Hunting Data Glitches in Massive Time Series Data

Tamraparni Dasu, (tamr@research.att.com)

Theodore Johnson, (johnsont@research.att.com)

Eleftherios Koutsofios, (ek@research.att.com)

AT&T Labs - Research

Abstract

In a previous paper [5] presented at IQ'99, we had proposed a method for isolating data glitches in massive data sets using a data mining method called DataSpheres. The technique runs in linear time, isolating sections of data that contain corrupted or abnormal data. In this paper, we propose using the DataSphere technique to isolate problems in time series data. We define two types of multivariate deviations, relative and within, in time and space for each data point. We discretize the attribute space into states and construct a one-step Markov chain model to summarize movement between the states. The relative deviation is based on low likelihood transitions and is used to flag suspicious movements. The within deviation is specific to a data point and helps us separate legitimate movements (e.g. bursty traffic) from data glitches (e.g. missing data). The methods we propose are distribution free, making them widely applicable. Furthermore, they are simple and can be computed from summaries, thus requiring very little storage. We demonstrate the method on real network data, isolating "abnormal" data movements over time. We conclude with a proposal for a set of general actions to take based upon the glitches detected by our algorithm.

1. Introduction

Data quality monitoring is critical in ensuring that corporate and scientific inferences are based on genuine phenomena observed in the data rather than artifacts induced by data aberrations. Detection of data problems becomes difficult when the data sets are massive, with many variables. In our previous work (IQ'99 [5]), we proposed a fast, automated method of screening snapshot data sets. In this paper, we propose a method for the rapid screening of longitudinal data for abnormal patterns. Our focus is on developing a method that is:

- Fast, runs in linear time,
- Widely applicable, (nonparametric, makes no distributional assumptions) and
- Requires very little storage

Based on the glitches that are detected, we propose a broad set of actions that need to be taken, such as:

- Raise awareness regarding data quality issues so that analysts and decision makers can incorporate this knowledge while interpreting results, especially exceptional results,
- Provide motivation and direction for data quality improvement programs and
- Specify techniques for cleaning glitches that can be incorporated into the overall data quality program.

A significant portion of data quality research is focused on managing and implementing data quality processes as in [13], [15] and [16]. Recently there has been an emphasis on building data warehouses and monitoring and measuring the information that resides in them. See [2] for details. Most commercial and academic efforts in the database community are focused on merging/purging/deletion of duplicates (see [8]) and issues related to name and address matching. In the statistical community the focus is on quality control methods borrowed from process control charts. See [14], [6] and [7]. Extensions of control charts to multivariate settings have been proposed by [1] and [12]. Multivariate methods that scale well for massive high dimensional data were proposed in IQ'99 in [5].

In this paper, we propose a general framework for detecting glitches in large databases of multivariate time series. The approach entails discretizing the attribute space using a space partitioning strategy. We use the DataSpheres partitioning technique since it scales linearly to a large number of attributes as well as a large number of data points. We treat each class of this partition as a state that a data point can be at any point in time. A given time series can then be expressed as a trajectory of the states. The trajectories can be characterized using transition probabilities that are estimated from the data. At any point in time, transitions can be ranked by their likelihood. We propose flagging "low-likelihood" transitions as data alerts. The data alerts can be further analyzed to separate abnormal but legitimate behavior (bursty traffic) from data glitches (missing data). Since the data alerts constitute a small subset of the original data, statistical methods intended for smaller data sets could be used to separate the truly bad data. The rest of the paper is organized as follows. In Section 2, we give a brief description of the DataSpheres technique. In Section 3, we give a general overview of the method we propose for finding glitches in large time series data. In Section 4, we apply the method to network data from an AT&T data warehouse. In Section 5, we outline methods for dealing with data glitches isolated by the technique proposed in this paper. Finally, in Sections 6 and 7, we present future work and conclusions. The figures that are referred to in the text of the paper are included in a separate section just before the bibliography.

2. DataSphere Partitioning

We present below a brief description of the DataSphere partitioning technique and refer the reader to [3], [10] and [9] for details. The fundamental idea is to partition the data into homogeneous sections and use representative summaries to analyze the data. Such an approach scales classical statistical methods for use on massive data. The DataSphere method partitions the attribute space based on two criteria, that of distance and direction. The DataSphere class summaries, which have special properties to be described later, are used as a basis for further analysis, including visualization. A DataSphere representation in two dimensions is included in Figure 1, included at the end of the paper.

2.1 Distance Layers

The first step in creating a DataSphere partition is defining distance layers using an appropriate subset of the numeric attributes. The choice of the subset depends on the user. If nothing is known about the data set, all the numeric attributes should be used. The attributes that are used to compute the distance are called depth attributes. The categorical attributes are used to stratify the data (if needed) subsequently. Such variables are called cohort attributes. The distance layers can be computed as follows:

- Compute a center for the data cloud using the depth attributes. Practical choices include multivariate mean, multivariate trimmed mean and componentwise median.
- Center and rescale the depth attributes using the center computed above and an appropriate measure of dispersion such as the standard deviation or interquartile range. Therefore, a data point $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ will now be

$$Y_i = (y_{i1}, y_{i2}, \ldots, y_{id}) = ((x_{i1} - \bar{x}_1)/\sigma_1, (x_{i2} - \bar{x}_2)/\sigma_2, \ldots, (x_{id} - \bar{x}_d)/\sigma_d)$$

  where $\bar{x}_j$ and σ_j are the mean and standard deviation respectively of the j^{th} component. We can replace the mean and standard deviation with other choices, such the dimensionwise median and the interquartile range. Standardizing the data makes attributes free of measurement units and scales, making them comparable.

- For each Y_i compute the distance d_i from the center.

$$d_i = \sqrt{\sum_{j=1}^{d} \left((x_{ji} - \bar{x}_j) / \sigma_j \right)^2}$$

  We have used the Euclidean distance, but other choices such the Manhattan distance can be used too.

- Sort the data points by distance and define the layer boundaries to be distance quantiles. (Quantiles divide the data set into regions of equal mass, e.g. quartiles divide the data into quarters and so on.) Using the distance quantiles as layer boundaries ensures that there is roughly the same number of data points in each layer. All data points whose distance lies between two consecutive quantiles constitute a layer.

The central layers represent "typical" observations since they are close to the measure of location we have chosen as the center. As we move to layers farther away from the center the observations become more "atypical", representing outliers. Note that the center and distance layer boundaries uniquely determine a DataSphere representation of a data set, hence they are known as the parameters of the DataSphere.

2.2 Directional Pyramids

Directional information is superimposed on the distance layers using the concept of pyramids. Briefly, a d-dimensional set can be partitioned into $2d$ pyramids P_i, $i = 1, \ldots, d$ whose tops meet at the center of the data cloud. That is, for a data point p:

$p \in P_i^+$ **, if** $|y_i| > |y_j|$ **and** $y_i > 0$**, where j = 1, …,** d **,** $j \neq i$

$p \in P_i^-$ **, if** $|y_i| > |y_j|$ **and** $y_i < 0$**, where j = 1, …,** d **,** $j \neq i$

In Figure 1, we show a two dimensional illustration of sectioning with data pyramids. The circles represent the layer (section) boundaries. The dotted diagonal lines represent the pyramid boundaries. The black and white dots might correspond to two different values of a cohort variable (e.g. gender), such as male and female.

2.3 Profiles of Summaries

Every layer-pyramid combination represents a class of the DataSphere partition. The data points in each class are summarized by a profile. A profile is a set of statistics, both scalars and vectors that summarizes the data points in a layer. Examples are counts, sums, sums of squares and cross products, special types of histograms and others. In order to be a member of the profile, a statistic should be easy to compute, be easy to combine across sections, and have the same interpretation when combined across sections or data sets.

3. Characterizing Glitches using Two Types of Deviations

We propose two measures of abnormality, which we call *deviation*, keeping with statistical terminology. A time series (such as a customer record with different types of communications usage like long distance, local, Internet) is characterized at any point in time by these two measures of deviation. Therefore, we effectively reduce a multivariate time series to a time series of two deviation attributes. We then define conditions under which the deviations are flagged as abnormal. The first measure of deviation is the *Relative Deviation*, which represents the movement of a data point relative to other data points over time. For example, an online customer might be purchasing merchandise at a faster rate than others. Another customer might continue at the same rate at which she started. The trajectories of purchases of these customers will be different. To capture this idea and to identify anomalies, we use the Markov chain approach proposed in our earlier work [4]. We employ the three following steps:

- Discretize the attribute space using the DataSphere partitioning technique. Each class (layer-pyramid combination) in the DataSphere (DS) partition is said to be a state in which the customer can be. The customers move from state to state over time. For example, a customer can start in the central layer with typical usage patterns and drop off to hardly any usage over time. Figure 2 at the end of the paper illustrates some sample movements over time.
- We observe the movements of customers over time among these states and summarize them using transition matrices. That is, the $(i,j)^{th}$ element of a transition matrix gives the probability that a customer will transition from state i to state j at time t. Note that there is an implicit assumption of a one-step Markov process here, but this assumption can be compensated for by using hazard regression to customize the transition probabilities to individual customers. A full discussion is outside the scope of this paper, but please see [4] for details and further references.

- We predict the likely states of a customer in the next time period using the transition matrices. Any observed low likelihood transitions are flagged as alerts.

The second measure of deviation is the *Within Deviation* that measures how different a data point is at any given time t with respect to its own expected behavior. The latter can be defined in several ways depending on the resource constraints. A simple strategy would be to fit a linear model to the time series of a given record using summaries and identify departures from the model.

Note that the relative deviation is more robust, since it is difficult to change state (i.e. position in the attribute space relative to others) without a significant change in the attributes. The relative deviation serves an additional purpose of identifying the data point as typical (states that are in the inner distance layers) or atypical (in the outer layers). Furthermore, the pyramid in which the attribute lies identifies the attribute that is causing the abnormality. In contrast, the within deviation is very sensitive to minor changes and is better for capturing long-term trends of the individual data point. Due to this property, we can use the within deviation to differentiate between legitimate changes and data glitches, to be discussed in a later section.

4. Example - Network Data

We used a data set that measured four attributes for every "connection", namely Bytes Received, Bytes Transmitted, Frames Received and Frames Transmitted, over the 31 day period March 1 - March 31, 2000. There were 15,596 connections observed daily. The data consisted of the daily totals of the four attributes during the 31-day period. We computed the within deviation of a point at time t simply to be the sum of the standardized deviations of the individual attributes,

$$dev_i(t) = \sum_{i=1}^{d} \left(\left((x_{ij}(t) - \overline{x_{ij}}) / s_{ij} \right)^2 \right)$$

where $x_{ij}(t)$ is the value of the j^{th} attribute of the i^{th} data point (in this case the connection) at time t, $\overline{x_{ij}}$ is the 31-day average of the j^{th} attribute for the i^{th} data point and s_{ij} is the standard deviation of the j^{th} attribute for the i^{th} individual over the 31-day period.

We used the connection average for 31 days to create a data partition with 4 layers and 8 pyramids, 2 pyramids for every attribute. For the purpose of simplification, we collapsed all the negative pyramids into a single "negative" orthant and all the positive pyramids into a single "positive" orthant within each of the four layers. Note that since the summaries in the profiles are aggregable, combining the classes of the partition to create a coarser partition is almost trivial (no need to revert to the raw data and recreate the partition anew).

Next, we computed the transition matrix elements using the sample proportion -

$$P(\mathbf{i}, \mathbf{j}, t) = n_{ij}(t) / n_i(t)$$

where P **(i, j,** t**)** is the probability of changing from state i to state j at time t, (P denotes an estimate), $n_i(t)$ is the number of points in state i at time t and $n_{ij}(t)$ is the number of points that move from state i at time t to state j at time $t + 1$.

We noticed that the estimated probability of changing states at any point in time was greater than 0.75, usually more than 0.8. Therefore, we extracted the entire multivariate time series for every data point that changed its state at least once in the 31-day period. That is, we used "change in state" to flag data alerts. The change in state happens when a data point crosses a class boundary, which in turn is a function of all four attributes. Note that if data points changed states more frequently we could have defined the most likely transition state(s) whose transition probabilities add up to some threshold (say, of 0.80) and flag all other states as abnormal. This is yet another way to define the relative deviation.

We also noticed that there was one distinctive feature that set the data problems apart from the changes caused by abnormal but genuine events. The characteristic feature was a successive flip-flop in states as in the sequence **i-to-j-to-i** over three consecutive time steps. The cause behind the flip-flop is usually missing data or a short-term outage that causes the traffic to drop.

To illustrate, we plotted four representative types of "abnormal patterns", included at the end of the paper. In Figure 3, the flip-flopping of states corresponding to the relative deviation (RLTV) indicates a data glitch. (For the purpose of plotting alone, we have used a suitable transformation of the change in state variable to denote relative deviation.) Note that the within deviation (WTHN) is much more volatile. The fact that the within deviation drops to the same level at each of the flagged events indicates that the values are being set to some default (like zero) due to missing data. In Figure 4, there is significant volatility indicating occasional bursts of activity. However, note that the within deviation at time 21 is in the opposite direction (dropping to a default value), indicative of missing data. In Figure 5, the behavior is quite different. The drop in the two deviations is indicative of migration of usage to other services or carriers, with occasional dribbles of traffic. Note that the flip-flop at times 15 and 30 correspond to within deviations of different amounts indicating that they are genuine bursts of traffic rather than data problems. Finally, Figure 6 seems to indicate a genuinely volatile customer. The flip-flop at time 17 is probably legitimate and not a data problem. Such a flip-flop could happen when a data point is close to the partition class boundary.

Thus, we have used a very simple technique to achieve very powerful results. The within and relative deviations isolate a handful (20%) of the data as potentially "dirty". The smaller subset of data can be investigated closely using methods suitable for small data sets. Note that automatic detection techniques such as logistic regression or machine learning or clustering can not be used for identifying glitches in large data sets since they are expensive and not effective on noisy data.

5. Dealing with Glitches

Once the glitches are identified, there are two major issues we need to address. First, we need to distinguish between real data problems and genuine but atypical changes in the data. Second, we need to define the action to be taken with respect to the data glitches.

5.1 Data Glitches or Genuine Changes?

How can we tell the difference between a genuine problem and a legitimate change in the data? We propose below broad guidelines:

- Genuine changes are usually persistent over time, whereas data problems appear and disappear quickly.
- Data glitches tend to appear randomly without any structure while genuine changes can be "rationalized". For example, a geographical proximity in the glitches would suggest a systemic cause such as a drought resulting in lower crop yields in that region. Similarly, a drop in revenues at a single point in time is more likely to be a data problem (missing data) than a sustained downward trend.
- We can use the within deviation of a data point to separate out differences with "structure" (systemic changes in the process that generates the data, resulting in shifts in the distribution) as opposed to random aberrations. Using departure from linear autoregressive models as measures of within deviation is a potential way of detecting structure.

5.2 Dealing with Data Glitches

There are several approaches to dealing with data glitches, depending on the type of glitch as well as the purpose of the analysis.

- If the original data set is sufficiently large, we can exclude or set aside the error prone data. However, we have to be careful that excluding parts of the data in this manner does not introduce any bias in the analysis, such as excluding data specific to a particular location or time.
- Another variation is to not include the glitch-ed record in analyses that require the corrupted attributes. For example, if only the March revenue is in question while the revenue for other months is accurate, the record should be included in all summaries and analyses except those that require the revenue for March. While this might seem obvious, many established ways of dealing with data problems would simply ignore the entire record, wasting large amounts of usable data.
- Yet another option would be to "clean" the glitched data through manual or automated methods and return it to the data set before any analysis is done. While manual methods will probably result in better quality, they are impractical due to the expense involved. Furthermore, they could introduce other unanticipated errors. Automated methods are preferred for this reason.

A frequently used automated approach entails substituting missing or suspicious values with a summary statistic such as a mean or median. Such a method clearly introduces a circular bias by increasing the data mass at the value used to fill in the missing values. Under this scheme all imputed values for a given attribute are the same.

A more sophisticated approach is based on regression. That is, the portion of data where the attribute is clean is used to develop a regression model and the resulting model is used to estimate (predict) the glitched value in the remaining data using the other attributes. Note that the

there are many possible models depending on the combination of attributes that are missing. Such an approach might be expensive. For a related discussion of the statistical treatment of missing data, see [11].

6. Future Work

Further research includes using outlier detection methods to find glitches and using learning techniques to characterize glitches (i.e. find "rules" for glitch detection) that are isolated by the automated technique proposed in this paper. Longitudinal data are particularly challenging since most extant methods do not generalize easily to nonstandard time series data due to strong assumptions made by techniques that employ, e.g. linear models.

7. Conclusion

We have used the DataSphere representation as a foundation to construct a Markov process that summarizes multivariate time series using transition probabilities. The low-likelihood transitions are sifted out using the relative deviation metric (deviation of a data point with respect to other data points) and further analyzed for structure using the within deviation metric (deviation of a data point with respect to its own expected behavior over time). If the within deviation exhibits no structure over time, it is most likely a data glitch. The above technique is fast, widely applicable and extremely effective.

FIGURES

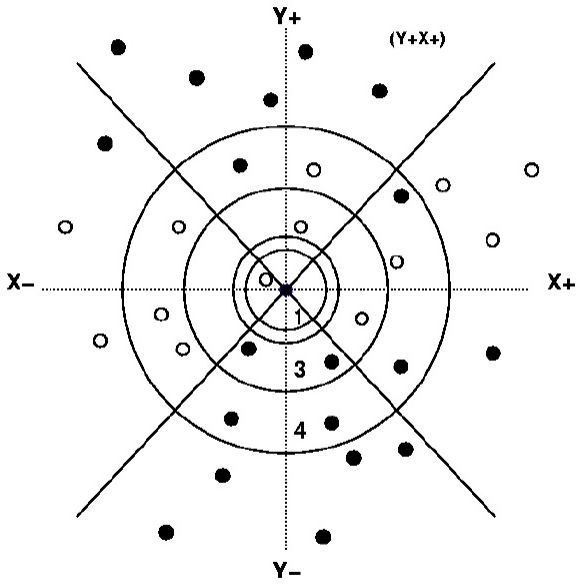

Figure 1: Data Sectioning with Pyramids.

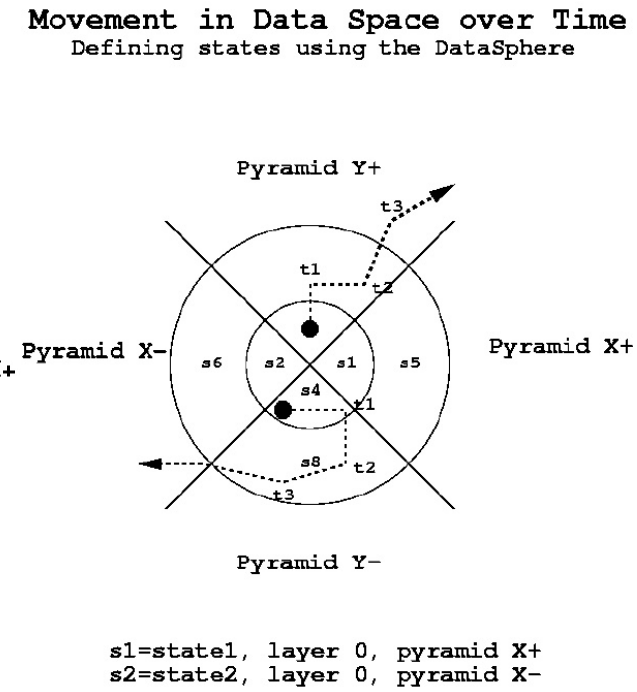

Figure 2: Movements over time.

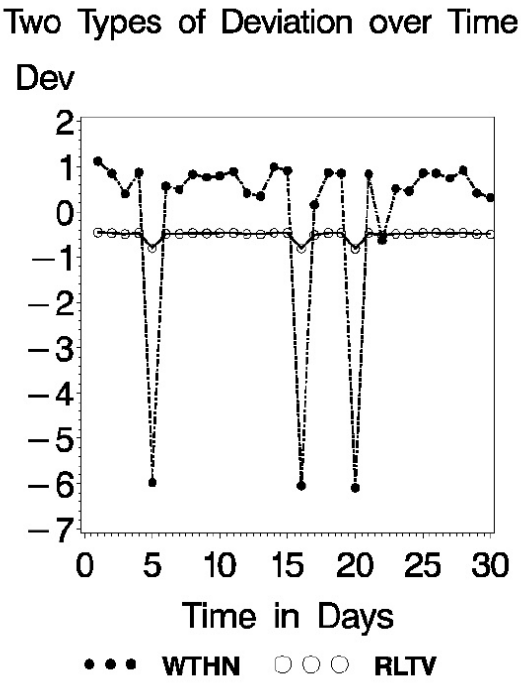

Fig 3: Flip-flop of states indicating a data glitch.

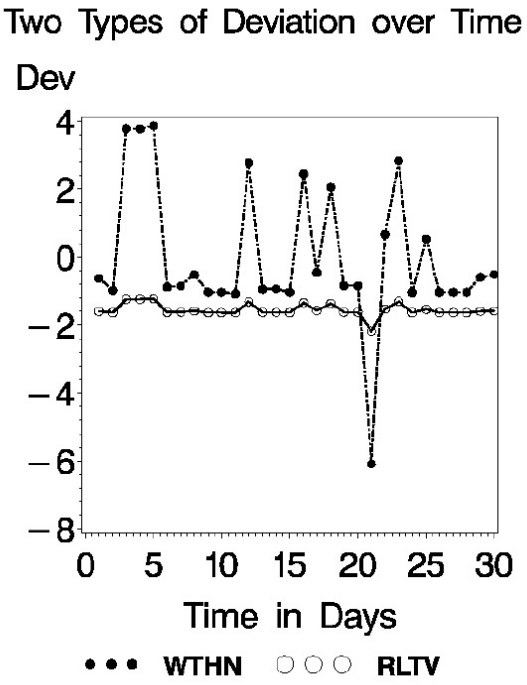

Fig 4: Bursty traffic with a data problem.

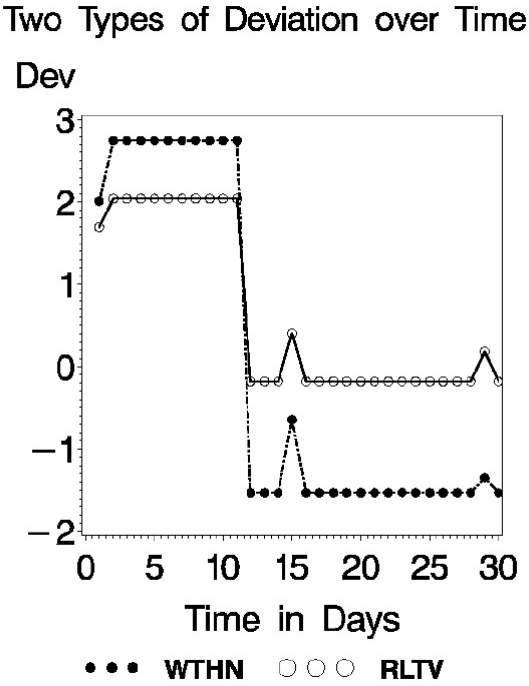

Figure 5: Big shifts in behavior.

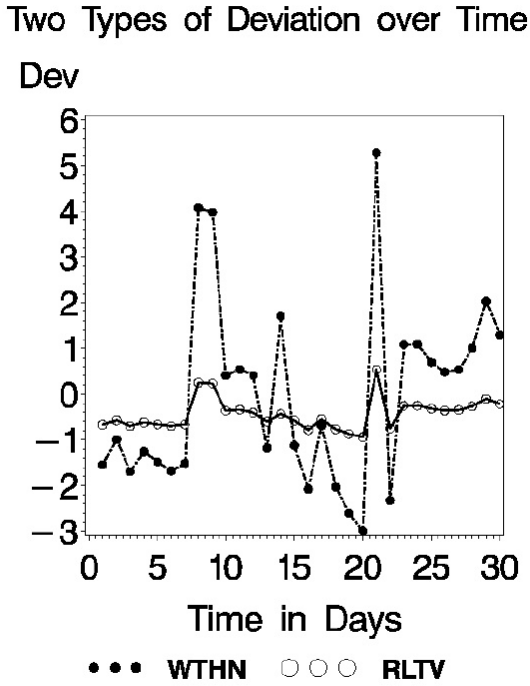

Figure 6: Legitimate volatile behavior

References

[1] Alt, F. and Smith, N. (1988). Multivariate Process Control. Handbook of Statistics, 7 (1988) 333- 351.

[2] Ballou, D. P. and Tayi G. K. (1999). Enhancing Data Quality in Data Warehouse Environments. Communications of the ACM, 42, (1999) 73-78.

[3] Dasu, T. and Johnson, T. (1997). An Efficient Method for Representing, Analyzing and Visualizing Massive High Dimensional Data Sets. Computing Sciences and Statistics. 29, (1997).

[4] Dasu, T. and Johnson, T. (1998). Efficient Modeling of Massive Longitudinal Data Using Transition Arrays. Computing Sciences and Statistics. 30, (1998).

[5] Dasu, T. and Johnson, T. (1999). Hunting of the Snark: Finding Data Glitches Using Data Mining Methods. Proceedings of the 1999 Conference on Information Quality, (pp 89-98), Cambridge MA.

[6] Deming, W. E. (1943). Statistical Adjustment of Data. John Wiley, New York.

[7] Duncan, A. J. (1986). Quality Control and Industrial Statistics. Fifth Edition, Irwin.

[8] Hernandez, M.A. and Stolfo S.J. (1995). The Merge/Purge Problem for Large Databases. Proc ACM SIGMOD Conference, (1995) 127-138.

[9] Johnson, T. and Dasu, T. (1998). Comparing Massive High Dimensional Data sets. The Fourth Int'l Conf. on Knowledge Discovery and Data Mining, 229-233.

[10] Johnson, T. and Dasu, T. (1999). Scalable Data Space Partitioning in High Dimensions. JSM99.

[11] Little, R.J.A. and Rubin, D.B. (1987). Statistical Analysis with Missing Data. John Wiley.

[12] Liu, R. Y. (1995). Control Charts for Multivariate Processes. Journal of the American Statistical Association, 90 (1995) 1380-1387.

[13] Redman, T. (1992). Data Quality: Management and Technology. Bantam Books.

[14] Shewhart, W. A. (1938). Application of Statistical Method in Mass Production. Proceedings of the Industrial Statistics Conference Held at Massachusetts Institute of Technology, September 8-9, 1938. New York: Pitman Publishing, 1939.

[15] Strong, D., Lee, Y. and Wang, R. (1997). Data Quality in Context. Communications of the ACM, 40, (1997) 103-110.

[16] Wand, Y. and Wang, R. (1996). Anchoring Data Quality Dimensions in Ontological Foundations. Communications of the ACM, November 1996.