tools like FIT to perform outlier detection.

The DFMS Feed Delivery Subsystem is responsible for the final step in data feed processing – the scalable delivery of logical consumer feeds to subscribers using a variety of supported protocols, such as SCP, SFTP and HTTP. Subscribers can choose to receive every file in a logical consumer feed or a configurable sample of those files in case the full feed rate provides more files than the subscriber can handle. Our configuration language is flexible enough to define samples in two ways: (a) a random sample based on a hash of the metadata fields, or (b) a longitudinal (panel) sample created by hashing on selected file metadata fields. FIT can take advantage of either sampling strategy to reduce the cost of file content analysis while still maintaining reasonable accuracy. We discuss the accuracy of data quality analysis on sampled feed data in Section 7.

3.2 DFMS-FIT Integration

As previously mentioned, the FIT data quality monitor fits naturally into the DFMS architecture and utilizes the publish-subscribe interface to interact with the rest of the system. FIT acts like a regular subscriber to receive the Metadata Feed and any other selected feeds, either in their entirety or sampled. Moreover, FIT can also act in the role of publisher. Rather than providing an interface for applications to query feed quality information, FIT posts the results of data quality analysis back into the DFMS using predefined data quality feeds. This approach allows us to layer a variety of different data quality applications such as visualizers, alerting applications, data cleaners and others on top of the output produced by FIT; it also allows us to share FIT output with other subscribers who might want to design their own plots or alerts.

For each logical consumer data feed F registered in the DFMS, we define a number of special data feeds that carry data quality information. In this section we describe all data quality feeds that we maintain for this purpose.

1. *Multi-scale temporal aggregates,* $A(F)$: FIT generates temporal aggregates for multiple *aggregation intervals*. This allows FIT to monitor feeds at several scales to detect problems that might only show up at one particular level of aggregation. $A(F)$ contains summary statistics and signatures that could be useful to other subscribers for generating feed reports.

2. *Multi-scale FIT model parameters,* $M(F)$: FIT builds models $M(F)$ at multiple scales by using historical data of aggregates $A(F)$ in *sliding windows* of different *lengths*. We discuss models generated by FIT in more detail in Section 4.3.

3. *Multi-scale FIT outliers,* $E(F)$: FIT tests the most recent set of aggregates $A(F)$ against the appropriate

model parameters in $M(F)$ and generates data outliers $E(F)$ when the feed behavior deviates from the expected behavior. The outliers are generated for different *monitoring time delays* to allow for minor variations in data arrival.

4. *DQ metrics, $DQ(F)$*: FIT uses the outliers $E(F)$ as the basis for DQ metrics related to missing or incomplete data, the number of alerts and their severity, and the proportion of alerts out of the number tested.

In Figure 3 at left, we show the flow of data feeds and associated quality feeds between publishers and subscribers via the DFMS.

4. FEED INSPECTION TOOL

As mentioned in Section 1, FIT's objective is to detect anomalous data from two distinct perspectives: anomalies in *data gathering* detected from feed metadata and anomalies in *data measurement* based on file contents. FIT uses *summaries* of metadata, e.g., file counts, average file size and average inter-arrival times, and also descriptive statistics of file content, e.g., trimmed mean or median of various attributes, to build models of feed behavior and detect anomalies. By focusing on aggregates, FIT ensures speed as well as robustness of models and results. We describe FIT in detail in this section.

FIT is implemented in the R language [19] to take advantage of R's rich library of statistical functions.

4.1 FIT Architecture

The FIT architecture is illustrated in Figure 3 (right). In the Data Module, FIT acquires data, formats and aggregates it at one or more temporal scales; FIT's Analysis Module reads aggregated data and generates model statistics and outliers for each level of aggregation. The role of the Alerting Module is to combine the output of the models with user requirements to prepare human-readable reports.

FIT is customarily run in partnership with a DFMS as described in Section 3.2, but it is also possible to run it independently. The Data Module always has to be tailored in some ways to the data to be processed, and part of that tailoring includes defining the method by which the data is acquired. The results produced by the Data Module and the Analysis Module can either be published to a DFMS or stored locally (or both); in either case, they are readily accessible to the Alerting Module, Visualizer, and Email Alerter. We now describe each module in more detail.

4.2 Data Module: Multi-scale Aggregation

The first step in FIT's data pipeline is to acquire the data. It may have subscribed to the data in a Data Feed Manager, in which case it is delivered to FIT, or it may pull it from a web site or other source at regular intervals. The data may consist solely of feed metadata or it may include some or all of the data files that comprise the stream.

In either case, FIT identifies two sets of variables: the *group-by variables* (categorical variables to be used for grouping, such as a time interval, source, and record type) and the *test variables* to be summarized and tested for anomalies (such as number of files, file size, and inter-arrival time). These variables may be present in the data or derived from the data, and could be of different types including numeric, string and timestamps.

While handling of numerical attributes is obvious, there are several approaches to handling other types of variables as well. One approach to handling string variables is to estimate the distribution of the lengths of the variable and identify unexpected string lengths. This method is commonly used when the string has fixed length such as a US telephone number. For more varied string lengths, lexicographic distributions of string variables are used to identify strings that are far from frequently occurring values. For example, if a file is typically named

```
FEEDNAME : FILENAME : OTHERNAME : TIMESTAMP.TYPE
```

then departures from this convention are identified by computing the lexicographic permutation distance of the file name attribute. Common data quality issues include an undocumented appending of a single character at the end of the filename or feedname that could potentially lead to mismatched configurations, resulting in dropped files. Similarly, timestamps could be converted to time intervals and analyzed for unexpected values such as data arriving from the future (negative duration) or files timestamped prior to the existence of the physical process that created the data.

FIT's Data Module aggregates data based on different aggregation intervals, computing statistical summaries (measures of centrality such as mean, trimmed mean and median; measures of dispersion such as standard deviation and MAD (Median Absolute Deviation); quantiles) for the quantitative variables for each combination of group-by variables.

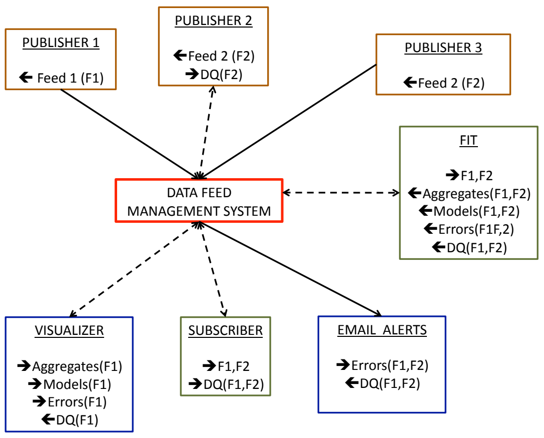
The granularity of aggregation often determines the sensitivity of statistical models and outlier detection. Multi-scale aggregation is important because applications have individual needs. Some focus on transient but potentially catastrophic outliers that can be captured only at finer levels of temporal aggregation, e.g., 5 minutes, while others are interested in systematic issues that persist even after aggregation over longer intervals such as hours and days. FIT subscribers have typically found aggregation intervals between 30 minutes to 3 hours to be most useful. We will discuss this aspect in detail in Section 6.

The Data Module is the publisher of the multi-scale temporal aggregates, $A(F)$. These aggregates are the key input to the next stage, but they are independently interesting. FIT subscribers can use them in reports, to generate feed signatures, or to create plots (see, e.g., Figure 5).

4.3 Analysis Module: Multi-scale History

The Analysis Module takes as input the aggregates $A(F)$ created by the Data Module, builds models and identifies data outliers. The models are *unsupervised* and built based on different amounts of historical data (sliding window sizes) in order to capture outliers at multiple scales–longer history implies models evolve more slowly while shorter histories capture local movement and detect outliers with respect to the most recent behavior of the data. In the absence of historical data, it is possible to specify the initial FIT models based on domain knowledge or other *a priori* specification.

Note that FIT creates a model for each combination of: (i) monitoring delay, (ii) level of temporal aggregation, (iii) historical window size, and (iv) values of group-by variables, effectively creating a temporal stream of models and corresponding outliers. By building and publishing multiple models at each update, FIT permits the subscriber to dy-



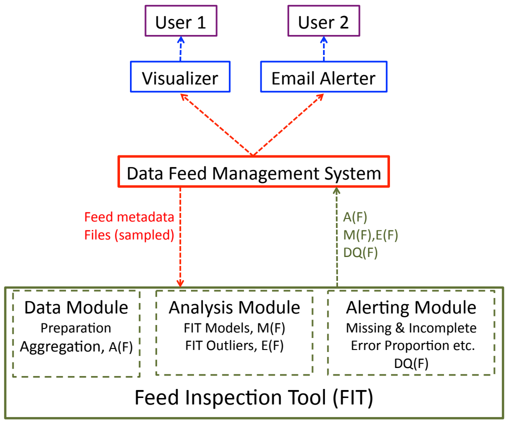


Figure 3: (a) A DFMS with multiple publishers and subscribers. Note that FIT can serve in both roles. The boxes in dark blue represent specialized subscribers like the visualizer or email alerts generator that focus on DQ tasks. (b) The FIT pipeline starts by ingesting data from the DFMS, performs analysis and alerting, and publishes results.

namically select the model that is most pertinent to their application at any point in time.

The Analysis Module performs the following two tasks. First, it builds baseline models $M(F)$ using a sliding window of history. The amount of history used, i.e., sliding window length, determines the ability to adapt. Too little history makes the models variable and noisy while too much makes the models slow to adapt to changes in statistical properties of the feed. We will see a specific example of the importance of multi-scale sliding windows in Section 5 in reacting to significant structural changes in feeds. FIT models rely on statistical summaries of centrality (mean, median, trimmed mean) and dispersion (variance, median absolute deviation) among other types of statistics. Variable transformations are a part of the model building task as well.

FIT models are adapted from well-known moving average and time decay models, for example see [18], but extended in novel ways to incorporate monitoring data feeds at multiple temporal scales. The models $M_t(F)$ at time t are typically of the form

$$E(\mathcal{T}(A_t(F))) = \sum_{g \in G} [\beta_g(M_{t-1}(F)) + \epsilon_t] * I_g(A_t(F))$$

where the model is the expected value of some functional statistic $\mathcal{T}$ estimated from the parameters β_g of the model at the previous time t. The indicator function I_g identifies the group to which a particular value of the aggregate $A_t(F)$ belongs. The parameters depend on the group g (e.g., time-of-day, day-of-week, feed, source) and the sliding window, in addition to the level of aggregation. The error ϵ_t depends on the sampling distribution of the statistic $\mathcal{T}$ but could also depend on g even though we do not explicitly denote it. When the sampling distribution of $\mathcal{T}$ is not known, FIT uses bootstrap methods to compute the error distribution.

Second, the Analysis Module tests the statistical characteristics of the data in the current aggregation interval against the most recently computed baseline models and identifies *data outliers*, $E(F)$ that are statistically different from model values.

The Analysis Module is the publisher of the models $M(F)$ and outliers $E(F)$ to be used by the Alerting Module or to be subscribed to by other applications. The Visualizer is one of the subscribers of $E(F)$, and uses it to generate time series plots (e.g., Figure 4).

4.4 Alerting Module: Multi-scale Monitoring

FIT's Alerting Module is the creator and publisher of the data quality feed, $DQ(F)$. The outliers $E(F)$ generated by the Analysis Module are data and model driven, and not necessarily of interest to all subscribers. The Alerting Module permits the publication of alerts at different scales so that subscribers can customize logical data quality feeds to alerts in order to derive a variety of data quality metrics for monitoring the health of the data. For instance, in Section 5, *missing and incomplete* data alerts are published by an Email Alerter via email along with interpretive text for the use of the data manager.

Another obvious data quality metric is the *proportion of outliers*, e.g. in Figure 8, right. The spike in outliers corresponds to the erratic behavior in the underlying data shown in the Figure 6.

In addition, FIT permits monitoring at multiple scales, to account for minor delays in data arrival. Delayed data result in immediate alerts (too little data) which disappear once the data arrives and fills in the gaps. Some users might want to act on these immediately while others might wait for the alerts to stabilize. Monitoring feeds at multiple scales of time delay is one way of addressing this issue. Alerts computed with different scales of time delay can be simultaneously posted to $DQ(F)$ with exact time delay encoded as file metadata. Subscribers can then define logical alert feeds within the DFMS to select only those alerts computed with desired time delay. These logical alert feeds configured within the DFMS can change over time as application requirements evolve. For example, subscribers in the case study in Section 5 switched to a longer monitoring delay because they expected a small portion of the data to be lagged by around half an hour.

4.5 Efficiency

FIT was deployed in a production Bistro environment responsible for managing about 100 feeds, delivering 50,000 files per day with a total volume of 120 GB/day. FIT was able to execute the entire data quality pipeline for the feed metadata with four different aggregation levels using just one processor core. Considering that the FIT pipeline is easily parallelizable by partitioning incoming data and metadata feeds, FIT's architecture is fully capable of handling a significantly higher number of feeds and data volumes.

5. CASE STUDY: RESEARCH DATA LAKE

The research data lake consists of a variety of high volume, high velocity feeds that arrive in real time. Before FIT was deployed, data managers would monitor their arrival by casually "eyeballing" daily aggregates of file counts, often only after a problem had been reported. Because several days might have passed by then, it was often too late to ask publishers to retransmit data. If the data could still be acquired, analyses and reports would have to be re-run to include delayed data or at least account for incompleteness. All these problems increase cost and cycle times.

The data managers welcomed FIT and provided useful feedback that enabled FIT to trivially create custom $DQ(F)$ feeds to incorporate their needs. In the following discussion, we have taken care to avoid revealing proprietary information; for example, we present anonymized versions of the data and omit axis labels in the figures.

5.1 FIT Deployment

FIT monitors feed metadata published by the data lake feed manager system, the Data Router. The metadata pertain to files that have been published (pub events) to a landing directory and files that have been delivered (del events) to a subscriber. Successful deliveries have a 2XX HTTP code (e.g., 204), while unsuccessful delivery attempts have non-2XX code (e.g., 503 or 100). Other metadata include feed identifier, file size, file delivery time, and a unique "publish ID". The FIT models are based on a sliding window of 112 days i.e., 16 weeks (necessary to capture day-of-week and time-of-day effects), at an hourly level of aggregation. The subscriber settled on a 45 minute monitoring time delay to ensure that the data for the prior hour is complete before processing. FIT created the following $DQ(F)$ feeds of outliers and alerts for the subscriber in two ways.

First, email alerts like the one below are sent when needed. They include interpretive text indicating the severity of the alert (critical, major, warning, status).

```
Subject:  2 critical alerts, 2 status alerts

CRITICAL: FEED1; Expected N del files, received 2%
CRITICAL: FEED1; Expected M pub files, received 2%
STATUS: FEED2; Expected del mean size X MB; received 85%
```

This alerts the data manager that FEED 1 had two critical alerts (too few files were delivered and too few were published) and that FEED 2 may also merit investigation due to the reduced average file size.

Second, graphics in which outliers are highlighted in time series plots were made available on the web (see in Figure 4). The normal feed behavior is exemplified by the figure on the left, where the dashed lines (green for "pub" and purple for "del") in the top panel indicate the expected behavior of the feed and the dots represent the observed behavior. The bottom panel contains the counts of different types of HTTP error codes. In this particular plot, feed behavior is captured through a 10% trimmed mean of the file size averaged over the files in the aggregation interval of one hour. The weekly and hourly cyclical variations are apparent in the peaks and troughs. Each outlier (i.e., unexpected mean file size) is represented by a dot attached by a line to the corresponding expected value. There are only a handful of outliers.

Recently, there was a structural change in the way files were delivered. Each file delivered is now orders of magnitude larger than it has been, and there are correspondingly fewer files. However, FIT relies on the ability to meaningfully compare new data with historical data. Because FIT used multi-scale sliding windows, the subscriber could adapt quickly by switching from a logical feed with sliding window of 112 days to that with 7 days. As a result, FIT's subscribers reacted quickly and started using appropriate models and alerts as seen in the plot on the right in Figure 4. Their new model now has much flatter peaks and troughs, which reflects the intention of the change in feed delivery, namely to even out the file sizes, thus distributing the load on the DFMS more uniformly.

5.2 Data Manager Testimonial

As an Operations Support Tool for feed ingestion, FIT is a useful tool for determining the health of the source data feeds we rely on. The graphical display provides a succinct status at a glance for key operating metrics so it is easier to see when something is abnormal. This would otherwise require complex manual queries and analysis of thousands of records. The Status Alert messages sent to our Ops mailing list have also alerted us to serious conditions during off hours. The tool's core software can be quickly configured and customized.

6. CASE STUDY: UNMATCHED FEED

Bistro receives and publishes hundreds of feeds which are the foundation for critical network applications. FIT ingests this data several times a day, and updates a web page with plots similar to the one in Figure 4. Recently there was an interest in FIT's $DQ(F)$ feeds associated with the Unmatched Feed described in Section 3.1.

6.1 Unmatched Feed

As previously described, Bistro uses pattern matching to classify files into user-defined feeds. Many files remain unclassified and are assigned to the Unmatched Feed, as Figure 5 shows. This row of plots represents a week, and each individual panel, a day. The bar plot for each day shows the distribution of DFMS actions. The dark green bar (I:Match) corresponds to files that were successfully assigned to feeds. At the other extreme, the dark red bar (E:NoMatch) corresponds to files that could not be matched. The other bars represent other types of DFMS actions. Even on Tuesday or Wednesday, when the feeds were relatively well-behaved, the percentage of unmatched files is at least 30%.

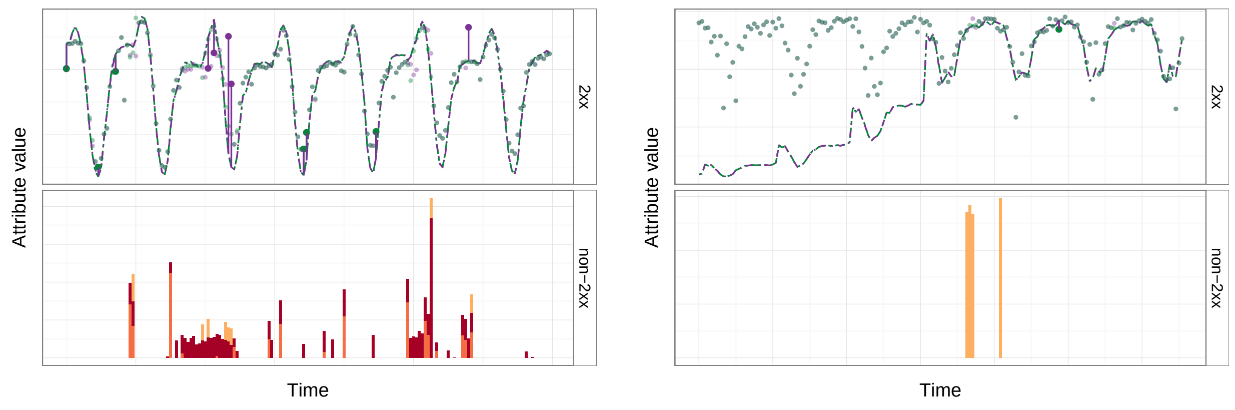


Figure 4: Online visualization of FIT's $DQ(F)$ feed of alerts; axes are unlabeled for proprietary reasons. The dashed lines (green for "pub" and purple for "del") in the top panel indicate FIT model values $M(F)$, the expected behavior of an aggregate attribute of the feed (e.g., average file size or number of files), and the dots represent the observed behavior as summarized by FIT's aggregate feed $A(F)$. The bottom panel contains the counts of different types of HTTP error codes, also a part of FIT's aggregate feed. The weekly and hourly cyclical variations are apparent in the peaks and troughs. FIT's output $DQ(F)$ contains outliers, each (e.g., unexpected mean file size) represented by a dot attached by a line to the corresponding expected value. The figure on the left represents a normal period, while the one on the right reflects FIT models rapidly adapting to a significant level shift in the attribute distribution.

The Unmatched Feed is important for two reasons: (i) known files could be labeled "unmatched", resulting in *incomplete* data that could bias downstream analytics and produce incorrect results; and (ii) it could contain important files hitherto unknown to subscribers. The filename matching process could fail for a variety of reasons. There could be a glitch in the pattern matching as a result of a very small change to the name of a file, such as a change in the formatting of an embedded time stamp; alternatively, there could be a transient system problem. Whatever be the reason, unmatched files merit further analysis. We present two interesting examples below.

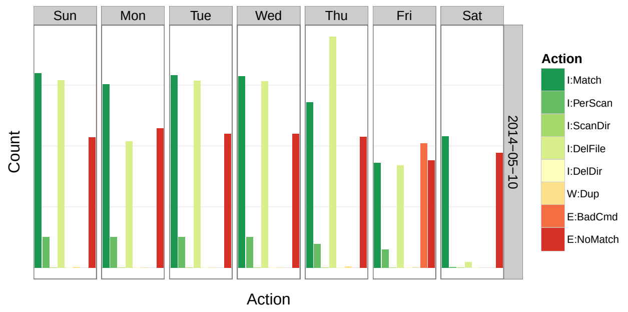


Figure 5: The row of plots represents a week, and each individual panel, a day. Bar chart for each day shows the distribution of the DFMS actions. The dark green bar (I:Match) corresponds to files that were successfully assigned to feeds. At the other extreme, the dark red bar (E:NoMatch) corresponds to files that could not be matched.

6.1.1 *Incompatible Configuration Files*

While investigating FIT's $DQ(F)$ feed during a retrospective analysis, we discovered an interesting phenomenon. In Figure 5, we noticed an unusually large prevalence of E:BadCmd (the orange bar) and a corresponding increase in the proportion of E:NoMatch actions—almost as many as Matched, whose number had fallen as well. That is, files that normally would have been assigned to customary feeds were instead part of the Unmatched Feed. This obviously resulted in losses or gaps to the feed they would normally have been assigned to and would give a false picture of activity in those feeds.

Upon further analysis, it was found that the feed had been switched to a different server which had an older configuration file. As a result, some of the filename patterns could not be processed. The gaps in the data were noticed only much later. Through a careful monitoring of FIT's $DQ(F)$ feeds associated with Bistro metadata feed, such problems can be addressed in a timely manner.

6.1.2 *Classifying Unmatched Files*

The metadata feed for the files in the Unmatched Feed contains useful information that can help assign them to feeds, such as filename, size, and arrival time. FIT's output $DQ(F)$ includes a stream of unmatched files labeled by the known feeds that they are most similar to. FIT does the labeling in the following manner. It groups the unmatched files based on filename patterns and runs clustering algorithms based on metadata such as file counts, file sizes and inter-arrival durations, and compares the results for the unmatched clusters with the results for matched files.

This particular information helped the subscribers identify an important feed that they thought had redundant information but in reality contained critical alerts that were being overlooked.

6.2 Analyst Testimonial

FIT provides a virtualized view of the feed health in a real time manner. We can capture feed quality issues faster and more efficiently. The FIT alerting system not only monitors the files counts, but also the file sizes and inter-arrival times. For example, we will see similar trends in all of the feeds when there is system or network issue, without needing to dig into each feed.

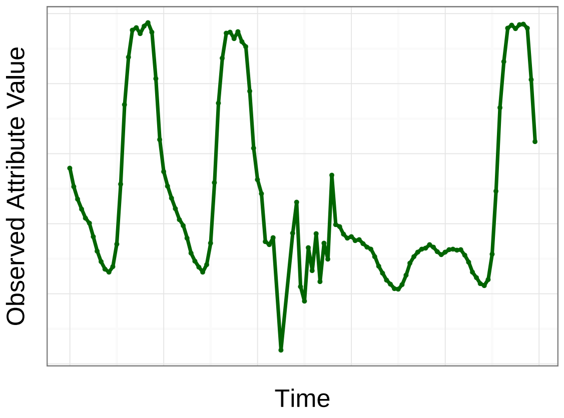


Figure 6: FIT's aggregate feed $A(F)$: A specific attribute (e.g., aggregate traffic volumes) of a specific feed. The first two cycles and the last (the three spikes) show peak behavior. The third and fourth cycles show with off-peak behavior. The first off-peak cycle is distorted due to abnormal file arrivals rather than problems with the content of files.

7. CASE STUDY: DQ OF FILE CONTENTS

Given the volume and velocity of data feeds, it is not feasible to analyze all the contents of each individual file, only judiciously chosen samples. FIT subscribes to the DFMS to receive the sampled feeds in order to keep up with the data arrival, and uses it to build statistical models and signatures of individual attributes.

A traditional sampling approach entails selecting a sample of records from *each* file, but this is inefficient in the presence of this much data. Instead, the DFMS samples files and includes the files in their entirety in the feed of file content.

In this section, we run experiments to investigate the effect of sampling on FIT's model parameters and alerts. This will enable FIT to subscribe to the DFMS feed with the suitable level of sampling. File-level samples can be chosen in two ways: a panel approach, where a selected set of feeds is sampled completely; and a random sampling approach where files are selected at random. Bistro can implement either sampling strategy, as described in Section 3.1. We compare the effects on FIT of sampling files versus records in Section 7.5.

A *longitudinal sample* or panel approach is useful when the user knows ahead of time which feeds are of the greatest interest. The advantage of this strategy is that it provides assurance that nothing will be missed in the analysis of those feeds. The drawback, of course, is the complete lack of information about the remaining feeds. The panel approach cannot capture correlations with non-panel feeds, nor can it accommodate new feeds. A *random sample* on the other hand, provides glimpses of all the feeds and could potentially capture correlations, but might require a larger sample to yield the desired level of robustness and accuracy. In this case, the panel consisted of 7 feeds that account for roughly 20% of the files handled by the DFMS. The random sample for the experiment consisted of 20% of files arriving during any given day, selected at random. The experiment was performed over a period of 30 days.

We created subsamples from the panel and random samples to study the effect of sampling on FIT's output, the model feeds, $M(F)$, and alert feeds, $DQ(F)$. Our studies include the following parameters: (1) aggregation interval, (2) sampling proportion, (3) sliding window length and (4) statistical threshold. We tested multiple combinations of these parameters, and present some of our results below.

7.1 Feed Behavior

For the purpose of illustration, we focus our discussion on one attribute in one feed of the DFMS. The methodology generalizes trivially to multiple feeds and attributes. Each file contains hundreds of thousands of records every hour. Figure 6 shows the behavior of the hourly aggregates $A(F)$ for a variable with a clear cyclic pattern. The first two cycles, and the last, show distinct troughs and peaks, while the third and fourth cycles have less range: they show off-peak behavior. The first off-peak cycle looks jagged and distorted: Further investigation showed that the distortion was caused by a disruption in *data gathering* – i.e., the arrival of files – rather than *data measurement*, the file contents.

7.2 Temporal Aggregation

As previously described, the FIT Analysis Module (Section 4.3) uses the aggregate feed, $A(F)$, which has been aggregated at multiple temporal scales. The aggregation interval will naturally influence the analyses and results. This is evident in Figure 7 which shows the expected value of the example attribute over time for one day at five levels of aggregation for both the random sample and the panel sample. The pair of hourly curves is the most detailed as well as variable. It shows the difference between the panel and the random samples. The curve for the random sample fails to show the highest peak seen in the panel sample. At longer aggregation intervals, the aggregates are smoother and the estimates based on the random sample increasingly resemble the panel until the daily aggregates (orange dots) are identical for both.

In the rest of the paper, we discuss analyses based on 3 hour aggregation intervals since it captures enough structure to see the general shape even though some details are lost.

7.3 Sampling

In this section, we present an empirical argument for choosing a sample size for FIT to subscribe to from the DFMS's feeds of files of various sampling proportions. There is no easy theoretical answer to the question given the complex nature of the data and estimates, but see [9] for a general discussion on sampling. While the discussion below compares three subsampling proportions, we studied all subsampling proportions from 10% to 100% in increments of 10%. Our procedure was to subsample from both of our original samples. A 50% random subsample of the panel

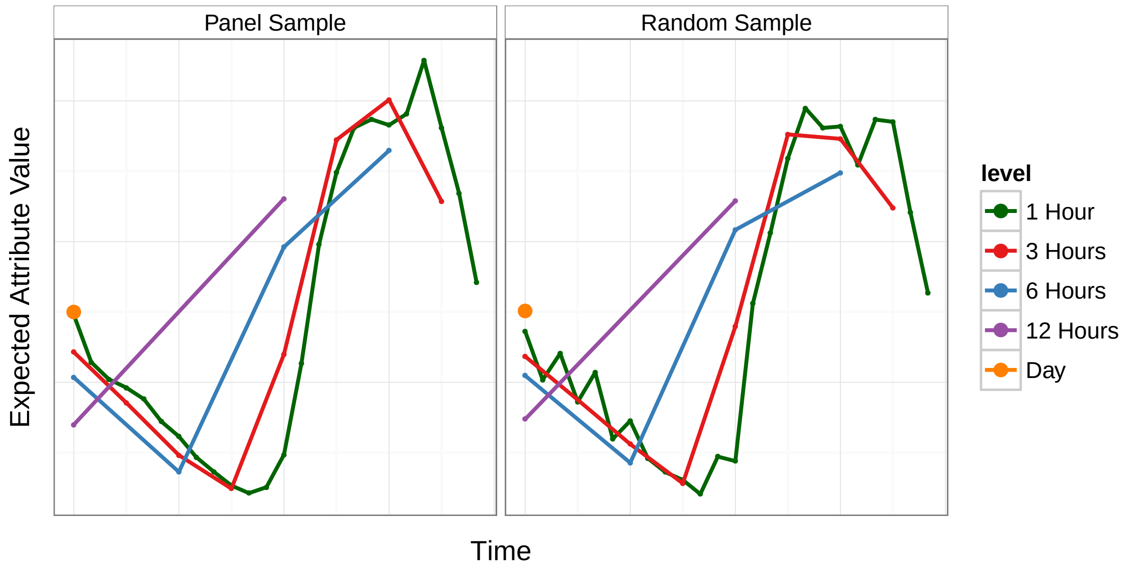


Figure 7: Temporal aggregation: FIT's model values, $M(F)$, for a given DFMS feed, for a given attribute (e.g., aggregate traffic volumes), for the two samples, panel and random, at different levels of temporal aggregation. Models based on hourly aggregates are most volatile and exhibit the greatest difference between the panel (all the data for a given feed) and the 20% random sample. The random sample estimates miss an important peak. Estimates for the two types of sample resemble each other increasingly as the level of aggregation grows coarser, until the estimates based on the daily aggregates are almost identical.

sample results in a 50% sample of the files in the panel, because the panel sample contains all files for a set of feeds. However, a 50% subsample of the random sample (a 20% sample) is equivalent to a 10% sample of the original data.

Figure 8 shows the effect of three subsampling proportions (20%, 50% and 80%) based on 100 replications each for the random sample and the panel sample. In effect, for the given feed and attribute in the figure, the subsample sizes are effectively 20%, 50% and 80% for the panel sample, and 4%, 10% and 16% for the random sample.

Each colored dot represents the expected value of the average trimmed mean of the attribute in a given aggregation interval, for a given replication of the sample. There are 100 such dots corresponding to 100 replications for each subsampling proportion, for each sample type.

The trimmed mean is one of the FIT model parameters. It is a nice way of summarizing an attribute value because it is a stable estimate that measures the general behavior of the feed. It is more robust than the mean and more efficient than the median. The argument holds in general for any other statistical estimate. For each replication of the subsample, FIT's Analysis module created the model stream $M(F)$ of the expected value based on the trimmed mean using the aggregates of the trimmed mean $A(F)$ from the Analysis module, at a scale of 21 day sliding window history as illustrated in Figure 1. Computing the confidence interval for the trimmed mean is difficult as explained in [22], but FIT's Analysis Module uses Student's t-statistic because the number of files is small.

The solid lines in the plots represent the expected value of the average of trimmed means, while the dashed lines represent the 10% confidence intervals and the dotted lines represent the 5% confidence intervals. In the panel sample plots, the lines are model values based on the ground truth since they use all the data, while the lines in the random sample are based on the original 20% random sample.

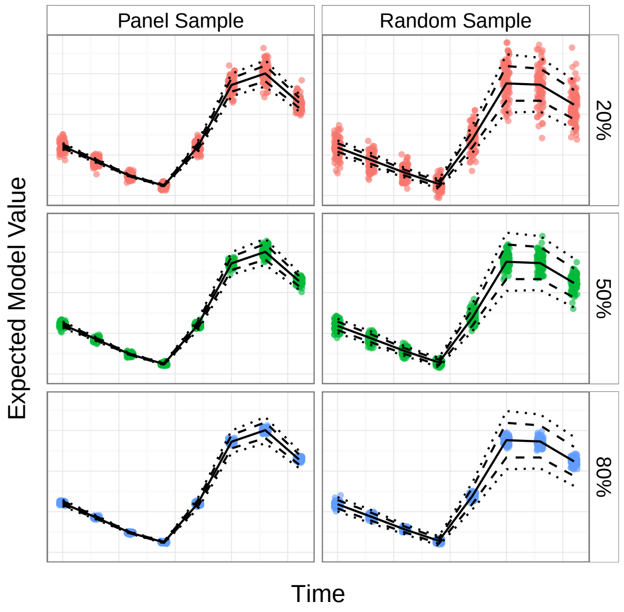
If the expected model values for a subsample fall within the confidence intervals of the "ground truth" expected values, then the subsample size is acceptable. The smallest subsample size that meets this criterion is the ideal size.

Recall that the panel samples include all files, and this is reflected in the tighter confidence bands in the plots in the left column, as well as in the tighter clustering of the sample estimates. This is expected since our samples are 20%, 50% and 80%.

Now consider the random sample in Figure 8, left. Random subsample sizes of 40% to 50% of the original 20% sample appear adequate. That is, for this particular attribute, a random sample of 8% to 10% gets us close to the results obtained from a 20% random sample. We observed the same pattern for other attributes and feeds leading us to believe that an empirical approach to choosing sample size would work well. Since FIT is a general tool agnostic to the specifics of an application, it is not possible to develop more specialized sampling strategies without additional context.

7.4 Tuning FIT Models

Given an aggregation interval and sampling proportion, FIT models and alerts are influenced by two tunable modeling choices. One is the length of the sliding window which contains the history for building models. The second is the choice of statistical threshold for generating data alerts. We run experiments to study the behavior of FIT's models and alerts next.



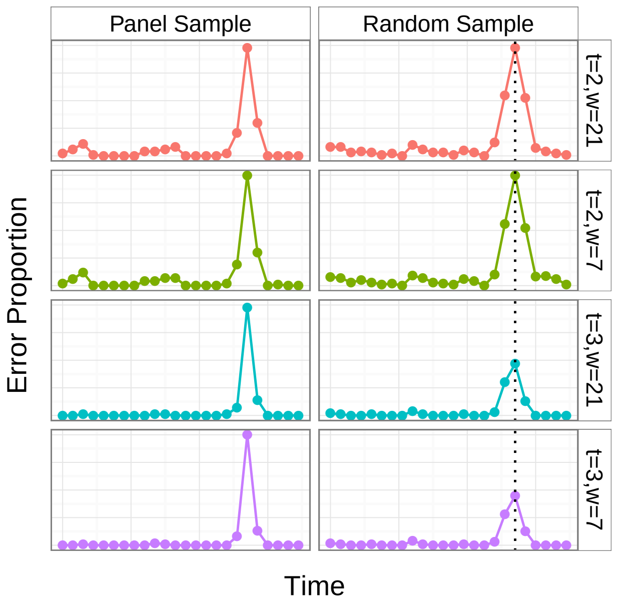


Figure 8: Left: FIT's models $M(F)$ based on subsamples: Each colored dot represents FIT's model value for a given attribute (e.g., aggregate traffic volumes) in a temporal partition for a given replication of the sample. There are 100 such dots corresponding to 100 FIT replications for each subsampling proportion, for each sample type. The solid lines represent the ground truth FIT model value, while the dashed line represents the 10% confidence intervals and the dotted line represents the 5% confidence intervals. These intervals are based on Student's t-statistic since the number of files in each partition is quite small. Right: FIT's $DQ(F)$ feed, in this case the proportion of alerts. Note the spike corresponding to a disruption in data feeds.

7.4.1 Window Size and Alerting Threshold

The length of the sliding window determines the ability of FIT's models to adapt. Longer windows dampen the effect of immediate events but also take longer to reflect changes, as shown in Section 5. We studied window sizes of 7, 14 and 21 days, and we tested 3 different alerting thresholds, depending on the test, to see how they influence FIT's output feeds, particularly $DQ(F)$, the feed of data quality alerts. The rightmost plot in Figure 8 shows the proportion of alerts generated by FIT as a part of its $DQ(F)$ output feed, for the example attribute over a 3-day period that includes the abnormality observed in Figure 6. The error proportion represents the number of replications that generated an alert out of the total 100, at a 3-hour temporal aggregation for a 50% sampling rate (that is, a 10% random sample and a 50% panel sample). Each panel represents a different combination of window size and threshold for one of the two sampling methods. Lower thresholds generate more alerts but the window size seems to have little or no influence.

Each plot of the alerts based on the random sample shows a spike corresponding to the abnormality in Figure 6, as indicated by a vertical dashed black line. However, note that panel sample plots do not have a black dashed line, indicating that analyses based on the whole data did not generate an alert! We investigated further and discovered that panel sample data generated alerts at an aggregation interval of one hour, but not at higher levels of aggregation where the test just missed the threshold. However, all the sampling proportions, including the 50% panel sample shown in the figure, alerted in a high proportion of replications.

This example shows the importance of multi scale alerting, where a blip might be masked at higher level of aggregation, or simply not be reported, i.e., a false negative, due to the statistical power of the test being less than 1.

7.5 File Sampling vs. Record Sampling

FIT models rely on DFMSs that sample entire files to avoid the overhead of parsing and reading. We ran experiments on a single feed to compare FIT's results based on file sampling versus record sampling. We noticed no significant difference as evidenced by Figure 9 which shows a three day period that corresponds to the first three days in Figure 6. The solid red line is taken from FIT's model feed $M(F)$, in this case the expected value of a given attribute derived from the the full panel sample, and the grey dots from FIT models based on a 40% sample created by sampling entire files. The black dots represent FIT models based on a 40% sample created by sampling records from the files. The values of FIT's output stream $M(F)$ from file samples versus record samples resemble each other quite closely.

The conclusion is no surprise, for two reasons. First, the files are fairly big hence quite representative of the population. Second, there are no known *a priori* correlations within records of the same file other than perhaps temporal adjacency. Therefore, it is quite reasonable to sample at the file level.

8. CONCLUSION

In this paper, we have proposed and demonstrated a new approach to monitoring data feeds. Our contributions are

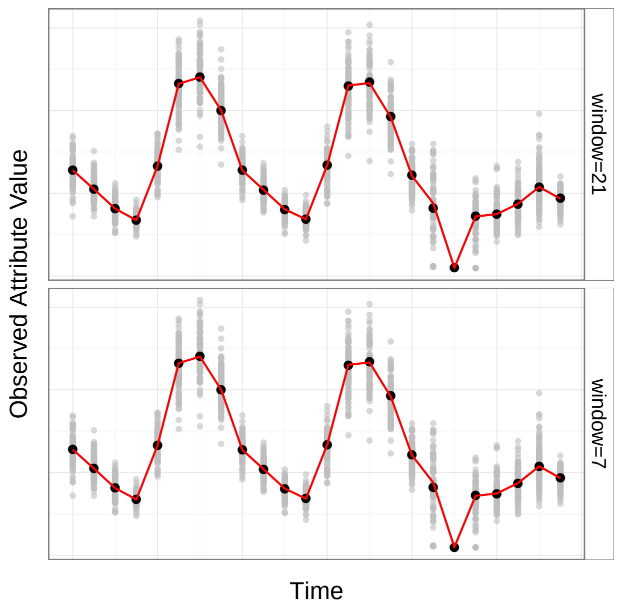


Figure 9: Sampling files versus records: The solid red line represents FIT's model feed, $M(F)$, in this case the expected value of a given attribute (e.g., aggregate traffic volumes) derived from the the full panel sample, and the grey dots from 100 replications of FIT models based on 40% samples created by sampling entire files. The black dots represent estimates from FIT's models $M(F)$ created from a 40% sample created by sampling records from the files. The model values of FIT under different sampling schemes (files vs records) resemble each other quite closely. This is to be expected since our files are quite large and are representative of the feed.

two-fold. We proposed enhancements to DFMSs, such as derived sample feeds, to permit a publish-subscribe approach to incorporate data quality modules into the DFMS architecture. In addition, we have demonstrated a unique temporal approach to outlier detection and alert generation that takes into account (a) multiple scales of temporal aggregation, (b) multiple scales of historical data, and (c) multiple scales of monitoring delay, to account for boundary conditions and small variations in data arrival. Such an approach enables us to associate greater confidence, adapt quickly, and avoid alerting on transient outliers caused by minor variations in arrival patterns.

We discussed three real case studies where FIT was effectively deployed, and shared user testimonials. FIT was able to identify outliers in *data gathering*, detected from feed metadata, as well as outliers in *data measurement*, detected from the contents of the sampled feed data. We demonstrated that FIT's file-level sampling strategy is efficient and effective for outlier detection over big data feeds.

There are many interesting directions of future work. First, when deploying FIT over a much larger set of data feeds, it would be useful to automatically identify correlated outliers across multiple data feeds that indicate systematic errors. Second, semantic errors in the content of data feeds are often detected by subscribers during the process of data analysis. We plan to enhance our DFMS architecture to support each data feed subscriber to act as a data quality feed publisher to provide data quality feedback; this would require a standard format to represent subscribers' feedback and a way for FIT to automatically incorporate such feedback into its feed quality analysis.

9. REFERENCES

[1] IBM InfoSphere DataStage. http://www.ibm.com/software/products/en/ibminfodata.

[2] Informatica PowerCenter. http://www.informatica.com/in/etl/.

[3] Oracle Data Integrator. http://www.oracle.com/us/products/middleware/data-integration/enterprise-edition/overview/index.html.

[4] SAP BusinessObjects Data Services. http://www.sap.com/pc/tech/enterprise-information-management/software/data-integrator/index.html.

[5] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer, 2006.

[6] L. Berti-Equille, J. M. Loh, and T. Dasu. A masking index for quantifying hidden glitches. In *ICDM*, pages 21–30, 2013.

[7] S. Chakraborti, P. Qiu, and A. Mukherjee. Editorial to the special issue: Nonparametric statistical process control charts. *Quality and Reliability Eng. Int.*, 31(1):1–2, 2015.

[8] K. Chen, H. Chen, N. Conway, J. M. Hellerstein, and T. S. Parikh. Usher: Improving data quality with dynamic forms. *IEEE Trans. Knowl. Data Eng.*, 23(8):1138–1153, 2011.

[9] W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, 1977.

[10] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley, 2003.

[11] T. Dasu and J. M. Loh. Statistical distortion: Consequences of data cleaning. *PVLDB*, 5(11):1674–1683, 2012.

[12] T. Dasu, J. M. Loh, and D. Srivastava. Empirical glitch explanations. In *KDD*, pages 572–581, 2014.

[13] X. L. Dong and D. Srivastava. *Big Data Integration*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2015.

[14] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[15] W. Fan, F. Geerts, and X. Jia. Conditional dependencies: A principled approach to improving data quality. In *Dataspace: The Final Frontier, BNCOD*, pages 8–20, 2009.

[16] L. Golab, F. Korn, and D. Srivastava. Efficient and effective analysis of data quality using pattern tableaux. *IEEE Data Eng. Bull.*, 34(3):26–33, 2011.

[17] R. Grover and M. J. Carey. Data ingestion in asterixdb. In *EDBT*, pages 605–616, 2015.

[18] J. D. Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[19] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.

[20] T. Redman. the impact of poor data quality on the typical enterprise. *Commun. ACM*, 41(2):79–82, 1998.

[21] V. Shkapenyuk, T. Johnson, and D. Srivastava. Bistro data feed management system. In *SIGMOD*, pages 1059–1070, 2011.

[22] M. Tsao and J. Zhou. A nonparametric confidence interval for the trimmed mean. *Nonparametric Statistics*, 14(6):665–673, 2002.

[23] M. Volkovs, F. Chiang, J. Szlichta, and R. J. Miller. Continuous data cleaning. In *ICDE*, pages 244–255, 2014.