Hunting of the Snark
Finding Data Glitches using Data Mining Methods

Tamraparni Dasu
AT&T Labs - Research

Theodore Johnson
AT&T Labs - Research

Abstract

Data quality is critical to data analysis because bad data can lead to incorrect conclusions. Problems with data are best detected early, before too much time and effort are spent ingesting and analyzing it. In this paper, we propose the use of data mining techniques for the automatic detection of data problems commonly encountered in large multivariate data sets. Data mining methods are ideal for this purpose, since they are designed for finding abnormal patterns in large volumes of data. We discuss some important types of data integrity issues. We demonstrate the use of a data mining method, the DataSphere set comparison technique (from our earlier work [6]) to detect glitches that mimic the error conditions discussed, using artificial data.

1 Introduction

Corporations base important decisions on results derived from data. Which segments of the customer base are growing? What kinds of products are they buying? What are the correlations between the products bought? Is the infrastructure performing well? What kind of capacity needs to be added to the infrastructure to meet the growth in demand? Given the critical role knowledge derived from data plays in corporate decisions, verifying the accuracy of data is important, if not sufficient (see [13] for a wholistic discussion on data quality). Furthermore, problem detection should be timely, since backtracking to fix data glitches and recover accurate analyses could be expensive and time consuming. In fact, if some transactions are overwritten or retrospective access is expensive, recovery of original data might not be possible at all.

Traditionally, the data sets used by statisticians were collected meticulously either according to a predetermined design or for answering specific substantive questions such as "Does drug A have a significant effect on reducing the symptoms of disease X?". The data were small, measured carefully and repeatedly, and the analyst had a fair idea of what the values should be. Anomalies were easy to detect just by going through the raw data or scatter plots. However the size and complexity of large data sets makes visual scanning an infeasible screening method. Some analysts adopt ad-hoc methods borrowed from quality control techniques used in manufacturing. William Edwards Deming [4], pioneered the field of quality control based on the work of Walter Shewhart [12]. While effective for process control in engineering and manufacturing, these methods do not translate well to immediate detection of inconsistencies in large complex data sets, especially for multivariate data.

Recent articles ([14], [2] and others) have highlighted conceptual frameworks for defining and enhancing data quality, some in the context of data warehouse environments. Complementary to such work, we propose the use of a previously developed data mining algorithm [3] as an instance of a greatly needed fast, automated method for the detection and screening of data glitches.

Data mining methods are especially suited to for the automatic detection of glitches in massive data since the methods are:

- designed to be scalable hence ideal for large data sets,
- aimed at isolating abnormal patterns and
- do not make distributional assumptions (usually).

Data glitches are abnormal patterns that are either aberrations from historical behavior, inconsistencies in sections of the data, or departures from acceptable tolerance limits. But, we need to distinguish between differences due to data problems and differences arising from genuine factors such as growth trends, rare but legitimate events like changes in pricing plans and change in the mixture of the subpopulations (e.g. demographic changes such as an increasingly "greying" population). In earlier work [6] we focused on detecting systematic changes that usually show up as differences that can be attributed to a substantive reason ("new subscribers have different usage patterns") and are accentuated over time. Data glitches on the other hand, tend to be scattered erratically across the data space and are not persistent over time (the cause usually disappears). Due to space constraints, we focus on the detection of changes in data, not on the difference between changes in data caused by glitches and those caused by genuine changes in distributions, which will be treated elsewhere.

The DataSphere technique is based on space partitioning. It is particularly useful for detecting subtle changes in distributions, small shifts in interactions among variables and other local differences that will be missed by aggregate based methods and univariate approaches. In addition, the method scales well and is free of distributional assumptions, making it widely applicable. Above all, the DataSphere method is truly multivariate as will be seen in the next few sections. Other data mining methods, such as clustering or classification can also be used to induce partitions on the data space. Summaries from the partitions can be used for set comparison and glitch detection. However, some of these methods do not scale well and are hard to interpret as the complexity in the data grows.

The paper is organized as follows. In Section 2, we give a brief description of types of data problems. In Section 3, we outline methods of quality control used for process management and product quality monitoring in manufacturing. In Section 4, we motivate the use of data mining techniques for ensuring data quality. In Section 5, we describe the DataSphere set comparison technique. In Section 6, we detail the experiments with artificial data sets to demonstrate the method. Finally in Section 7 we present conclusions.

2 Types of Data Glitches

Data quality is measured in terms of accuracy, completeness and timeliness. See [11] and [14] for detailed formal discussions. Considerable time is spent on ensuring completeness and timeliness of data during the process of collection, subject to resource constraints. Accuracy is often the last item to get attention and is sometimes skipped altogether. There are several reasons for this phenomenon. First, the data collection process is independent of the analysis process. The two steps involve different goals, different individuals and different time lines. While data accuracy is almost a pre-requisite for analysis, it is a secondary objective for data gathering. Second, there is no formal definition of data glitches in a general context. Third, there is no well defined methodology designed specifically for detecting glitches in an automated fashion for large data sets to ensure an acceptable level of quality.

A *data glitch* is any change introduced in the data by causes external to the process that generates the data and is different from the normal level of random noise present in most data sets. Noise is caused by uncontrollable measurement errors such as imprecise instruments, subtle variations in measurement conditions (normal wear and tear of hardware, software degeneration, climatic conditions) and human factors. Data glitches on the other hand are systematic changes caused by mega phenomena such as unintended duplicate records, switched fields and so on. Some inconsistencies are obvious and easy to detect while others are subtle, and are noticed only after they have been compounded several times resulting in significant deviations from the true values, necessitating expensive backtracking. Localized errors are swamped in aggregates, and therefore go undetected for quite some time. We present below an anecdotal discussion of a subset of commonly encountered data glitches.

2.1 Unreported changes in layout

When data processing centers make changes, downstream users are not aware of these changes for a small interval of time. However, they continue to receive and use the data feeds during this interval. Some of these changes are obvious, such as a change in layout where the position of a 13 character string variable is switched with a float. But sometimes the changes are subtle affecting only a few variables that resemble each other in their univariate behavior but differ in their interaction with other important variables. Univariate

tests and aggregates will not detect such changes. An example of such a condition could be the switching of the fields that measure the customer usage of a service with two competing providers. While the overall patterns might be similar, each provider might be used for a different purpose, affecting its interaction with other variables such as time of day, application and others.

2.2 Unreported changes in measurement/scale/format

In some situations, a field or variable is sent to the users without being processed completely. This could be due to a program exiting without completing, yet generating no error message. The processing could be such that it affects only a small proportion of the records, but would have serious consequences. An example is the application of volume discounts to generate customer bills. Some discounts are so structured that only a fraction of percentage of top users qualify. However, failure to process the discounts can create serious customer satisfaction problems alienating valuable customers. Again, tests based on aggregates often fail to detect such an error.

2.3 Temporary reversion to defaults

A third kind of frequently encountered glitch is caused by the defaulting of measuring devices to pre-set limits. For example, the reported length of any telephone call exceeding 100 minutes could be defaulted to 100, due to some temporary condition in the switches. Aggregates do not reveal such glitches unless the error condition persists for a prolonged period of time. However, the existence of such a condition for even short periods of time could result in lost revenues. Therefore it is important to detect such errors as close to real time as possible.

The following types of errors will not be considered in detail in this paper due to space constraints. The first two types of errors below pertain to completeness while the third one is related to timeliness.

2.4 Missing and default values

Missing values are very frequent in data sets. There are many different ways of dealing with them such as dropping them from analysis or substituting typical values for them. The approach depends on the amount of data missing as well as the nature of the application. There is extensive literature on the treatment of missing values in statistics. For example see [8]. An extra complication occurs when the missing values are defaulted to a valid value of the variable itself, usually an infrequent one. An example is representing missing values by zero, even though zero might be a valid but unlikely value of the variable. The implications can be serious if there is a sudden increase in the valid "zero" values, which will be masked by the missing "zero" values. While it is obvious that setting such defaults is incorrect, decisions for data collection and measurement processes are not necessarily made with a view to future analyses. Sometimes the limitations of the systems that process the data force such ambiguous defaults.

2.5 Gaps in time series records

Discontinuities in historical or transactional records can be detected easily once the need for detection has been established. For example, in a system that updates the status of a data point, it is simple to verify that the update applied to the old status results in the new status. Consider the following sequence of updates:

1. current status = 3 cellular phones, 4 phone lines
2. update: drop = 2 cellular phones, add = 1 phone line
3. new status=0 cellular phones, 9 phone lines.

Clearly, some intermediate updates are missing. However, the problem becomes serious when there are many such missing records and a large portion of the data set is quickly disqualified.

2.6 Different subpopulations have different recency

When data is collected from different sources to create a composite view, the constituent segments might have different recency. A company might receive updates from its various branches. But to ensure the validity of the aggregates and comparative studies, the data should all pertain to roughly the same time. This might not be possible always, so care should be taken to at least verify the comparability.

2.7 Changes in small subpopulations not reflected in aggregates

The data set is divided into subpopulations, usually based upon the values of categorical variables. For example, "women scientists in New Jersey in the telecommunications industry" could be a subpopulation defined by the variables gender, occupation, region and industry. Some subpopulations are so small that they are clubbed together with others to form the residual subpopulation "Other". The following types of errors might not be easily detected or traced :

- type1 - small subpopulation data missing
- type2 - small subpopulation data duplicated.

We present below a brief description of some quality control techniques that have been borrowed to monitor data behavior. We then motivate the use of data mining techniques for quality control.

3 Quality Control Charts

Quality control is a well researched area in industrial production, used extensively since World War II. Production lots are routinely sampled and tested for conformance to quality control limits with respect to some attribute such as weight or thickness. Lots for which sample estimates of the attribute (such as the mean), fall outside these limits are rejected and the production process that produced the defective lot is tested for anomalies.

Let there be N samples of objects such as bolts, from a big production batch. For each object, an attribute (e.g. weight, type(round/angular)) is measured. Ideally all samples should be of the same size but it is not necessary. For each sample, a statistic such as a mean or range (max value - min value) of the attribute is computed. Sample proportions are used in the case of categorical variables. Control limits for these sample statistics are computed using the confidence intervals of the sampling distributions. See [10] for a discussion of sampling distributions. Some of the computations are complex due to the intractability of the sampling distributions. The statistics for individual samples are plotted to see if they fall within the control limits. Some famous examples are Shewhart charts, named after W. A. Shewhart who first proposed them in 1938 ($\overline{X}$-chart, R-chart). See [12] and [5] for details. Cumulative Sum charts monitor the accumulated sum of the deviations from an expected value for the samples. It should hover around zero. If there is a trend or pre-determined thresholds are crossed, a problem is indicated. The thresholds can be target values or computed from the data such as standard deviations. Cusum charts are more sensitive than Shewhart charts. There are many variations of the above charts as well as other charts such as the Operating Characteristics Curve, Average Run Length, p-chart and others, designed for different situations including adjusting for trends over time. These are presented in great detail in [5].

4 Quality Control for Large Data Sets

The quality control methods described above are the current standard for quality control of large data sets. The methods are aimed at detecting a process that drifts out of control over time. Typically no action is taken unless there is a run of abnormal outcomes. Moreover, sampling plays a very critical role in the implementation of the charts. The method of sampling as well the sample sizes will strongly influence the conclusions drawn from the charts. The assumption of normality is indirectly required. While the methods have been used with success in industry, the following issues need to be addressed for the automatic monitoring of large, high-dimensional data sets.

- Heterogeneity – Large data sets tend to be heterogeneous.
- Localized changes – Averages tend to be very stable, not easily moved by changes in small subsections of the data. Therefore charts based on overall aggregates will not detect such glitches.
- Large number of attributes – Multivariate control charts are rare, especially without the normality assumption. Computing simultaneous confidence intervals is hard and visualizing them even harder. Bivariate charts for normally distributed data are discussed in [1]. Depth based control charts have been proposed in [9] for multivariate data using ranks to create the analogs of the traditional quality control charts.

Furthermore, in the context of automatic screening of data there are two other factors that are important. These are:

- Immediate detection of glitches rather than over time or a sequence of samples.
- Isolating the areas or sections of the data (such as heavy users, long calls, the variable revenue) that are corrupt.

We propose the use of DataSphere set comparison technique [6], a data mining method, to address the above issues in the automatic screening of massive data for data glitches.

5 DataSphere Partitioning

We present below a brief description of the DataSphere partitioning technique and refer the reader to [3], [7] and [6] 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.

5.1 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 datapoint $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ will now look like

$$\begin{aligned} Y_i &= (y_{i1}, y_{i2}, \ldots, y_{id}) \\ &= \left(\frac{x_{i1} - \overline{x_1}}{\sigma_{x_1}}, \ldots, \frac{x_{id} - \overline{x_d}}{\sigma_{x_d}} \right) \end{aligned}$$

  where $\overline{x_j}$ and σ_{x_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.

$$d_i = \sqrt{\sum_{j=1}^{d} y_{ij}^2}$$

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

- Sort the datapoints 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 are 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.

5.2 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 $P_{i\pm}, i = 1, \ldots, d$ whose tops meet at the center of the data cloud. That is, for a data point p

$$p \in P_{i+} \quad \text{if } |\, y_i \,| > |\, y_j \,|,\ y_i > 0 \quad j = 1, \ldots, d\ j \neq i$$
$$p \in P_{i-} \quad \text{if } |\, y_i \,| > |\, y_j \,|,\ y_i < 0 \quad j = 1, \ldots, d\ j \neq i$$

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.

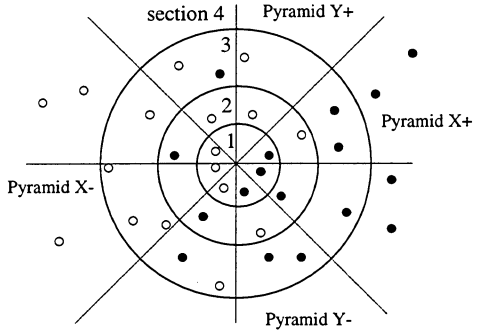

Figure 1: Data Sectioning with Pyramids.

5.3 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. 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. For the purposes of data set comparison, the statistics in the profile are the count of the data points, the vector of means and the covariance matrix.

6 Detecting Glitches Through DataSphere Set Comparison

Let D_g be the benchmark data set that is known to be clean. Let θ_g be the parameter vector, namely the center and distance layer boundaries, computed from the clean data set D_g. Let $\mathcal{P}_g(\theta_g)$ be the set of profiles of D_g, using the parameters θ_g to create the partition. $\mathcal{P}_g(\theta_g)$ will contain the vector of means, the count of data points, and the covariance matrix for every layer pyramid combination. Let D_t be the test data set whose accuracy is to be verified.

6.1 Descriptive Profiles

If the two data sets D_g and D_t are indeed similar (generated by the same multivariate distribution) and D_t is free of glitches, the center and distance cutoffs of the benchmark data set D_g can be used to partition the test data set D_t. Therefore, we use θ_g to partition D_t and compute the profiles $\mathcal{P}_t(\theta_g)$ of the resulting partition classes of D_t. We verify the hypothesis of similarity of D_g and D_t, by comparing their profiles $\mathcal{P}_g(\theta_g)$ and $\mathcal{P}_t(\theta_g)$ using the tests below. Due to space constraints, we refer the reader to [10] for a theoretical discussion of the tests and [6] for implementation in DataSphere partitions. Note that the glitch detection is done very fast since it is based on the profiles $\mathcal{P}_g(\theta_g)$ and $\mathcal{P}_t(\theta_g)$ which are several orders of magnitude smaller than the original data sets D_t and D_g.

- The Multinomial Test for Proportions
  A data glitch would cause the distribution of the points in the attribute space to change. Such a glitch can be detected using the multinomial test for distribution of points among the classes (layer-pyramid combination) defined by the DataSphere partition. The baseline or expected proportions are based on the profiles of D_g (or a pre-determined standard) $\mathcal{P}_g(\theta_g)$. A multinomial χ^2 test is used to test the difference in proportions of points in each class of the partition, between D_g and D_t. We will demonstrate this test using an example in the next section.

- Mahalanobis Distance Test for Class Means
  The corresponding multivariate means of each class in the DataSphere partitions of D_g and D_t are compared using the Mahalanobis distance test. See [6] for details. The results can be displayed graphically using "bubble charts", such as Fig 2. The absence of a bead implies that there is no statistically significant difference between the multivariate means of the class (a particular layer-pyramid combination) being compared. Problem areas as well as clustering of the problem areas are easily identified.

- Drill Down to Problem Data Sections
  The data points in the problem sections identified by the Mahalanobis test can be extracted ("drill down") and analyzed further to determine the nature of differences.

- Attribute Analysis
  In addition to multivariate tests, we can further analyze the problem sections using univariate control charts such as the Shewhart charts, to monitor individual attributes and their variability over time.

7 DataSpheres for Screening Data - An Example

We conducted two experiments to illustrate the detection of the first three glitches discussed in Section 2. The first experiment simulates the switching of two fields that are similar in their marginal behavior but differ in their interactions with other variables. The second experiment mimics two errors, incomplete processing of a variable and the truncation of another.

We generated a clean data set of 62,500 observations, each with three variables Wait, Length and Fee. All three are numeric variables. The first two are negatively correlated but have very similar marginal distributions. They are uncorrelated to the third variable Fee. The baseline DataSphere parameters which serve as a benchmark for comparison are computed using this data set.

7.1 Switched Fields

For the first experiment, we created a corrupted data set by switching fields Wait and Length. There are 43 populated classes of the DataSphere partition of D_t. The Chi-square for the Multinomial test is 56.16. The p-value for the χ^2 at 42=43-1 degrees of freedom is 0.07. The χ^2 from the Multinomial test shows significant differences at 90% level of confidence. This indicates a possible difference between D_t and D_g in the distribution of points among the partition classes, so we performed the Mahalanobis test for difference in multivariate means of D_t and D_g each DataSphere partition class. There were 43 such tests, one for each populated DataSphere class (layer-pyramid combination, also known as section). The Mahalanobis test identifies many classes where the multivariate means of the corresponding DataSphere classes are significantly different, see Figure 2. The X-axis represents the distance layers where negative layers correspond to negative pyramids. The Y-axis represents the pyramid. For example, the tuple (-5,WAIT) represents the class in the DataSphere partition corresponding to distance layer number 5 (from the center) and the pyramid of the attribute WAIT, where the deviation from the center is maximum for the attribute WAIT and and the deviation is below average (negative sign) with respect to WAIT. As mentioned earlier, the presence of a bead indicates a significant difference in the multivariate mean of the DataSphere class (X,Y) between the data sets D_g and D_t. There are many differences all across the dataset as a result of switching the fields.

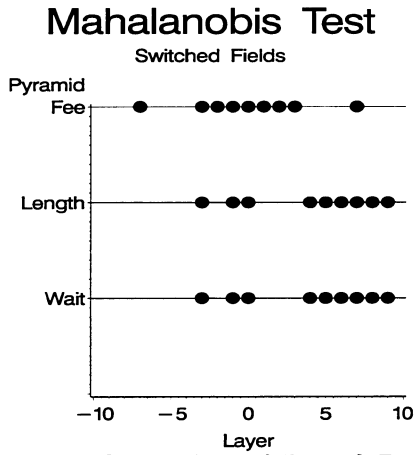

Figure 2: Comparison of D_g and D_t.

7.2 Improper Processing and Truncation of Variables

In the second experiment, we introduced two glitches. A volume charge that was applied to obtain Fee in the good data set was not applied. Only very large values of Wait qualify for the penalty. Measurements of Length were truncated so that any value below 0.05 was set to 0.05 and any value above 20 was set to 20. In short, the glitches affected a small proportion of data in the tails of the distribution.

The multinomial test came out to be not significant since the corrupted points fall in outlying layers in both the good and bad data sets. However, the Mahalanobis test of DataSphere section centers identifies the problem segments. The multivariate means for the corrupted sections of the data are significantly different and are immediately identifiable on the bubble plot shown in Figure 3.

On the other hand, the univariate tests do not identify any differences in any of the variables, in any of the sections, even after partitioning the data using DataSpheres.

Once the corrupted segments have been identified, the individual data points that fall into those segments can be extracted and studied further to determine the cause of the difference. Figure 4 shows the univariate box plots of data points in the subset identified by the Mahalanobis test in the second experiment of incomplete processing and truncation. It is clear that the variable Wait is unchanged, while Length is truncated at the tails and the distribution of Fee is less spread out for the "BAD" data set when compared to the "OK" data set. Since we could isolate the corrupt sections, we could examine them closely with methods suitable for small data sets.

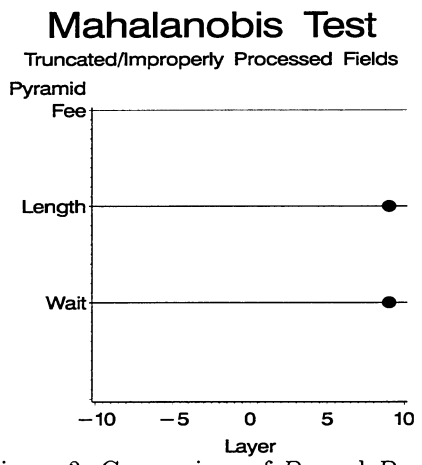

Figure 3: Comparison of D_g and D_t.

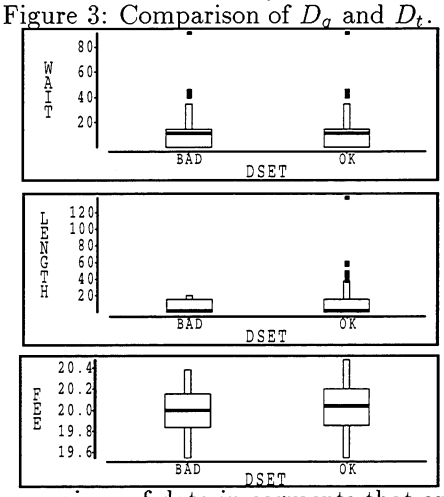

Figure 4: Comparison of data in segments that are different.

8 Conclusion

Using DataSpheres, we have demonstrated that data mining techniques are effective tools for detecting faults in data. Data mining techniques scale well and are tailored to handle heterogeneity. The DataSphere technique in particular detects (at very little expense) subtle shifts in variable interactions and distributions. Existing techniques such as those based on aggregates, univariate approaches or conventional quality control methods requiring distributional assumptions cannot detect such small changes.

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

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

[6] 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.

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

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

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

[10] Rao, C. R. (1965). *Linear Statistical Inference and Its Applications.* John Wiley.

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

[12] 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.

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

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