SampleClean: Fast and Reliable Analytics on Dirty Data

Sanjay Krishnan[1], Jiannan Wang[1], Michael J. Franklin[1], Ken Goldberg[1],
Tim Kraska[2], Tova Milo[3], Eugene Wu[4]

[1]UC Berkeley, [2]Brown University, [3]Tel Aviv University, [4]Columnia University

Abstract

An important obstacle to accurate data analytics is dirty data in the form of missing, duplicate, incorrect, or inconsistent values. In the SampleClean project, we have developed a new suite of techniques to estimate the results of queries when only a sample of data can be cleaned. Some forms of data corruption, such as duplication, can affect sampling probabilities, and thus, new techniques have to be designed to ensure correctness of the approximate query results. We first describe our initial project on computing statistically bounded estimates of `sum`*,* `count`*, and* `avg` *queries from samples of cleaned data. We subsequently explored how the same techniques could apply to other problems in database research, namely, materialized view maintenance. To avoid expensive incremental maintenance, we maintain only a sample of rows in a view, and then leverage SampleClean to approximate aggregate query results. Finally, we describe our work on a gradient-descent algorithm that extends the key ideas to the increasingly common Machine Learning-based analytics.*

1 Introduction

Data are susceptible to various forms of corruption such as missing, incorrect, or inconsistent representations [42]. Real-world data are commonly integrated from multiple sources, and the integration process may lead to a variety of errors [20]. Data analysts report that data cleaning remains one of the most time consuming steps in the analysis process [3]. Identifying and fixing data error often requires manually inspecting data, which can quickly become costly and time-consuming. While crowdsourcing is an increasingly viable option for correcting some types of errors [29, 22, 48, 24, 38, 4, 14], it comes at the significant cost of additional latency and the overhead of managing human workers.

On the other hand, ignoring the effects of dirty data is potentially dangerous. Analysts have to choose between facing the cost of data cleaning or coping with consequences of unknown inaccuracy. In this article, we describe a middle ground that we call SampleClean. SampleClean leverages insights from statistical estimation theory, approximate query processing, and data cleaning to devise algorithms for estimating query results when only a sample of data is cleaned. The intriguing part of SampleClean is that results computed using a small clean sample, results can be more accurate than those computed over the full data due to the data cleaning. This approximation error is boundable, unlike the unknown data error, and the tightness of the bound is parametrized by a flexible cleaning cost (i.e., the sampling size).

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

The motivation behind SampleClean is in many ways analogous to that of Approximate Query Processing (AQP) [36, 15, 23, 6]. For decision problems, exploratory analysis problems, and visualization, it often suffices to return an approximate query result bounded in confidence intervals. For many common aggregates, a sample size of k results in an approximation error $O(\frac{1}{\sqrt{k}})$, and therefore every additional ϵ factor of accuracy costs quadratically more. In applications where approximation can be tolerated, sampling avoids the expensive "last mile" of processing and timely answers facilitate improved user experiences and faster analysis.

In traditional AQP, approximation necessarily sacrifices accuracy for reduced latency. However, the goal of SampleClean differs from AQP, as SampleClean trades off data cleaning cost for gradual improvements in query accuracy. While SampleClean introduces approximation error, the data cleaning mitigates bias due to dirty data. There is a break-even point where a sufficient amount of data is cleaned to facilitate an accurate approximation of queries on the cleaned data, and in this sense, sampling actually improves the accuracy of the query result.

SampleClean [45] and all of its extensions [30, 31, 25], work in the *budgeted data cleaning* setting. An analyst is allowed to apply an expensive data transformation $C(\cdot)$ to only $k \ll N$ rows in a relation. One solution could be to draw k records uniformly at random and apply data cleaning, e.g., a direct extension of AQP [6] to the cleaned sample. However, data cleaning presents a number of statistical methodology problems that make this hard. First, $C(\cdot)$ may change the sampling statistics, for example, duplicated records are more likely to be sampled. Next, query processing on partially clean data, i.e., a mix of dirty and clean data, can lead to unreliable results due to the well known Simpsons Paradox. Finally, high-dimensional analytics such as Machine Learning may be very sensitive to sample size, perhaps even more so than to dirty data, and techniques for avoiding sample size dependence are required. Our research contribution in SampleClean is to study estimation techniques that avoid or mitigate these challenges.

There are two contrasting estimation techniques to address every budgeted data cleaning problem: *direct* estimation and *correction*. The direct estimate applies a query, possibly with some re-weighting and scaling to account for data cleaning, to the cleaned sample of data. Alternatively, a sample of cleaned data can also be used to correct the error in a query result over the dirty data. There is an interesting theoretical tradeoff between these approaches, where the direct approach is *robust* as its accuracy is independent of the magnitude of data error, and the correction is *sample-efficient* as its accuracy is less dependent on sample size than the direct estimate.

There are a number of new research opportunities at the intersection of data cleaning and approximate query processing. We applied the same principles to other domains with expensive data transformations such as Materialized View Maintenance and Machine Learning. In this article, we highlight three projects:

SampleClean [45] [1]: SampleClean estimates aggregate (`sum`, `count`, `avg`) queries using samples of clean data. SampleClean reweights the data to compensate for changes in sampling statistics such that query result estimations remain unbiased and bounded in confidence intervals.

View Cleaning [30]: View Cleaning generalizes the notion of data cleaning to include expensive incremental maintenance of out-of-date views. Staleness in materialized views (MVs) manifests itself as data error, i.e., a stale view has missing, superfluous, and incorrect rows. Like data cleaning, eager MV maintenance is expensive. Aggregate queries are approximated from a stale MV using a small sample of up-to-date data, resulting in bounded estimates.

ActiveClean [31]: ActiveClean extends SampleClean to a class of analytics problems called Convex Data Analytics; subsuming the aggregates studied in SampleClean and including Machine Learning such as Support Vector Machines and Linear Regression. ActiveClean exploits the convex structure of the problem to prioritize cleaning data that is likely to affect the model. ActiveClean directly integrates cleaning into the model training loop and as a result gives a bounded approximation for a given cleaning budget.

This article is organized as follows. Section 2 introduces the project and its main ideas. Section 4/5/6

[1] SampleClean refers to both the entire project and our initial publication.

describes SampleClean, View Cleaning, and ActiveClean respectively. Section 7 reviews the related work in this field. In Section 8, we highlight some of the open problems and future directions of the SampleClean project. Finally, we conclude in Section 9.

2 Background and Main Ideas

This section describes the key idea of SampleClean, namely, that data cleaning can be integrated with approximate query processing leading to bounded approximations of clean query results for a fraction of the cleaning cost.

2.1 Traditional Approximate Query Processing

A number of approximation schemes have been proposed including using Sampling, Wavelets, Sketching, and Hashing (see Cormode et al. for a survey [16]). This article focuses on Sampling-based approximations and we will use the term AQP to refer to such systems (e.g., BlinkDB[6]). Sampling-based approximate query processing is a powerful technique that allows for fast approximate results on large datasets. It has been well studied in the database community since the 1990s [27, 5, 36, 35], and methods such as BlinkDB [6] have drawn renewed attention in recent big data research. An important aspect of this work is confidence intervals, as many types of aggregates can be bounded with techniques such as concentration inequalities (e.g., Hoeffding bounds), large-deviation inequalities (e.g., Central Limit Theorem), or empirically (e.g., Bootstrap). Suppose, there is a relation R and a uniform sample S. AQP applies a query Q to S (possibly with some scaling c) to return an estimate:

$$Q(R) \approx est = c \cdot Q(S)$$

Traditionally, AQP sacrifices accuracy due to sampling for improved query latency. However in AQP, the bounds on est assume that the only source of error is approximation error introduced by sampling, however, the data itself may contain errors which could also affect query results. When the data itself is erroneous, a query result on the full data–let alone a sample, will be incorrect. The main argument for SampleClean is that when data errors significantly affect query results, sampling can be combined with data cleaning to actually improve accuracy. This leads to a counter-intuitive result where it is possible that a query on a cleaned sample of data is more accurate than a query on the entire dirty data.

2.2 Approximate Query Processing on Dirty Data

2.2.1 Two Sources of Errors: Sampling Error and Data Error

If R is dirty, then there is a true relation R_{clean}.

$$Q(R_{clean}) \neq Q(R) \approx est = c \cdot Q(S)$$

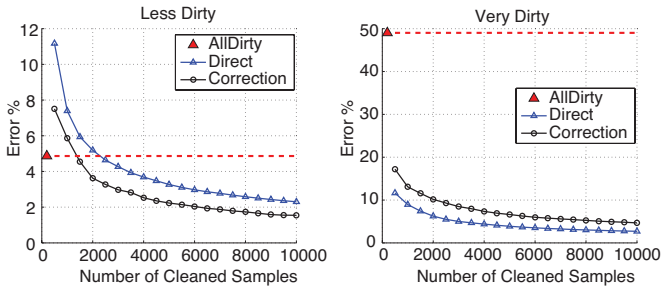
The error in est has two components: error due to sampling ϵ_s and error due to the difference with the cleaned relation $\epsilon_c = Q(R_{clean}) - Q(R)$:

$$\mid Q(R_{clean}) - est \mid \leq \epsilon_s + \epsilon_c$$

While they are both forms of query result error, ϵ_s and ϵ_c are very different quantities. ϵ_s is a random variable due to the sampling, and different samples would result in different realizations of ϵ_s. As a random variable introduced by sampling, ϵ_s can be bounded by a variety of techniques as a function of the sample size. On the other hand, ϵ_c is deterministic, and by definition is an unknown quantity until all the data is cleaned. Thus, the bounds returned by a typical AQP framework on dirty data would neglect ϵ_c.

It is possible that $R_{clean} \neq R$ but $\epsilon_c = 0$. Consider a `sum` query on the relation $R(a)$, where a is a numerical attribute. If half of the rows in R are corrupted with $+1$ and the other half are corrupted with -1, then

Figure 1: Comparison of the convergence of the methods on two TPC-H datasets of 6M tuples with simulated errors 50% error and 5% error. On the dataset with larger errors, the direct estimate gives a narrower confidence interval, and on the other the correction is more accurate.



$Q(R_{clean}) = Q(R)$. The interesting problem is when there are *systematic errors*[43] i.e., $\mid \epsilon_c \mid > 0$. In other words, the corruption that is correlated with the data, e.g., where every record is corrupted with a $+1$.

2.2.2 Key Idea I: Direct Estimate vs. Correction

The key quantity of interest is ϵ_c, and to be able to bound a query result on dirty data, requires that ϵ_c is 0 or bound ϵ_c.

Direct Estimate: This technique is a direct extension of AQP to handle data cleaning. A set of k rows is sampled uniformly at random from the dirty relation R resulting in a sample S. Data cleaning is applied to the sample S resulting in S_{clean}. Data cleaning and sampling may change the statistical and scaling properties of the query Q, so Q may have to be re-written to a query $\widehat{Q}$. $\widehat{Q}$ is applied to the sample S_{clean} and the result is returned. There are a couple of important points to note about this techniques. First, as in AQP, the direct estimate only processes a sample of data. Next, since it processes a cleaned sample of data, at no point is there a dependence on the dirty data. As we will show later in the article, the direct estimate returns a result whose accuracy is independent of the magnitude or rate of data error. One way to think about this technique is that it ensures $\epsilon_c = 0$ within the sample.

Correction: The direct estimate suffers a subtle drawback. Suppose, there are relatively few errors in the data. The errors introduced by sampling may dominate any error reductions due to data cleaning. As an alternative, we can try to estimate ϵ_c. A set of k rows is sampled uniformly at random from the dirty relation R resulting in a sample S. Data cleaning is applied to the sample S resulting in S_{clean}. The difference in applying $\widehat{Q}$ to S and $\widehat{Q}$ to S_{clean} gives an estimate of ϵ_c. The interpretation of this estimate is a correction to the query result on the full dirty data. In contrast to the direct estimate, this technique requires processing the entire dirty data (but only cleaning a sample). However, as we will later show, if errors are rare this technique gives significantly improved accuracy over the direct estimates.

2.2.3 Key Idea II: Sampling to Improve Accuracy

Figure 1 plots error as a function of the cleaned sample size on a corrupted TPCH dataset for a direct estimate, correction, and AllDirty (query on the full dirty data). In both cases, there is a break-even point (in terms of number of cleaned samples) when the data cleaning has mitigated more data error than the approximation error introduced by sampling. After this point, SampleClean improves query accuracy in comparison to AllDirty. When errors are relatively rare (5% corruption rate), the correction is more accurate. When errors are more significant (50% corruption rate), the direct estimate is more accurate. Note that the direct estimate returns results of the same accuracy regardless of the corruption rate.

3 SampleClean: Aggregate Query Processing on Dirty Data

This section introduces the SampleClean framework where the results of aggregate queries on dirty relations are estimated and bounded.

3.1 Problem Setup

Dirty Relation: Let R be a dirty relation corrupted with the following errors: (*Attribute Errors*) a row $r \in R$ has an attribute error in an attribute a if $r(a)$ is incorrect or has a missing value, (*Duplication Errors*) a row $r \in R$ is said to be a duplicate if there exists another distinct $r' \in R$ such that they refer to the same entity. For every dirty relation R, there is a cleaned version R_{clean} where attribute errors are corrected (or filled) and duplicates are merged to a canonical version.

Data Cleaning Model: For each row $r \in R$ the user-specified data cleaning technique $C(\cdot)$ must provide the following quantities: `Correct`(r) [a] the corrected value of the attribute, `Numdup`(r) the number of times the record is duplicated in the entire dataset.

Queries: SampleClean addresses aggregate queries of the form:

```
SELECT f(a) FROM R WHERE predicate GROUP BY gb_attrs
```

where f is `avg`, `sum`, or `count`.

SampleClean Problem Given a dirty relation R, and a user-specified data cleaning function $C(\cdot)$, the SampleClean problem is to estimate the result of an aggregate query q applied to the hypothetical cleaned relation $q(R_{clean})$ with a budget of applying C to at most k rows of R.

3.2 Sample Estimates

Consider a simpler problem; suppose we want to estimate the `mean` value of a set of real numbers R ignoring data error from a sample S. If S is sampled uniformly at random from R (with or without replacement), we can calculate the `mean` of S and for a large enough sample, the Central Limit Theorem (CLT) states that these estimates follow a normal distribution:

$$N(\texttt{mean}(R), \frac{var(R)}{k})$$

Since the estimate is normally distributed, we can define a confidence interval parametrized by λ (e.g., 95% indicates $\lambda = 1.96$)[2].

$$\texttt{mean}(S) \pm \lambda \sqrt{\frac{var(S)}{k}}. \tag{10}$$

It turns out that we can reformulate `sum`, `count`, and `avg` on an attribute a of a *relation* R as calculating a `mean` value so we can estimate their confidence intervals with the CLT $f(S) = \frac{1}{k}\sum_{r \in S} \phi(r)$. where $\phi(\cdot)$[3] expresses all of the necessary scaling to translate the query into a `mean` value calculation:

- `count`: $\phi(t) = \texttt{Predicate}(\text{r}) \cdot N$
- `sum`: $\phi(t) = \texttt{Predicate}(\text{r}) \cdot N \cdot r(a)$
- `avg`: $\phi(t) = \texttt{Predicate}(\text{r}) \cdot \frac{k}{k_{\texttt{pred}}} \cdot r(a)$

It turns out that these estimates are unbiased or conditionally unbiased; that is the expectation over all samples of the same size is the true answer.

[2]When estimating means from samples without replacement there is a finite population correction factor of $FPC = \frac{N-k}{N-1}$ which scales the confidence interval.

[3]`Predicate`(t) is the predicate of the aggregate query, where `Predicate`(t) = 1 or 0 denotes r satisfies or dissatisfies the predicate, respectively. $k_{\texttt{pred}}$ is the number of tuples that satisfy the predicate in the sample.

3.3 Direct Estimation with Data Errors

We are actually interested in estimating an aggregate query on $R_{\texttt{clean}}$. However, since we do not have the clean data, we cannot directly sample from $R_{\texttt{clean}}$. We must draw our sample from the dirty data R and then clean the sample. Running an aggregate query on the cleaned sample is not equivalent to computing the query result on a sample directly drawn from the clean data. Consider the case where data is duplicated, sampling from the dirty data leads to an over representation of the duplicated data in the sample. Even if cleaning is subsequently applied it does not change the fact that the sample is not uniform; and thus, the estimation method without errors presented before does not apply. Our goal is to define a new function $\phi_{\texttt{clean}}(\cdot)$, an analog to $\phi(\cdot)$, that corrects attribute values and re-scales to ensures that the estimate remains unbiased.

3.3.1 Attribute Errors

Attribute errors affect an individual row and thus do not change the sampling statistics. Consequently, if we apply the $\phi(\cdot)$ to the corrected tuple, we still preserve the uniform sampling properties of the sample S. In other words, the probability that a given tuple is sampled is not changed by the cleaning, thus we define $\phi_{\texttt{clean}}(t)$ as:

$$\phi_{\texttt{clean}}(t) = \phi\left(\texttt{Correct}(t)\right).$$

Note that the $\phi(\cdot)$ for an `avg` query is dependent on the parameter $k_{\texttt{pred}}$. If we correct values in the predicate attributes, we need to recompute $k_{\texttt{pred}}$ in the cleaned sample.

3.3.2 Duplication Errors

The duplicated data is more likely to be sampled and thus be over-represented in the estimate of the `mean`. We can address this with a weighted mean to reduce the effects of this over-representation. Furthermore, we can incorporate this weighting into $\phi_{\texttt{clean}}(\cdot)$. Specifically, if a tuple r is duplicated $m = \texttt{Numdup}(r)$ times, then it is m times more likely to be sampled, and we should down weight it with a $\frac{1}{m}$ factor compared to the other tuples in the sample. We formalize this intuition with the following lemma (proved in [45]):

Lemma 1: Let R be a population with duplicated tuples. Let $S \subseteq R$ be a uniform sample of size k. For each $r_i \in S$, let m_i denote its number of duplicates in R. (1) For `sum` and `count` queries, applying $\phi_{\texttt{clean}}(r_i) = \frac{\phi(r_i)}{m_i}$ yields an unbiased estimate; (2) For an `avg` query, the result has to be scaled by the duplication rate $d = \frac{k}{k'}$, where $k' = \sum_i \frac{1}{m_i}$, so using $\phi_{\texttt{clean}}(r_i) = d \cdot \frac{\phi(r_i)}{m_i}$ yields an unbiased estimate.

These results follow directly from importance sampling [32], where expected values can be estimated with respect to one probability measure, and corrected to reflect the expectation with respect to another.

3.3.3 Summary and Algorithm

In Table 1, we describe the transformation $\phi_{\texttt{clean}}(\cdot)$. Using this function, we formulate the direct estimation procedure:

1. Given a sample S and an aggregation function $f(\cdot)$
2. Apply $\phi_{\texttt{clean}}(\cdot)$ to each $t_i \in S$ and call the resulting set $\phi_{\texttt{clean}}(S)$
3. Calculate the mean μ_c, and the variance σ_c^2 of $\phi_{\texttt{clean}}(S)$
4. Return $\mu_c \pm \lambda\sqrt{\frac{\sigma_c^2}{K}}$

Table 1: $\phi_{\texttt{clean}}(\cdot)$ for `count`, `sum`, and `avg`. Note that N is the total size of dirty data (including duplicates).

Query	$\phi_{\texttt{clean}}(\cdot)$
`count`	$\texttt{Predicate(Correct(r))} \cdot N \cdot \frac{1}{\texttt{Numdup}(r)}$
`sum`	$\texttt{Predicate(Correct(r))} \cdot N \cdot \frac{\texttt{Correct}(r)[a]}{\texttt{Numdup}(r)}$
`avg`	$\texttt{Predicate(Correct(t))} \cdot \frac{dk}{k_{\texttt{pred}}} \cdot \frac{\texttt{Correct}(r)[a]}{\texttt{Numdup}(r)}$

3.4 Correction with Data Errors

Due to data errors, the result of the aggregation function f on the dirty population R differs from the true result $f(R) = f(R_{\texttt{clean}}) + \epsilon$. We derived a function $\phi_{\texttt{clean}}(\cdot)$ for the direct estimation. We contrasted this function with $\phi(\cdot)$ which does not clean the data. Therefore, we can write:

$$f(R) = \frac{1}{N}\sum_{r\in R}\phi(r) \qquad f(R_{\texttt{clean}}) = \frac{1}{N}\sum_{r\in R}\phi_{\texttt{clean}}(t)$$

If we solve for ϵ, we find that:

$$\epsilon = \frac{1}{N}\sum_{r\in R}\Big(\phi(r) - \phi_{\texttt{clean}}(r)\Big)$$

In other words, for every tuple r, we calculate how much $\phi_{\texttt{clean}}(r)$ changes $\phi(r)$. For a sample S, we can construct the set of differences between the two functions:

$$Q = \{\phi(r_1) - \phi_{\texttt{clean}}(r_1), \phi(r_2) - \phi_{\texttt{clean}}(r_2), \cdots, \phi(r_K) - \phi_{\texttt{clean}}(r_K)\}$$

The `mean` difference is an unbiased estimate of ϵ, the difference between $f(R)$ and $f(R_{\texttt{clean}})$. We can subtract this estimate from an existing aggregation of data to get an estimate of $f(R_{\texttt{clean}})$.

We derive the correction estimation procedure, which corrects an aggregation result:

1. Given a sample S and an aggregation function $f(\cdot)$
2. Apply $\phi(\cdot)$ and $\phi_{\texttt{clean}}(\cdot)$ to each $r_i \in S$ and call the set of differences $Q(S)$.
3. Calculate the mean μ_q, and the variance σ_q of $Q(S)$
4. Return $(f(R) - \mu_q) \pm \lambda\sqrt{\frac{\sigma_q^2}{k}}$

3.5 Analysis

Direct Estimate vs. Correction: In terms of the confidence intervals, we can analyze how direct estimation compares to correction for a fixed sample size k. Direct estimation gives an estimate that is proportional to the variance of the clean sample view: $\frac{\sigma_c^2}{k}$. Correction gives and estimate proportional to the variance of the *differences* before and after cleaning: $\frac{\sigma_q^2}{k}$. σ_q^2 can be rewritten as

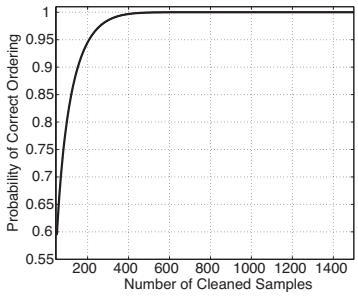
$$\sigma_c^2 + \sigma_q^2 - 2cov(S, S_{clean})$$

$cov(S, S_{clean})$ is the covariance between the the variables $\phi(r)$ and $\phi_{\texttt{clean}}(r)$. Therefore, a correction will have less variance when:

$$\sigma_S^2 \leq 2cov(S, S_{clean}) \tag{11}$$

If there are no errors $S_{clean} = S$ and then $cov(S, S_{clean}) = \sigma_c^2$ clearly satisfying the condition. Generally, if errors are small (i.e., the cleaned data is highly correlated with the dirty data) corrections will give higher accuracy. In practice, we can run both the correction and the direct estimate and take the one with a narrower confidence interval:

$$error^2 \leq O(\frac{\min\{\sigma_c^2, \sigma_q^2\}}{k}) \tag{12}$$



Name	Dirty	Clean	Pred %	Dup
Rakesh Agarwal	353	211	18.13%	1.28
Jeffery Ullman	460	255	05.00%	1.65
Michael Franklin	560	173	65.09%	1.13

Figure 2: We can return the correct ranking with 95% probability after cleaning only 210 total samples. To achieve a correct ranking with 99% probability, we require 326 samples to be cleaned.

Selectivity: Let p be the selectivity of the query and k be the sample size; that is, a fraction p records from the relation satisfy the predicate. For these queries, we can model selectivity as a reduction of effective sample size $k \cdot p$ making the estimate variance: $O(\frac{1}{k*p})$. Thus, the confidence interval's size is scaled up by $\frac{1}{\sqrt{p}}$. Just like there is a tradeoff between accuracy and maintenance cost, for a fixed accuracy, there is also a tradeoff between answering more selective queries and maintenance cost.

3.6 Results: Ranking Academic Authors

Microsoft maintains a public database of academic publications[4]. The errors in this dataset are primarily duplicated publications and mis-attributed publications. We selected publications from three database researchers: Jeffrey Ullman, Michael Franklin, and Rakesh Agarwal. To clean a sample of publications, we first manually removed the mis-attributions in the sample. Then, we applied the technique used in [44] to identify potential duplicates for all of publications in our sample, and manually examined the potential matches. For illustration purpose, we cleaned the entire dataset, and showed the cleaning results in Figure 2.

This table shows the difference between the reported number of publications (Dirty) and the number of publications after our cleaning (Clean). We also diagnosed the errors and recorded the duplication ratio (Dup) and the percentage of mis-attributed papers (Pred). Both Rakesh Agarwal and Michael Franklin had a large number of mis-attributed papers due to other authors with the same name (64 and 402 respectively). Jeffery Ullman had a comparatively larger number of duplicated papers (182).

If we were interested in ranking the authors, the dirty data would give us the wrong result. In Figure 2, we plot the probability of a correct ranking as a function of number of cleaned records with SampleClean. We show how we can return the correct ranking with 95% probability after cleaning only 210 total samples. To achieve a correct ranking with 99% probability, we require 326 samples to be cleaned. In comparison, AllDirty always returns an incorrect ranking. SampleClean provides a flexible way to achieve a desired confidence on decision based on dirty data queries.

4 View Cleaning: Stale Views are Dirty Data [30]

Suppose the relation R is in fact a derived relation V of an underlying dirty database D. We explored how we can efficiently apply a data cleaning operation to a sample of V. This extension has an important application in approximate Materialized View maintenance, where we model a stale Materialized View as dirty data, and the maintenance procedure as cleaning.

[4] http://academic.research.microsoft.com (Accessed Nov. 3, 2013)

4.1 Motivation

Some materialized views are computationally difficult to maintain and will have maintenance costs that can grow with the size of data (e.g, correlated aggregate in a sub-query). When faced with such challenges, it is common to batch updates to amortize maintenance overheads and add flexibility to scheduling. Like dirty data, any amount of staleness can lead to erroneous query results where the user has no idea about the magnitude or the scope of query error. Thus, we explore how samples of "clean" (up-to-date) data can be used for improved query processing on MVs without incurring the full cost of maintenance.

4.2 Notation and Definitions

View Cleaning returns a bounded approximation for aggregate queries on stale MVs for a flexible additional maintenance cost.

Materialized Views: Let $\mathcal{D}$ be a database which is a collection of relations $\{R_i\}$. A *materialized view* V is the result of applying a *view definition* to $\mathcal{D}$. View definitions are composed of standard relational algebra expressions: Select (σ_ϕ), Project (Π), Join ($\bowtie$), Aggregation (γ), Union ($\cup$), Intersection ($\cap$) and Difference ($-$).

Staleness: For each relation R_i there is a set of insertions ΔR_i (modeled as a relation) and a set of deletions ∇R_i. An "update" to R_i can be modeled as a deletion and then an insertion. We refer to the set of insertion and deletion relations as "delta relations", denoted by $\partial\mathcal{D}$:

$$\partial\mathcal{D} = \{\Delta R_1, ..., \Delta R_k\} \cup \{\nabla R_1, ..., \nabla R_k\}$$

A view S is considered *stale* when there exist insertions or deletions to any of its base relations. This means that at least one of the delta relations in $\partial\mathcal{D}$ is non-empty.

Maintenance: There may be multiple ways (e.g., incremental maintenance or re-computation) to maintain a view V, and we denote the up-to-date view as V'. We formalize the procedure to maintain the view as a *maintenance strategy* $\mathcal{M}$. A maintenance strategy is a relational expression the execution of which will return V'. It is a function of the database $\mathcal{D}$, the stale view V, and all the insertion and deletion relations $\partial\mathcal{D}$. In this work, we consider maintenance strategies composed of the same relational expressions as materialized views described above.

$$V' = \mathcal{M}(V, \mathcal{D}, \partial D)$$

Uniform Random Sampling: We define a sampling ratio $m \in [0, 1]$ and for each row in a view V, we include it into a sample with probability m. The relation S is a *uniform sample* of V if

$$(1)\ \forall s \in S : s \in V; \quad (2)\ Pr(s_1 \in S) = Pr(s_2 \in S) = m.$$

A sample is *clean* if and only if it is a uniform random sample of the up-to-date view V'.

4.3 Stale View Cleaning Problem

We are given a stale view S, a sample of this stale view S with ratio m, the maintenance strategy $\mathcal{M}$, the base relations $\mathcal{D}$, and the insertion and deletion relations $\partial\mathcal{D}$. We want to find a relational expression $\mathcal{C}$ such that:

$$V' = \mathcal{C}(S, \mathcal{D}, \partial\mathcal{D}),$$

where V' is a sample of the up-to-date view with ratio m.

Query Result Estimation: This problem can be addressed with the direct estimation and correction techniques described previously once we have a sample of up-to-date rows.

4.4 Cleaning a Sample View

We need to find an efficient maintnenance plan that avoids extra effort (i.e., materialization of rows outside the sample). The challenge is that $\mathcal{M}$ does not always commute with sampling. To address the commutativity problem, we need to ensure that for each $s \in V'$ all contributing rows in subexpressions to s are also sampled. We address this with a two-step process: (1) build a relation expression tree that preserves primary key relationships, and (2) use a hashing operator to push down along these relationships.

Primary Key: We recursively define a set of primary keys for all relations in the expression tree to define tuple provenance. The primary keys allow us to determine the set of rows that contribute to a row r in a derived relation. The following rules define a constructive definition for these keys:

Definition 2 (Primary Key Generation): For every relational expression R, we define the primary key attribute(s) of every expression to be:

- Base Case: All relations (leaves) must have an attribute p which is designated as a primary key.
- $\sigma_\phi(R)$: Primary key of the result is the primary key of R
- $\Pi_{(a_1,\ldots,a_k)}(R)$: Primary key of the result is the primary key of R. The primary key must always be included in the projection.
- $\bowtie_{\phi(r1,r2)}(R_1, R_2)$: Primary key of the result is the tuple of the primary keys of R_1 and R_2.
- $\gamma_{f,A}(R)$: The primary key of the result is the group by key A (which may be a set of attributes).
- $R_1 \cup R_2$: Primary key of the result is the union of the primary keys of R_1 and R_2
- $R_1 \cap R_2$: Primary key of the result is the intersection of the primary keys of R_1 and R_2
- $R_1 - R_2$: Primary key of the result is the primary key of R_1

For every node at the expression tree, these keys are guaranteed to uniquely identify a row.

Hashing: Next, instead of sampling with pseudo-random number generation, we use a hashing procedure. This procedure is a deterministic way of mapping a primary key to a Boolean, we can ensure that all contributing rows are also sampled. Let us denote the hashing operator $\eta_{a,m}(R)$. For all tuples in R, this operator applies a hash function whose range is $[0, 1]$ to primary key a (which may be a set) and selects those records with hash less than or equal to m. In a process analgous to predicate push down, we can optimize the maintenance expression by applying $\eta_{a,m}(\mathcal{M})$. The result $\mathcal{C}$ is an optimized maintenance plan expression that materializes a sample of rows.

Definition 3 (Hash push-down): For a derived relation R, the following rules can be applied to push $\eta_{a,m}(R)$ down the expression tree.

- $\sigma_\phi(R)$: Push η through the expression.
- $\Pi_{(a_1,\ldots,a_k)}(R)$: Push η through if a is in the projection.
- $\bowtie_{\phi(r1,r2)}(R_1, R_2)$: Push down is possible for foreign key equality joins.
- $\gamma_{f,A}(R)$: Push η through if a is in the group by clause A.
- $R_1 \cup R_2$: Push η through to both R_1 and R_2
- $R_1 \cap R_2$: Push η through to both R_1 and R_2
- $R_1 - R_2$: Push η through to both R_1 and R_2

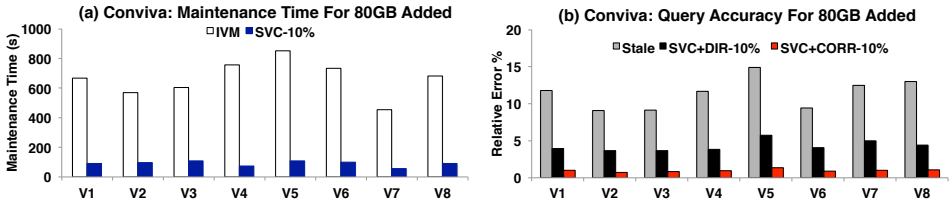


Figure 3: (a) We compare the maintenance time of View Cleaning with a 10% sample and full incremental maintenance (IVM). (b) We also evaluate the accuracy of the estimation techniques: (Direct DIR), Correction (CORR), and Dirty (Stale).

4.5 Results: Video Streaming Log Analysis

We evaluate View Cleaning on Apache Spark 1.1.0 with 1TB of logs from a video streaming company, Conviva [2]. This is a denormalized user activity log corresponding to video views and various metrics such as data transfer rates, and latencies. Accompanying this data is a four month trace of queries in SQL. We identified 8 common summary statistics-type queries that calculated engagement and error-diagnosis metrics. We populated these view definitions using the first 800GB of user activity log records. We then applied the remaining 200GB of user activity log records as the updates (i.e., in the order they arrived) in our experiments. We generated aggregate random queries over this view by taking either random time ranges or random subsets of customers.

In Figure 3(a), we show that on average over all the views, View Cleaning with a 10% sample gives a 7.5x speedup. For one of the views full incremental maintenance takes nearly 800 seconds, even on a 10-node cluster, which is a very significant cost. In Figure 3(b), we show that View Cleaning also gives highly accurate results with an average error of 0.98% for the correction estimate. This experiment highlights a few salient benefits of View Cleaning: (1) sampling is a relatively cheap operation and the relative speedups in a single node and distributed environment are similar, (2) for analytic workloads like Conviva (i.e., user engagement analysis) a 10% sample gives results with 99% accuracy, and (3) savings are still significant in systems like Spark that do not support selective updates.

5 ActiveClean: Machine Learning on Dirty Data [31]

Analytics is moving beyond SQL, and the growing popularity of predictive models [1, 7, 17, 28] leads to additional challenges in managing dirty data.

5.1 Simpson's Paradox

The challenge is that the high-dimensional models are very sensitive to systematic biases, and many of the techniques applied in practice suffer methodological problems. Consider the following approach: let k rows be cleaned, but all of the remaining dirty rows are retained in the dataset. Figure 4 highlights the dangers of this approach on a very simple dirty dataset and a linear regression model i.e., the best fit line for two variables. One of the variables is systematically corrupted with a translation in the x-axis (Figure 4a). The dirty data is marked in brown and the clean data in green, and their respective best fit lines are in blue. After cleaning only two of the data points (Figure 4b), the resulting best fit line is in the opposite direction of the true model. This is a well-known phenomenon called Simpsons paradox, where mixtures of different populations of data can result in spurious relationships [41]. Training models on a mixture of dirty and clean data can lead to unreliable results, where artificial trends introduced by the mixture can be confused for the effects of data cleaning. Figure 4c also illustrates that, even in two dimensions, models trained from small samples can be as incorrect as the mixing solution described before.

Figure 4: (a) Systematic corruption in one variable can lead to a shifted model. (b) Mixed dirty and clean data results in a less accurate model than no cleaning. (c) Small samples of only clean data can result in similarly inaccurate models.

5.2 Problem Setup

This work focuses on a class of well analyzed predictive analytics problems; ones that can be expressed as the minimization of convex loss functions. Examples includes all generalized linear models (including linear and logistic regression), all variants of support vector machines, and in fact, `avg` and `median` are also special cases.

Formally, for labeled training examples $\{(x_i, y_i)\}_{i=1}^N$, the problem is to find a vector of *model parameters* θ by minimizing a loss function ϕ over all training examples:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \phi(x_i, y_i, \theta)$$

Where ϕ is a convex function in θ. Without loss of generality, we will include regularization as part of the loss function i.e., $\phi(x_i, y_i, \theta)$ includes $r(\theta)$.

Definition 4 (Convex Data Analytics): A convex data analytics problem is specified by a set of features X, corresponding set of labels Y, and a parametrized loss function ϕ that is convex in its parameter θ. The result is a **model** θ that minimizes the sum of losses over all features and labels.

ActiveClean Problem: Let R be a dirty relation, $F(r) \mapsto (x, y)$ be a featurization that maps a record $r \in R$ to a feature vector x and label y, ϕ be a convex regularized loss, and $C(r) \mapsto r_{clean}$ be a cleaning technique that maps a record to its cleaned value. Given these inputs, the ActiveClean problem is to return a **reliable** estimate $\hat{\theta}$ of the clean model for any limit k on the number of times the data cleaning $C(\cdot)$ can be applied.

Reliable precisely means that the expected error in this estimate (i.e., L2 difference w.r.t a model trained on a fully cleaned dataset) is bounded above by a monotonically decreasing function in k and a monotonically decreasing function of the error of the dirty model. In other words, more cleaning implies more accuracy, and less initial error implies faster convergence.

5.3 Model Updates

The main insight of this work is that, in Convex Data Analytics, sampling is naturally part of the query processing. Mini-batch stochastic gradient descent (SGD) is an algorithm for finding the optimal value given the convex loss and data. In mini-batch SGD, random subsets of data are selected at each iteration and the average gradient is computed for every batch. Instead of calculating the average gradient for the batch w.r.t to the dirty data, we apply data cleaning at that point–inheriting the convergence bounds from batch SGD. It is well known that even for an arbitrary initialization SGD makes significant progress in less than one epoch (a pass through the entire dataset) [10]. Furthermore in this setting, the dirty model can be much more accurate than an arbitrary initialization; leading to highly accurate models without processing the entire data.

ActiveClean is initialized with $\theta^{(1)} = \theta^{(d)}$ which is the dirty model. At each iteration $t = \{1, ..., T\}$, the cleaning is applied to a batch of data b selected from the set of candidate dirty rows R. Then, an average gradient is estimated from the cleaned batch and the model is updated. Iterations continue until $k = T \cdot b$ rows are cleaned. The user sets the learning rate λ initially.

1. Calculate the gradient over the sample of clean data and call the result $g_S(\theta^{(t)})$
2. Apply the following update rule:
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \lambda \cdot g_S(\theta^{(t)})$$

5.4 Optimizations

ActiveClean has a number of additional optimizations exploiting the structure of Convex Data Analytics problems.

Detector: In this step, the detector select a candidate set of dirty rows $R_{dirty} \subseteq R$. There are two techniques to do this: (1) an *a priori* case, and (2) and an adaptive case. In the *a priori* case, the detector knows which data is dirty in advance. In the adaptive case, the detector learns classifier based on previously cleaned data to detect corruption in uncleaned data. This allows ActiveClean to prioritize cleaning data expected to be dirty.

Non-uniform Sampler: The sampler draws a sample of rows $S_{dirty} \subseteq R_{dirty}$. This is a non-uniform sample where each record r has a sampling probability $p(r)$. We derive the theoretical minimum variance sampling distribution, which is impossible to realize as it requires knowing the clean data in advance. Therefore, we use a first order first-order approximation of this distribution based on estimates of the clean data.

Estimator: The estimator approximates the optimal distribution derived in the Sample step. Based on the change in the featurized data $F(S_{clean})$ and $F(S_{dirty})$, it directs the next iteration of sampling to select points that will have changes most valuable to the next model update.

6 Related Work

Approximate Query Processing: AQP has been studied for more than two decades [23, 16]. Many AQP approaches [12, 5, 40, 8, 27, 37, 15, 47] were proposed, aiming to enable interactive query response times. There are also many studies on creating other synopsis of the data, such as histograms or wavelets [16]. While a substantial works on approximate query processing, these works mainly focus on how to deal with sampling errors, with little attention to data errors.

Data Cleaning: There have been many studies on various data-cleaning techniques, such as rule-based approaches [21, 18], outlier detection [26, 19], filling missing values, and duplicate detection [13, 9, 44]. In order to ensure reliable cleaning results, most of these techniques require human involvement. For example, Fan et al. [22] proposed to employ editing rules, master data and user confirmation to clean data, and proved that their approaches can always lead to correct cleaning results. Wang et al. [44] proposed a hybrid human-machine approach to detect duplicate entities in data, which can achieve higher detection accuracy than machine-only approaches. In SampleClean, the main focus is not on a specific data-cleaning technique, but rather on a new framework that enables a flexible trade-off between data cleaning cost and result quality. Indeed, we can apply any data-cleaning technique to clean the sample data, and then utilize our framework to estimate query results based on the cleaned sample.

Views and Cleaning: Meliou et al. [33] proposed a technique to trace errors in an MV to base data and find responsible erroneous tuples. They do not, however, propose a technique to correct the errors as in View Cleaning. Correcting general errors as in Meliou et al. is a hard constraint satisfaction problem. However,

in View Cleaning, through our formalization of staleness, we have a model of how updates to the base data (modeled as errors) affect MVs, which allows us to both trace errors and clean them. Wu and Madden [46] did propose a model to correct "outliers" in an MV through deletion of records in the base data. This is a more restricted model of data cleaning than View Cleaning, where the authors only consider changes to existing rows in an MV (no insertion or deletion) and do not handle the same generality of relational expressions (e.g., nested aggregates). Challamalla et al. [11] proposed an approximate technique for specifying errors as constraints on a materialized view and proposing changes to the base data such that these constraints can be satisfied. While complementary, one major difference between the three works [33, 46, 11] and View Cleaning is that they require an explicit specification of erroneous rows in a materialized view. Identifying whether a row is erroneous requires materialization and thus specifying the errors is equivalent to full incremental maintenance. However, while these approaches are not directly applicable for staleness, we see View Cleaning as complementary to these works in the dirty data setting.

7 Future Work and Open Problems

We further describe a number of open theoretical and practical problems to challenge the community:

Sample-based Optimization of Workflows: In practical data cleaning workflows, there are numerous design choices e.g., whether or not to use crowdsourcing, similarity functions, etc. An open problem is using samples of cleaned data to estimate and tune parameters on data cleaning workflows.

Optimality: For aggregate queries in the budgeted data cleaning setting, variance of the clean data σ_c^2, variance of the pairwise differences between clean and dirty data σ_d^2, and sample size k, is $O(\frac{\min\{\sigma_c,\sigma_d\}}{\sqrt{k}})$ (derived in this work) an optimal error bound? By optimal error bound, we mean that given no other information about the data distribution, the bound cannot be tightened.

Point-Lookup Dichotomy: This work focuses on aggregate analytics such as queries and statistical models. In fact, as the selectivity of the analytics goes to 0 (i.e., single row lookup), the bounds in this work limit to infinity. However, in practice, cleaning a sample of data can be used to address such queries, where a statistical model can be trained on a sample of data to learn a mapping between dirty and clean data. An open problem is exploring how much looser is a generalization bound (e.g., via Learning Theory) compared to the bounds on aggregate queries.

Confirmation Bias and Sample Re-use: Confirmation bias is defined as a "tendency to search for or interpret information in a way that confirms one's preconceptions"[39]. In systems like SampleClean, users repeatedly query and clean the sample of data. This process may encourage confirmation bias as users are allowed to modify data based on reviewing a query result (i.e., what prevents a user from removing data that does not match his or her hypothesis). An open problem is designing efficient APIs to mitigate the effects of *confirmation bias*, perhaps by limiting the number of times a user can query the sample to review the effects of a cleaning operation.

8 Conclusion

An important challenge in data analytics is presence of dirty data in the form of missing, duplicate, incorrect or inconsistent values. Data analysts report that data cleaning remains one of the most time consuming steps in the analysis process, and data cleaning can require a significant amount of developer effort in writing software or rules to fix the corruption. SampleClean studies the integration of Sample-based Approximate Query Processing and data cleaning; to provide analysts a tradeoff between cleaning the entire dataset and avoiding cleaning altogether. To the best of our knowledge, this is the first work to marry data cleaning with sampling-based query processing. While sampling introduces approximation error, the data cleaning mitigates errors in query

results. This idea opened up a number of new research opportunities, and we applied the same principles to other domains such as Materialized View Maintenance and Machine Learning.

We would like to thank Mark Wegman whose ideas helped inspire SampleClean project. This research would not have been possible without collaboration with Daniel Haas and Juan Sanchez. We would also like to acknowledge Kai Zeng, Ben Recht, and Animesh Garg for their input, feedback, and advice throughout the course of this research. This research is supported in part by NSF CISE Expeditions Award CCF-1139158, DOE Award SN10040 DE-SC0012463, and DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, IBM, SAP, The Thomas and Stacey Siebel Foundation, Adatao, Adobe, Apple, Inc., Blue Goji, Bosch, Cisco, Cray, Cloudera, EMC2, Ericsson, Facebook, Guavus, HP, Huawei, Informatica, Intel, Microsoft, NetApp, Pivotal, Samsung, Schlumberger, Splunk, Virdata and VMware.

References

[1] Berkeley data analytics stack. `https://amplab.cs.berkeley.edu/software/`.

[2] Conviva. `http://www.conviva.com/`.

[3] For big-data scientists, 'janitor work' is key hurdle to insights. http://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html.

[4] Sampleclean. http://sampleclean.org/, 2015.

[5] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. The aqua approximate query answering system. In *SIGMOD Conference*, pages 574–576, 1999.

[6] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. BlinkDB: queries with bounded errors and bounded response times on very large data. In *EuroSys*, pages 29–42, 2013.

[7] A. Alexandrov, R. Bergmann, S. Ewen, J. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke. The stratosphere platform for big data analytics. *VLDB J.*, 23(6):939–964, 2014.

[8] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *SIGMOD Conference*, pages 539–550, 2003.

[9] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, pages 39–48, 2003.

[10] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[11] A. Chalamalla, I. F. Ilyas, M. Ouzzani, and P. Papotti. Descriptive and prescriptive data cleaning. In *SIGMOD Conference*, pages 445–456, 2014.

[12] S. Chaudhuri, G. Das, and V. R. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32(2):9, 2007.

[13] P. Christen. Febrl: a freely available record linkage system with a graphical user interface. In *HDKM*, pages 17–25, 2008.

[14] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1247–1261, 2015.

[15] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot, K. Elmeleegy, and R. Sears. Online aggregation and continuous query support in mapreduce. In *SIGMOD Conference*, pages 1115–1118, 2010.

[16] G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1-3):1–294, 2012.

[17] A. Crotty, A. Galakatos, and T. Kraska. Tupleware: Distributed machine learning on small clusters. *IEEE Data Eng. Bull.*, 37(3):63–76, 2014.

[18] M. Dallachiesa, A. Ebaid, A. Eldawy, A. K. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. Nadeef: a commodity data cleaning system. In *SIGMOD Conference*, pages 541–552, 2013.

[19] T. Dasu and T. Johnson. *Exploratory data mining and data cleaning*. Wiley, 2003.

[20] X. L. Dong and D. Srivastava. Big data integration. *PVLDB*, 6(11):1188–1189, 2013.

[21] W. Fan and F. Geerts. Foundations of data quality management. *Synthesis Lectures on Data Management*, 2012.

[22] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Towards certain fixes with editing rules and master data. *PVLDB*, 3(1):173–184, 2010.

[23] M. N. Garofalakis and P. B. Gibbons. Approximate query processing: Taming the terabytes. In *VLDB*, 2001.

[24] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu. Corleone: Hands-off crowdsourcing for entity matching. In *SIGMOD*, 2014.

[25] D. Haas, S. Krishnan, J. Wang, M. J. Franklin, and E. Wu. Wisteria: Nurturing scalable data cleaning infrastructure. *Proceedings of the VLDB Endowment*, 8(12), 2015.

[26] J. M. Hellerstein. Quantitative data cleaning for large databases. *United Nations Economic Commission for Europe (UNECE)*, 2008.

[27] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *SIGMOD Conference*, pages 171–182, 1997.

[28] J. M. Hellerstein, C. Ré, F. Schoppmann, D. Z. Wang, E. Fratkin, A. Gorajek, K. S. Ng, C. Welton, X. Feng, K. Li, and A. Kumar. The madlib analytics library or MAD skills, the SQL. *PVLDB*, 5(12):1700–1711, 2012.

[29] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy. Pay-as-you-go user feedback for dataspace systems. In *SIGMOD Conference*, pages 847–860, 2008.

[30] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, and T. Kraska. Stale view cleaning: Getting fresh answers from stale materialized views. *Proceedings of the VLDB Endowment*, 8(12), 2015.

[31] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, and E. Wu. Activeclean: Progressive data cleaning for convex data analytics. 2015.

[32] J. S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, 6(2):113–119, 1996.

[33] A. Meliou, W. Gatterbauer, S. Nath, and D. Suciu. Tracing data errors with view-conditioned causality. In *SIGMOD Conference*, pages 505–516, 2011.

[34] F. Olken. *Random sampling from databases*. PhD thesis, University of California, 1993.

[35] F. Olken and D. Rotem. Simple random sampling from relational databases. In *VLDB*, pages 160–169, 1986.

[36] F. Olken and D. Rotem. Maintenance of materialized views of sampling queries. In *ICDE*, pages 632–641, 1992.

[37] N. Pansare, V. R. Borkar, C. Jermaine, and T. Condie. Online aggregation for large mapreduce jobs. *PVLDB*, 4(11):1135–1145, 2011.

[38] H. Park and J. Widom. Crowdfill: collecting structured data from the crowd. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 577–588, 2014.

[39] S. Plous. *The psychology of judgment and decision making.* Mcgraw-Hill Book Company, 1993.

[40] L. Sidirourgos, M. L. Kersten, and P. A. Boncz. Sciborq: Scientific data management with bounds on runtime and quality. In *CIDR*, pages 296–301, 2011.

[41] E. H. Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 238–241, 1951.

[42] N. Swartz. Gartner warns firms of 'dirty data'. *Information Management Journal*, 41(3), 2007.

[43] J. R. Taylor. An introduction to error analysis: The study of uncertainties in physical measurements, 327 pp. *Univ. Sci. Books, Mill Valley, Calif*, 1982.

[44] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

[45] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *SIGMOD Conference*, pages 469–480, 2014.

[46] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8):553–564, 2013.

[47] S. Wu, S. Jiang, B. C. Ooi, and K.-L. Tan. Distributed online aggregation. *PVLDB*, 2(1):443–454, 2009.

[48] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *PVLDB*, 2011.