A Sample-and-Clean Framework for Fast and Accurate Query Processing on Dirty Data

Jiannan Wang, Sanjay Krishnan, Michael J. Franklin, Ken Goldberg, Tim Kraska[†], Tova Milo[#]
UC Berkeley, [†]Brown University, [#]Tel Aviv University
{jnwang, sanjaykrishnan, franklin, goldberg}@berkeley.edu
tim_kraska@brown.edu, milo@cs.tau.ac.il

ABSTRACT

In emerging Big Data scenarios, obtaining timely, high-quality answers to aggregate queries is difficult due to the challenges of processing and cleaning large, dirty data sets. To increase the speed of query processing, there has been a resurgence of interest in sampling-based approximate query processing (SAQP). In its usual formulation, however, SAQP does not address data cleaning at all, and in fact, exacerbates answer quality problems by introducing sampling error. In this paper, we explore an intriguing opportunity. That is, we explore the use of sampling to actually improve answer quality. We introduce the Sample-and-Clean framework, which applies data cleaning to a relatively small subset of the data and uses the results of the cleaning process to lessen the impact of dirty data on aggregate query answers. We derive confidence intervals as a function of sample size and show how our approach addresses error bias. We evaluate the Sample-and-Clean framework using data from three sources: the TPC-H benchmark with synthetic noise, a subset of the Microsoft academic citation index and a sensor data set. Our results are consistent with the theoretical confidence intervals and suggest that the Sample-and-Clean framework can produce significant improvements in accuracy compared to query processing without data cleaning and speed compared to data cleaning without sampling.

1. INTRODUCTION

Aggregate query processing over very large datasets can be slow and prone to error due to dirty (missing, erroneous, duplicated, or corrupted) values. Sampling-based approximate query processing (SAQP) 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 [2,32], and methods such as BlinkDB [3] have drawn renewed attention in recent big data research.

SAQP's result estimates assume that the only source of error is uncertainty introduced by sampling, however, the data itself may contain errors which could also affect query results. According to an industry survey, more than 25% of the critical data in top companies contained significant data errors [51]. Real-world data is commonly integrated from multiple sources, and the integration process may lead to a variety of data errors, such as incorrect values and duplicate representations of the same real-world entity [21]. Therefore, even running a query on the entire data may still not get an accurate query result, and sampling further reduces result quality.

Data cleaning typically requires domain-specific software that can be costly and time-consuming to develop. Particularly, in order to obtain reliable cleaning results, many data-cleaning techniques need humans to get involved [24,36,58]. While crowdsourcing makes this increasingly feasible, it is still highly inefficient for large datasets [54].

In this paper, we explore an intriguing opportunity that sampling presents, namely, that when integrated with data cleaning, sampling has potential to improve answer quality. We present SampleClean, a novel framework that cleans only samples of the data to process aggregate queries (e.g., `avg`, `count`, `sum`, etc). SampleClean employs two approaches to estimate a query from the cleaned sample: (1) NormalizedSC uses the cleaned sample to correct the error in a query result over the dirty data; (2) RawSC directly estimates the true query result based on the cleaned sample. Both of these approaches return unbiased query results and confidence intervals as a function of sample size.

Comparing the two approaches, we find that NormalizedSC gives more accurate results on datasets with relatively small errors, whereas RawSC is more robust to datasets with more errors. Since SampleClean can return the better result of NormalizedSC and RawSC, it gives accurate estimates for a variety of different error distributions.

In summary, our paper makes the following contributions:

- We present SampleClean, a novel framework which only requires users to clean a sample of data, and utilizes the cleaned sample to process aggregate queries.
- We propose NormalizedSC that can use a cleaned sample to correct the bias of the query results over the uncleaned data.
- We develop RawSC that can use a cleaned sample to directly estimate the query results of the cleaned data.
- We conduct extensive experiments on both real and synthetic data sets. The results suggest that estimated values can rapidly converge toward the true values with surprisingly few cleaned samples, offering significant improvement in cost over cleaning all of the data and significant improvement in accuracy over cleaning none of the data.

The paper is organized as follows. Section 2 presents SampleClean framework for query processing on dirty data. We present RawSC in Section 3 and discuss NormalizedSC in Section 4. We describe the experimental results in Section 5. Section 6 covers related work and Section 7 makes the conclusion.

SIGMOD'14, June 22–27, 2014, Snowbird, Utah, USA.
ACM 978-1-4503-2376-5/14/06 ...$15.00.
http://dx.doi.org/10.1145/2588555.2610505.

2. QUERY PROCESSING ON DIRTY DATA

Like other SAQP systems, our main focus is on aggregate numerical queries (`avg`, `sum`, `count`, `var`, `geomean`, `product`) of the form:

```
SELECT f(attrs) FROM table
WHERE predicate
GROUP BY attrs
```

When running the aggregate queries on large and dirty datasets, there may be two separate sources of errors that affect result quality. (1) Sampling error: since data is large, we may execute queries on a sample of the data to reduce query times. (2) Data error: since real-world data is dirty, queries on the dirty data also lead to inaccurate query results.

In this section, we first precisely characterize sampling and data errors, and then present our SampleClean framework to deal with these two types of errors. Throughout the section, we will refer to the following example query on a dataset of academic publications:

```
SELECT AVG(citation_count) FROM papers
GROUP BY pub_year
```

which finds the average number of citations of the publications published every year.

2.1 Sampling Error

There are many different ways to sample data; a data sample could be either created online during the query time [14,32,47,57] or built offline from past query workloads [2,3,5,11]. Consider our example citation query. A uniform random-sampling scheme randomly selects a set of papers from `papers` such that every paper has an equal probability of selection. To answer queries with a highly selective predicate or a `group-by` clause, prior works employ stratified-sampling [1,3,32], which performs a uniform random sampling scheme in each group, to guarantee that every group has a large enough sample size to estimate a good result. The approaches presented in this paper can support both uniformly random samples and stratified samples. However, for simplicity, we present our analysis with uniform samples.

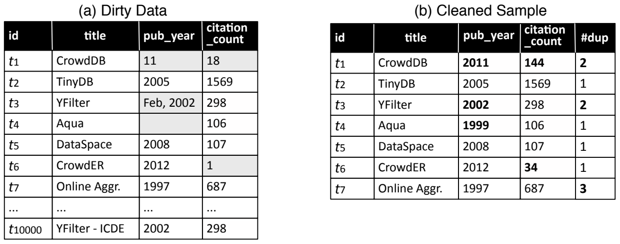
Answering queries on a sample has an inherent uncertainty since a different sample may yield a different result. Quantifying this uncertainty has been extensively studied in statistics [43]. Due to this uncertainty, we return confidence intervals in addition to results. For example, given a confidence probability (e.g., 95%), we can apply results from sampling statistics to estimate the average number of citations along with a confidence interval (e.g. ±10), which means that the estimated average number is within ±10 of the actual value with 95% probability. The confidence interval quantifies the uncertainty introduced by sampling the data.

2.2 Data Error

In this work, we focus on three types of data errors: value error, condition error, and duplication error. We use our example query to illustrate how these errors can affect results.

Value error: When an error occurs in the aggregation attributes of the query (i.e. `citation_count`), it will lead to an incorrect aggregate result. For example, consider the dirty data in Figure 1(a). The first paper t_1 involves value error since its citation count should be 144 instead of 18.

Condition error: When an error occurs in the predicate or group-by attribute of the query (i.e. `pub_year`), there may be some tuples that are *falsely* added into or excluded from a group, leading to an incorrect result. In Figure 1(a), the first paper t_1 also has condition error since it was published in the year 2011 rather than 11.



(a) Dirty Data

id	title	pub_year	citation _count
t_1	CrowdDB	11	18
t_2	TinyDB	2005	1569
t_3	YFilter	Feb, 2002	298
t_4	Aqua		106
t_5	DataSpace	2008	107
t_6	CrowdER	2012	1
t_7	Online Aggr.	1997	687
...	...	...	...
t_{10000}	YFilter - ICDE	2002	298

(b) Cleaned Sample

id	title	pub_year	citation _count	#dup
t_1	CrowdDB	**2011**	**144**	**2**
t_2	TinyDB	2005	1569	1
t_3	YFilter	**2002**	298	**2**
t_4	Aqua	**1999**	106	1
t_5	DataSpace	2008	107	1
t_6	CrowdER	2012	**34**	1
t_7	Online Aggr.	1997	687	**3**

Figure 1: An example of dirty data and cleaned sample (Shaded cells denote dirty values, and their cleaned values are in bold font).

Duplication error: If data contains duplicate tuples (e.g., different representations of the same paper), the aggregate result will also be affected. This type of error commonly happens when the data is integrated from multiple sources. For instance, in Figure 1(a), the third paper t_3 has duplication error as it refers to the same paper as t_{10000}.

While data cleaning can fix the data errors, cleaning the entire data is usually time consuming, often requiring user confirmation or crowdsourcing. For this reason, we have developed the SampleClean framework.

2.3 SampleClean Framework

Figure 2 illustrates all of the components of our framework. SampleClean first creates a random sample of dirty data, and then applies a data-cleaning technique to clean the sample. After cleaning the sample, SampleClean uses the cleaned sample to answer aggregate queries. SampleClean gives results that are unbiased which means in expectation the estimates are equal to the query results if the entire dataset was cleaned by the data-cleaning technique.

The SampleClean framework is independent of how samples are cleaned, and in this paper, we consider data cleaning as a user-provided module. Specifically, for each tuple in the sample, the cleaning module corrects the attribute values of the tuple, and estimates the number of duplicates for the tuple from the dirty data. For example, consider a sample, $S = \{t_1, t_2, \cdots, t_7\}$ of the dirty data in Figure 1(a). Figure 1(b) shows the corresponding cleaned sample. For the first paper t_1, we correct `pub_year` from 11 to 2011, correct `citation_count` from 18 to 144, and identify two duplicate papers (including t_1 itself) in the dirty data.

2.3.1 *Cleaning Value and Condition Errors*

To reduce value errors and condition errors, the data-cleaning technique only needs to clean attribute values in the sample, and we can apply a variety of recently proposed data cleaning techniques to achieve this. For example, outlier detection [31,35] and rule-based approaches [17,23] have been proposed to solve this problem. In addition, Fan et al. [24] proposed editing rules, master data and user confirmation to correct attribute values, and they proved that their approaches can always obtain perfect cleaning results. There are also some data-cleaning tools [19,46] that can facilitate users to clean data based on their domain knowledge. For example, OpenRefine [46] allows users to define facets on a per attribute basis, and helps them to quickly identify incorrect attribute values via faceted search.

2.3.2 *Identifying Duplicates*

The SampleClean framework defines the duplicate factor for a tuple as the number of times the tuple appears in the

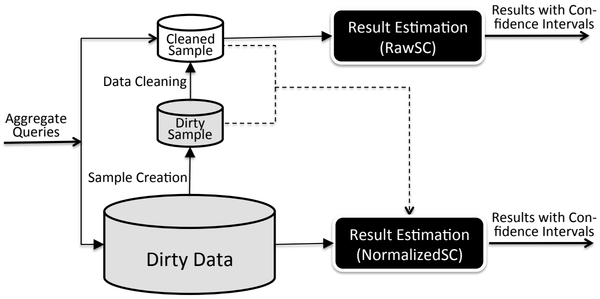

Aggregate Queries
Cleaned Sample
Data Cleaning
Dirty Sample
Sample Creation
Dirty Data
Result Estimation (RawSC)
Result Estimation (NormalizedSC)
Results with Confidence Intervals
Results with Confidence Intervals

Figure 2: The SampleClean framework.

entire table. To determine it, one way would be to estimate its value from the sample. However, both analytical proofs and empirical tests have shown that this method can lead to highly inaccurate query results [10]. Therefore, in our paper, we determine the duplication factor from the complete relation.

It is important to note, however, that compared to full cleaning, we only need to determine the duplication factor for those tuples in the sample. As with other uses of sampling, this can result in significant cost savings in duplicate detection. In the following, we will describe how to apply existing deduplication techniques to compute the duplication factor, and explain why it is cheaper to determine the duplication factor for a sample of the data, even though doing so requires access to the complete relation.

Duplicate detection (also known as entity resolution) aims to identify different tuples that refer to the same real-world entity. This problem has been extensively studied for several decades [22]. Most deduplication approaches consist of two phases:

1. *Blocking. Due to the large (quadratic) cost of all-pair comparisons, data is partitioned into a number of blocks, and duplicates are considered only within a block. For instance, if we partition* `papers` *based on* `conference_name`*, then only the papers that are published in the same conference will be checked for duplicates;*
2. *Matching. To decide whether two tuples are duplicates or not, existing techniques typically model this problem as a classification problem, and train a classifier to label each tuple pair as duplicate or non-duplicate [9]. In some recent research (and also at many companies) crowdsourcing is used to get humans to match tuples [20,54].*

A recent survey on duplicate detection has argued that the matching phase is typically much more expensive than the blocking phase [13]. For instance, an evaluation of the popular duplicate detection technique [9] shows that the matching phase takes on the order of minutes for a dataset of thousands of tuples [39]. This is especially true in the context of crowdsourced matching where each comparison is performed by a crowd worker costing both time and money. SampleClean reduces the number of comparisons in the matching phase, as we only have to match each tuple in the sample with the others in its block. For example, if we sample 1% of the table, then we can reduce the matching cost by a factor of 100.

2.3.3 Result Estimation

After cleaning a sample, SampleClean uses the cleaned sample to estimate the result of aggregate queries. Similar to existing SAQP systems, we can estimate query results directly from the cleaned sample. However, due to data error, result estimation can be very challenging. For example, consider the `avg(citation_count)` query in previous section. Assume that the data has duplication errors and that papers with a higher citation count tend to have more duplicates. The greater the number of duplicates, the higher probability a paper is sampled, and thus the cleaned sample may contain more highly cited papers, leading to an over-estimated citation count. We formalize these issues and propose the RawSC approach to address them in Section 3.

Another quantity of interest is how much the dirty data differs from the cleaned data. We can estimate the mean difference based on comparing the dirty and cleaned sample, and then correct a query result on the dirty data with this estimate. We describe this alternative approach, called NormalizedSC, and compare its performance with RawSC in Section 4.

SampleClean v.s. SAQP: SAQP assumes perfectly clean data while SampleClean relaxes this assumption and makes cleaning feasible. In RawSC, we take a sample of data, apply a data cleaning technique, and then estimate the result. The result estimation is similar to SAQP, however, we require a few additional scaling factors related to the cleaning. On the other hand, NormalizedSC is quite different from typical SAQP frameworks. NormalizedSC estimates the average difference between the dirty and cleaned data, and this is only possible in systems that couple data cleaning and sampling. What is surprising about SampleClean is that sampling a relatively small population of the overall data makes it feasible to manually or algorithmically clean the sample, and experiments confirm that this cleaning often more than compensates for the error introduced by the sampling.

2.3.4 Example: SampleClean with OpenRefine

In this section, we will walk through an example implementation of SampleClean using OpenRefine [46] to clean the data. Consider our example dirty dataset of publications in Figure 1(a). First, the user creates a sample of data (e.g., 100 records) and loads this sample into the OpenRefine spreadsheet interface. The user can use the tool to detect data errors such as missing attributes, and fill in the correct values (e.g., from another data source or based on prior domain expertise). Next, for deduplication, the system will propose potential matches for each publication in the sample based on a blocking technique and the user can accept or reject these matches. Finally, the clean sample with the deduplication information is loaded back into the dataset. In this example, sampling reduces the data cleaning effort for the user. The user needs to inspect only 100 records instead of the entire dataset, and has no more than 100 sets of potential duplicates to manually check.

To query this clean sample, we need to apply SampleClean's result estimation to ensure that the estimate remains unbiased after cleaning since some records may have been corrected, or marked as duplicates. In the rest of the paper, we discuss the details of how to ensure unbiased estimates, and how large the sample needs to be to get a result of acceptable quality.

3. RawSC ESTIMATION

In this section, we present the RawSC estimation approach. RawSC takes a sample of data as input, applies a data cleaning technique to the sample, runs an aggregate query directly on the clean sample, and returns a result with a confidence interval.

3.1 Sample Estimates

We will first introduce the estimation setting without data errors and explain some results about estimates from sampled data. We start with the table of N tuples which we call P, the *population*. From P, we sample a subset S of size K uniformly; that is, every tuple is sampled with equal probability. In this setting, we have two problems: (1) Estimate an aggregate query result on the sample S; (2) Quantify the uncertainty of the query result.

Consider a simpler problem; suppose we want to estimate the mean value of P. We can calculate the mean of S and the Central Limit Theorem (CLT) states that these estimates follow a normal distribution:

$$N(\texttt{mean}(P), \frac{var(P)}{K}) \tag{1}$$

Since the estimate is normally distributed, we can define a confidence interval parametrized by λ (e.g., 95% indicates $\lambda = 1.96$)[1].

$$\texttt{mean}(S) \pm \lambda\sqrt{\frac{var(S)}{K}}. \tag{2}$$

This interval has two interpretations: (1) if we re-compute the mean value on another random sample, the result will be within the confidence interval with the specified probability, and (2) the true value $\texttt{mean}(P)$ is within the confidence interval with the specified probability. Furthermore, we call this estimate *unbiased* since the expected value of the estimate is equal to the true value.

Our primary focus is answering **avg**, **count**, and **sum** queries with predicates[2] (see [25] for other queries). We can estimate these queries by re-formulating them as mean value calculations. We first define some notation:

- $f(\cdot)$: a function representing any of the supported aggregate queries with a predicate.
- $\texttt{Predicate}(t)$: the predicate of the aggregate query, where $\texttt{Predicate}(t) = 1$ or 0 denotes t satisfies or dissatisfies the predicate, respectively.
- K: the number of tuples in the sample.
- $K_{\texttt{pred}}$: the number of tuples that satisfy the predicate in the sample.
- $t[a]$: the aggregation-attribute value. If it is clear from the context that t refers to an attribute value rather than a tuple, we will omit '$[a]$' for brevity.

We can reformulate all of the queries as calculating a mean value so we can estimate their confidence intervals with the CLT:

$$f(S) = \frac{1}{K}\sum_{t\in S}\phi(t) \tag{3}$$

where $\phi(\cdot)$ expresses all of the necessary scaling to translate the query into a mean value calculation:

- count: $\phi(t) = \texttt{Predicate}(\texttt{t}) \cdot N$
- sum: $\phi(t) = \texttt{Predicate}(\texttt{t}) \cdot N \cdot t[a]$
- avg: $\phi(t) = \texttt{Predicate}(\texttt{t}) \cdot \frac{K}{K_{\texttt{pred}}} \cdot t[a]$

For example, the avg query is estimated from the sample as:

$$\texttt{avg}(S) = \frac{1}{K_{pred}}\sum_{t\in S}\texttt{Predicate}(t) \cdot t[a], \tag{4}$$

which computes the average value of the tuples that satisfy the predicate in the sample. In order to represent $\texttt{avg}(S)$ in the form of Equation 3, we rewrite it to the following equivalent Equation:

$$\texttt{avg}(S) = \frac{1}{K}\sum_{t\in S}\texttt{Predicate}(t) \cdot \frac{K}{K_{\texttt{pred}}} \cdot t[a]. \tag{5}$$

Therefore, we have $\phi(t) = \texttt{Predicate}(\texttt{t}) \cdot \frac{K}{K_{\texttt{pred}}} \cdot t[a]$ for the avg query.

3.2 Unbiased Estimation with Data Errors

If we ignore data errors, the estimates described in the previous section are unbiased. Suppose $P_{\texttt{clean}}$ is the corresponding clean population for the dirty data population P. We are interested in estimating an aggregate query on $P_{\texttt{clean}}$. However, since we do not have the clean data, we cannot directly sample from $P_{\texttt{clean}}$. We must draw our sample from the dirty data P and then clean the sample.

The key question is whether running an aggregate query on the cleaned sample is equivalent to computing the query result on a sample directly drawn from the clean data. When this is true, our estimate is unbiased, and we can derive confidence intervals using the CLT. In the following section, we explore this question on different types of data errors. 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.

Recall, that we model three types of errors: value error, condition error, and duplication error. To correct the errors, we can define two data cleaning primitive functions:

- $\texttt{Correct}(\texttt{t})$: for the tuple t return a tuple with correct attribute values
- $\texttt{Numdup}(\texttt{t})$: for the tuple t return the number of times that the tuple appears in the population P

3.2.1 Value and Condition Errors

Both value and condition errors are caused by incorrect attribute values of the dirty data. These errors do not affect the size of the population, i.e., $|P| = |P_{\texttt{clean}}|$. Furthermore, correcting a value or condition error only affects an individual tuple. 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 our correction of value and condition errors, thus we define $\phi_{\texttt{clean}}(t)$ as:

$$\phi_{\texttt{clean}}(t) = \phi\left(\texttt{Correct}(t)\right). \tag{6}$$

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.

3.2.2 Duplication Error

Since duplication errors affect multiple tuples and the size of $P_{\texttt{clean}}$ is different from the size of P, they do affect the uniformity of the sampling. The following example illustrates the consequences of duplication errors:

EXAMPLE 1. *Consider a population $P = \{t_1, t_2, t_2'\}$ with two distinct tuples, t_1 and t_2 (=t_2'). If we draw samples of size 2 from this population uniformly:*

$$Pr(\{t_1, t_2\}) = \frac{1}{3},\ Pr(\{t_1, t_2'\}) = \frac{1}{3},\ Pr(\{t_2, t_2'\}) = \frac{1}{3}.$$

Now, assume $t_1 = 1$ and $t_2 = t_2' = 2$. The expected mean value over all random samples is $\frac{1}{3} \cdot \frac{3}{2} + \frac{1}{3} \cdot \frac{3}{2} + \frac{1}{3} \cdot 2 = \frac{5}{3}$, however the cleaned population is $P_{clean} = \{t_1, t_2\}$ and its mean value is actually $\frac{3}{2}$.

[1] When estimating means of finite population there is a finite population correction factor of $FPC = \frac{N-K}{N-1}$ which scales the confidence interval.

[2] Group-by queries can be implemented by adding group-by keys into predicates.

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 t is duplicated $m = \texttt{Numdup}(t)$ 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:

LEMMA 1. *Let P be a population with duplicated tuples. Let $S \subseteq P$ be a uniform sample of size K. For each $t_i \in S$, let m_i denote its number of duplicates in P. (1) For `sum` and `count` queries, applying $\phi_{clean}(t_i) = \frac{\phi(t_i)}{m_i}$ 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 $\phi_{clean}(t_i) = d \cdot \frac{\phi(t_i)}{m_i}$ yields an unbiased estimate.*

PROOF SKETCH. We can interpret the population as a discrete probability distribution, and the sample as drawing K elements from this distribution. Since some elements are duplicated there is an increased probability of drawing these elements. We reduce these effects by re-weighting the samples, which can be thought of as drawing fractional samples (i.e., if we sample an element which is duplicated twice, we cancel it out by treating it as sampling only half an element). As a result, while we may draw K total samples, the total number of fractional samples drawn (sum of the weights) K', may be different, so we have to scale the result accordingly. See [25] for a detailed proof. □

We apply this lemma to the following example:

EXAMPLE 2. *Consider the dirty data in Figure 1, $P = \{t_1, t_2, \cdots, t_{10000}\}$, and the sample data, $S = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. In this example, we assume the data only has duplication error, and our goal is to estimate the average citation count of all the papers in the data.*

Firstly, we compute the duplication rate of the sample:

$$\frac{K}{\sum_{t \in S} \frac{1}{Numdup(t)}} = \frac{7}{\frac{1}{2}+\frac{1}{1}+\frac{1}{2}+\frac{1}{1}+\frac{1}{1}+\frac{1}{1}+\frac{1}{3}} = 1.31$$

Then, we apply $\phi_{clean}(\cdot)$ to each paper $t \in S$. Specifically, we divide its citation count by the number of duplicates and then multiple it by the duplication rate, i.e., $\phi_{clean}(S) = \{\frac{1.31 \cdot 18}{2}, \frac{1.31 \cdot 1569}{1}, \frac{1.31 \cdot 298}{2}, \cdots, \frac{1.31 \cdot 687}{3}\}$. For example, the first paper's citation count is 18 and it has two duplicates, thus we have $\phi_{clean}(t_1) = \frac{1.31 \cdot 18}{2}$.

Finally, we estimate the average citation count as the `mean` of $\phi_{clean}(S)$.

3.2.3 Combinations of Errors

We can also address data with combinations of errors (e.g., duplicated tuples that also have incorrect values). Value and condition errors affect a tuple's values. Duplication errors affect a tuple's sampling probability. The two classes of errors affect the tuple in different ways, and consequently, we define a single function $\phi_{\texttt{clean}}(\cdot)$ which can correct for all three error types:

- `count`: $\phi_{\texttt{clean}}(t) = \frac{\phi(\texttt{Correct}(t))}{\texttt{Numdup}(t)}$
- `sum`: $\phi_{\texttt{clean}}(t) = \frac{\phi(\texttt{Correct}(t))}{\texttt{Numdup}(t)}$
- `avg`: $\phi_{\texttt{clean}}(t) = d \cdot \frac{\phi(\texttt{Correct}(t))}{\texttt{Numdup}(t)}$

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`	$\texttt{Predicate}(\texttt{Correct}(\texttt{t})) \cdot N \cdot \frac{1}{\texttt{Numdup}(t)}$
`sum`	$\texttt{Predicate}(\texttt{Correct}(\texttt{t})) \cdot N \cdot \frac{\texttt{Correct}(t)[a]}{\texttt{Numdup}(t)}$
`avg`	$\texttt{Predicate}(\texttt{Correct}(\texttt{t})) \cdot \frac{dK}{K_{\texttt{pred}}} \cdot \frac{\texttt{Correct}(t)[a]}{\texttt{Numdup}(t)}$

We plug $\phi(\cdot)$ (as described in Section 3.1) into the above equations, and obtain a more detailed form of $\phi_{\texttt{clean}}(t)$ as shown in Table 1.

RawSC Estimation: We can now formulate the RawSC estimation procedure, as follows:

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 $\mu_c \pm \lambda\sqrt{\frac{\sigma_c^2}{K}}$

To summarize, we state that the RawSC approach gives an unbiased estimate:

THEOREM 1. *Given an aggregation function f and a population P, where there are three types of errors: value, condition, and duplication. Let S be a uniform sample $S \subseteq P$ of size K. Let $\phi_{clean}(S)$ be the set of tuples where $\phi_{clean}(\cdot)$ is applied to every tuple. Then the estimate on this sample is given by:*

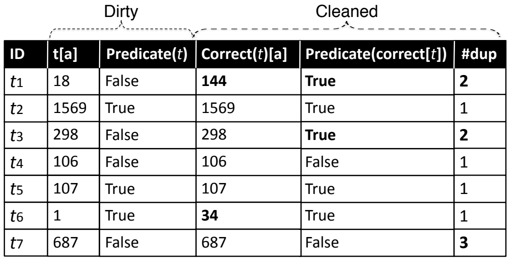
$$\texttt{mean}\,(\phi_{clean}(S)) = \frac{1}{K} \sum_{t \in S} \phi_{clean}(t)$$

The estimate is an unbiased estimate of $f(P_{clean})$.

PROOF SKETCH. We need to show that the aggregation $\phi_{\texttt{clean}}(S)$ is equivalent to the aggregation $\phi(S_{\texttt{clean}})$. Lemma 1 shows that this is true for duplication errors. We further argued that with value errors and condition errors $\phi_{\texttt{clean}}(S) = \phi(S_{\texttt{clean}})$. Finally, since duplication error correction and value/condition correction can be composed this is true. As the `mean` of a uniform random sample of $P_{\texttt{clean}}$, this is an unbiased estimate. See [25] for a detailed proof. □

Consider the following end-to-end numerical example with RawSC:

EXAMPLE 3. *Consider the sample data and the cleaning results in Figure 3, and assume the data may involve three types of errors. To estimate the query result, we need to map each $t \in S$ to a new value $\phi_{clean}(t)$. Since the sample contains seven papers, and t_4 and t_7 do not satisfy the predicate, the scaling for predicates is $\frac{K}{K_{pred}} = \frac{7}{5} = 1.40$. As shown in Example 2, the duplication rate is $d = 1.31$. After applying $\phi_{clean}(\cdot)$ for `avg` query in Table 1 to each sampled paper, we obtain the transformed sample data of $\phi_{clean}(S) = \{133, 2895, 275, 0, 197, 63, 0\}$. For example, since t_1 satisfies the predicate, and its correct citation count is 144 and the number of duplicates is 2, we have $\phi_{clean}(t_1) = 1.31 \cdot 1.40 \cdot \frac{144}{2} = 133$. Similarly, as t_4 does not satisfy the predicate, we have $\phi_{clean}(t_4) = 0$. We calculate the `mean` μ_c and the variance σ_c^2 of $\phi_{clean}(S)$, and return $\mu_c \pm \lambda\sqrt{\frac{\sigma_c^2}{K}}$ as the estimated average citation count, where λ is a constant value derived from the user-specified confidence probability.*


Dirty
Cleaned

ID	t[a]	Predicate(t)	Correct(t)[a]	Predicate(correct[t])	#dup
t_1	18	False	**144**	**True**	**2**
t_2	1569	True	1569	True	1
t_3	298	False	298	**True**	**2**
t_4	106	False	106	False	1
t_5	107	True	107	True	1
t_6	1	True	**34**	True	1
t_7	687	False	687	False	**3**

Figure 3: The cleaning results for the sample $S = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ w.r.t "SELECT AVG(citation_count) FROM papers WHERE pub_year>2000". The values in bold font indicate the changed values after data cleaning.

Remarks. (1) For an avg query, we can achieve tighter confidence intervals by skipping the tuples that dissatisfy the predicate instead of considering them as 0 values. The reason for this is that the avg query has a scaling factor $r = \frac{K}{K_{\texttt{pred}}}$, which is a random variable itself. The confidence intervals we present incorporate the additional variance of r, but due to the skipping we can get estimates not affected by that variance. (2) Algorithmically, we contrast RawSC from SAQP with $\phi_{\texttt{clean}}(\cdot)$ vs. $\phi(\cdot)$, which makes it very convenient to implement RawSC in an existing SAQP system.

4. NormalizedSC ESTIMATION

RawSC estimates a result directly on a clean sample of data. The size of confidence intervals in the RawSC estimate are function on the variance of the cleaned sample $var(\phi_{\texttt{clean}}(S))$ and the sample size K. This property implies that the accuracy of RawSC is only dependent on the cleaned values (independent of the magnitude of incorrect values), and thus makes the technique robust to large errors. This dependence may not be desirable in datasets where the data itself has high variance or where errors are small in magnitude.

This motivates the NormalizedSC approach, where we take an existing aggregation of the data, estimate its difference from the true value, and then use the estimate to correct the existing aggregation. The resulting technique gives us confidence intervals that are dependent on the variance of the errors, which can allow us to estimate very accurately on datasets where this variance is small. Furthermore, we provide the same unbiased guarantees on NormalizedSC as RawSC.

4.1 Estimating the Difference

Due to data errors, the result of the aggregation function f on the dirty population P differs from the true result by ϵ:

$$f(P) = f(P_{\texttt{clean}}) + \epsilon$$

In the previous section, we derived a function $\phi_{\texttt{clean}}(\cdot)$ for RawSC estimation. We contrasted this function with $\phi(\cdot)$ which does not clean the data. Therefore, we can write:

$$f(P) = \frac{1}{N}\sum_{t\in P}\phi(t) \qquad f(P_{\texttt{clean}}) = \frac{1}{N}\sum_{t\in P}\phi_{\texttt{clean}}(t) \qquad (7)$$

If we solve for ϵ, we find that:

$$\epsilon = \frac{1}{N}\sum_{t\in P}\Big(\phi(t) - \phi_{\texttt{clean}}(t)\Big) \qquad (8)$$

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

$$Q = \{\phi(t_1) - \phi_{\texttt{clean}}(t_1), \phi(t_2) - \phi_{\texttt{clean}}(t_2), \cdots, \ \phi(t_K) - \phi_{\texttt{clean}}(t_K)\}$$

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

NormalizedSC Estimation We derive the NormalizedSC 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 $t_i \in S$ and call the set of differences $Q(S)$.
3. Calculate the mean μ_q, and the variance σ_q of $Q(S)$
4. Return $(f(P) - \mu_q) \pm \lambda\sqrt{\frac{\sigma_q^2}{K}}$

Similar to RawSC, we can prove that the result in unbiased:

Theorem 2. *Given an aggregation function f and a population P, where there are three types of errors: value, condition, and duplication. Let S be a uniform sample $S \subseteq P$ of size K. Let Q be the set of tuples where $q(t) = \phi(t) - \phi_{clean}(t)$ is applied to every tuple. Then the estimate on this sample is given by:*

$$mean(Q) = \frac{1}{K}\sum_{t\in S}\Big(\phi(t) - \phi_{clean}(t)\Big)$$

is an unbiased estimate of the bias ϵ. It follows, that $f(P) - \epsilon$ is an unbiased estimate of the result.

Proof sketch. We can apply the theory developed for RawSC to the estimate of ϵ. Since it is unbiased, due to the linearity of expectation the estimate $f(P) - \epsilon$ is also unbiased. See [25] for a detailed proof. □

Consider the Example 3 discussed in the previous section.

Example 4. *Based on the definition of $\phi(\cdot)$ in Section 3.1, we have*

$$\phi(S) = \{0, \ 3661, \ 0, \ 0, \ 250, \ 2, \ 0\}.$$

For example, as shown in Figure 3, since t_1 does not satisfy the predicate, we have $\phi(t_1) = 0$. Additionally, as t_2 satisfies the predicate, and its dirty citation is 1569 and the scaling for predicates is $\frac{K}{K_{pred}} = \frac{7}{3}$, we have $\phi(t_2) = \frac{7}{3} \cdot 1569 = 3661$.

Over the same data we apply $\phi_{clean}(\cdot)$ (as shown in Example 3)

$$\phi_{clean}(S) = \{133, \ 2895, \ 275, \ 0, \ 197, \ 63, \ 0\},$$

we can obtain the difference between the two samples

$$Q(S) = \{-133, \ 766, \ -275, \ 0, \ 53, \ -61, \ 0\}.$$

We calculate the mean *μ_q and the variance σ_q^2 of $Q(S)$, and return $(f(P) - \mu_q) \pm \lambda\sqrt{\frac{\sigma_q^2}{K}}$.*

4.2 NormalizedSC vs. RawSC

We compare RawSC and NormalizedSC on result accuracy and processing time. Both methods achieve unbiased estimates, but may differ greatly in the accuracy (the size of the confidence interval) of these estimates. The confidence interval of NormalizedSC is given by $\pm\lambda\sqrt{\sigma_q^2/K}$ and for RawSC it is $\pm\lambda\sqrt{\sigma_c^2/K}$. Therefore, for a fixed sample size, if $\sigma_c \geq \sigma_q$, NormalizedSC will be more accurate. In cases when either σ_c is large or σ_q is small, NormalizedSC can give a result with narrower confidence intervals than RawSC. For example, if we had no data errors then $\sigma_q = 0$ (a perfect NormalizedSC estimate), but σ_c would still be non-zero as it is the variance of the cleaned data. Conversely, the more unpredictable (high variance in the difference between the clean and dirty data) the error offset is, the worse NormalizedSC will perform.

In terms of processing time, NormalizedSC is very different from RawSC as it needs to run an aggregation query on the entire dataset. We highlight two situations that are not affected by this additional processing. (1) When the time required to clean a sample is much larger than the time needed to scan the entire dataset. (2) When the user has an existing result on the dirty dataset and wants to assess how far away it is from the true value. In these settings, we can maintain results for both RawSC and NormalizedSC and return the result with a tighter confidence interval.

The analysis in this section assumes that we can compute an exact aggregation result on the dirty dataset. We can also extend NormalizedSC by estimating the aggregate result based on a sample without accessing the entire data. In this way, NormalizedSC requires less response time but may lose some result accuracy. Interested readers are referred to [25] for details.

5. EXPERIMENTS AND RESULTS

We conducted a set of experiments on real and synthetic data to evaluate our framework. We designed our experiments to evaluate the following characteristics: (1) Comparing RawSC with NormalizedSC by varying the amount of each type of error (value, condition, duplication). (2) Exploring the trade-off between result quality and cleaning cost provided by RawSC and NormalizedSC. (3) Measuring the required cleaning cost to achieve a certain accuracy by varying data size. (4) Comparing with existing approaches that either clean all of data or none of data. (5) Evaluating our framework on real datasets.

5.1 Experimental Settings and Datasets

We evaluate the efficacy of our approach with the following quantities: (1) *Number of Cleaned Samples.* The number of tuples that are sampled and cleaned to produce an estimate; (2) *Error %.* An error % of q% signifies that with 95% probability the estimate is within $\pm q\%$ of the true value.

We refer to our framework SampleClean as the one that can dynamically choose the better result between NormalizedSC and RawSC. We compare SampleClean with existing solutions that either clean none of the data (AllDirty) or clean all of data (AllClean). To compare different types of errors, we classify them by the error rate (the number of tuples affected by the error) ranging from 0% (all of the tuples clean) to 100% (all of the tuples dirty) for value, condition, and duplication errors.

5.1.1 TPC-H Dataset

We generated a 1GB TPC-H benchmark[3] dataset (6,001,199 Records in `lineitem` table). The `lineitem` table schema simulates industrial purchase order records. We used this dataset to model errors where the purchase orders were digitized using optical character recognition (OCR). We denote a value-error percentage a% as a% of digits in the database were recognized as their most likely OCR false positive digit. For condition errors, we simulated missing values by randomly selecting p% of tuples and removing their predicate attribute. We also randomly duplicated d% of tuples with the following distribution: 80% one duplicate, 15% two duplicates, 5% three duplicates.

For this dataset, we experiment with `avg`, `count`, and `sum` aggregations applied to this query:

```
SELECT f(quantity) FROM lineitem
WHERE returnflag = ‘A’ AND linestatus = ‘F’;
```

which finds aggregate quantities of purchases that satisfy simulated conditions (returnflag = ‘A’ and linestatus = ‘F’). In the clean data, there are 1,478,493 tuples satisfying the conditions that corresponds to a 24.6% selectivity of this query.

5.1.2 Microsoft Academic Search Dataset

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 [54] 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 Table 2.

Table 2: Microsoft Academic Search Dataset.

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

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 323 respectively). Jeffery Ullman had a comparatively larger number of duplicated papers (182).

5.1.3 Sensor Dataset

We also applied our approach to a dataset of indoor temperature, humidity, and light sensor readings in the Intel Berkeley Research Lab. The dataset is publicly available for data cleaning and sensor network research from MIT CSAIL[5]. We selected one month of readings and aggregated the values of all the sensors into a single aggregate reading for each minute in the month. The resulting dataset had 44,460 sensor readings spaced one minute apart. We applied algorithmic cleaning techniques (as opposed to manual cleaning) as described in [35] to a sample of data.

5.2 RawSC vs. NormalizedSC

We evaluated RawSC and NormalizedSC for a fixed cleaned sample size, and for each type of error, we varied the error percentage. This illustrates the regimes in which our framework will have good performance since we can apply the more accurate of the two approaches. For all of our TPC-H experiments, we cleaned a fixed set of 10,000 samples (0.17% of all tuples), and evaluated the performance on the `avg` query.

In Figure 4(a), we explored `avg` queries on tuples with only value errors. We find that NormalizedSC gives results with narrower confidence intervals when the value errors are

[3] http://www.tpc.org/tpch

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

[5] http://db.csail.mit.edu/labdata/labdata.html

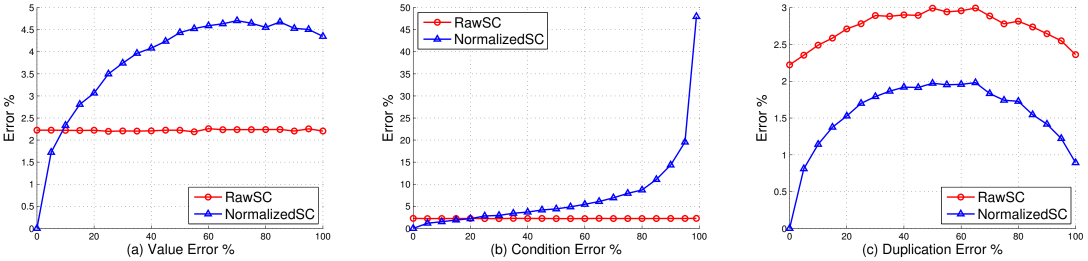


Figure 4: Varying the amount of each type of error on the TPC-H dataset and measured the performance of NormalizedSC vs. RawSC.

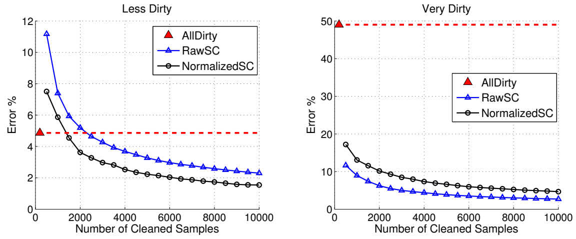


Figure 5: Comparison of the convergence of the methods on two TPC-H datasets of 6M tuples with simulated errors (30% value, 10% condition, 20 % duplication) and (3% value, 1% condition, 2% duplication). On the dataset with larger errors, we find that RawSC gives a narrower confidence interval, and on the other NormalizedSC is more accurate.

small. RawSC, on the other hand, is actually independent of the value error rate as it only reports the aggregation of clean data. Accordingly, since our framework dynamically chooses the more accurate approach, we can give tight bounds for datasets with higher and lower error rates.

We repeat the same experiment with condition errors in Figure 4(b) and observed similar behavior. For small error rates, NormalizedSC gives a narrower confidence interval than RawSC, but RawSC returns a result that is not dependent on the rate of errors.

Figure 4(c) compares the `avg` query results of RawSC and NormalizedSC on the data with duplication errors. Consistent with all of our other experiments, NormalizedSC was more accurate when there were a small number of duplicates. A counter-intuitive result is that the estimation error actually decreases after a point for both RawSC and NormalizedSC. To better understand this phenomenon, suppose every tuple had exactly one duplicate, then `avg` query would be correct even on the dirty data.

5.3 Cleaning Cost v.s. Result Quality

We evaluated how the number of cleaned samples affects the result error for RawSC and NormalizedSC. We ran the `avg` query on the two TPC-H datasets that we considered before: one where the percentages for each error type were (30%,10%,20%), and one where each was (3%,1%,2%). Our results are shown in Figure 5.

Both of the methods converge at a rate $\frac{1}{\sqrt{K}}$ w.r.t. the sample size. We can easily estimate how many samples we need to clean to achieve a specified error. Another key insight is that since both converge at the same rate with respect to the sample size, the lines will never cross. Consequently, there will always be a single *better* choice between RawSC and NormalizedSC, and we do not have to worry about our choice being suboptimal for more cleaned samples.

In comparison to AllDirty, we can see both RawSC and NormalizedSC achieved a better result by cleaning only a small number of samples. Section 5.5 contains a more detailed comparison with AllDirty and AllClean.

5.4 Scalability of Cleaning Cost

For both RawSC and NormalizedSC, we return confidence intervals that are independent of the size of the dataset. In terms of tuples cleaned, there is no difference between evaluating our approach on a 1GB dataset or on a 1TB dataset. In Figure 6, we show that if we simulate the 30%, 10%, 20% errors in different sized TPC-H datasets (1GB, 10GB, 100GB, 1TB) as before, the number of cleaned tuples needed for a certain accuracy remains roughly constant. Each dataset was generated in the same way; the errors were simulated from the same distribution. It underscores that the number of samples needed to get a good estimate is a property of the variance of the data and its errors.

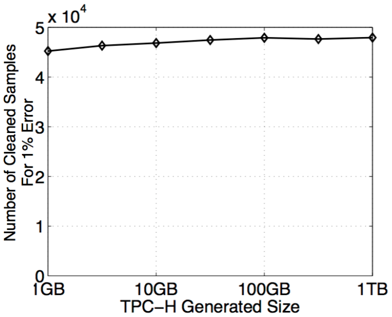


Figure 6: Scalability of SampleClean on TPC-H (30%,10%,20%) of different sizes. The number of cleaned tuples needed to achieve a certain accuracy does not increase with the size of the dataset.

5.5 End-to-End Experiment

We tested the end-to-end performance of the SampleClean framework on a TPC-H dataset with small errors: 3% value, 2% duplication, and 1% condition errors. Under these errors, we evaluated the `avg`, `sum`, and `count` aggregations and compared the performance of our framework with AllClean and AllDirty in terms of result quality and cleaning cost. To process the queries (refer to Section 5.1.1), we uniformly sampled from the dataset. Our results in Figure 7 suggest that SampleClean quickly converges to the right answer, giving a flexible trade-off between tuples cleaned and the size of the confidence interval in the estimate. We found that after cleaning only 1000 tuples (0.016%), we were to estimate more accurately than AllDirty.

This was a dataset with small errors (mostly clean), and we wanted to understand how SampleClean performs on a much dirtier dataset. We repeated experiment under 30% value, 20% duplication, and 10% condition errors (Figure 8). On this dataset AllDirty differs from AllClean by 52% in the `avg` function. The results in Figure 8 show that RawSC

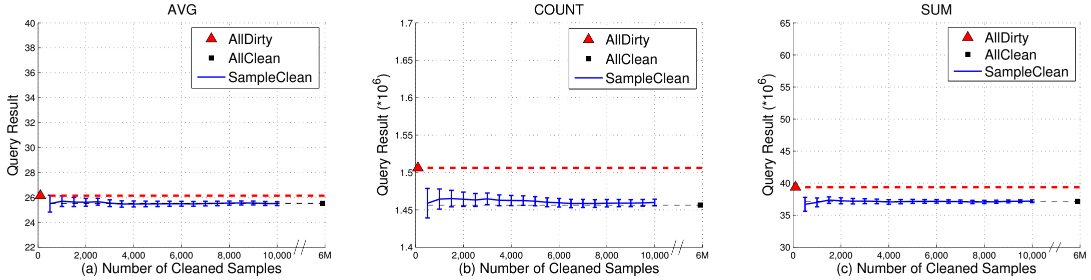


Figure 7: TPC-H evaluation with 3% Value Error, 1% Condition Error, 2% Duplication Error on 6M tuples. Our technique can efficiently estimate on datasets with a small number of errors due to the trade-off between RawSC and NormalizedSC.

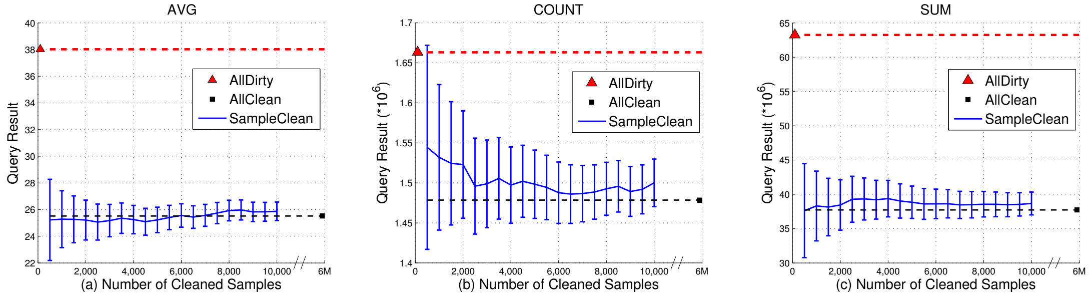


Figure 8: TPC-H evaluation with 30% Value Error, 10% Condition Errors, 20% Duplication Error on 6M tuples. We find that even for a small number of cleaned samples, we can return results with high confidence. We are able to achieve 95% accuracy after cleaning only 0.08% of the total samples.

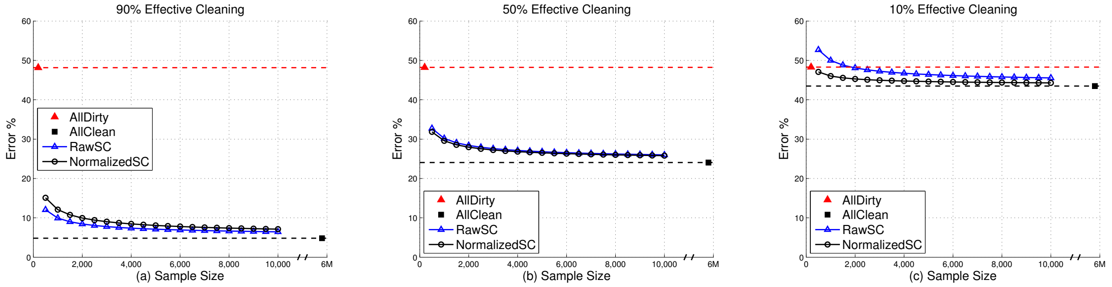


Figure 9: Imperfect Data Cleaning on TPC-H (30%, 10%, 20%) of 6M tuples. Even with a data cleaning technique that is not always correct, we find that our results rapidly converge to the AllClean values. We found that a 10% effective cleaning module can give more accurate results than AllDirty with a sample size of 0.03% of the dataset.

still rapidly converges to the true values and outperforms AllDirty. In addition, for all queries, our estimate is within 5% of AllClean after cleaning only 5000 tuples (0.08% of the total data).

Due to the trade-off between NormalizedSC and RawSC, we can estimate accurately for datasets that are both mostly clean or very dirty. On the dataset with small errors, NormalizedSC was the more accurate method and RawSC was more accurate on one in the presence of larger errors.

5.6 Imperfect Data Cleaning

In Figure 9, we experimentally show that SampleClean is valuable even when the cleaning is incomplete. We ran our experiments on a dataset with 30% Value Errors, 10% Condition Errors, and 20% Duplication Errors, and simulated a data cleaning technique that can only identify and clean a fixed percentage of errors. We compared the Error % of the query and the total sample size including tuples where the data cleaning failed.

Even though our estimates are now biased with respect to the true values, they are still unbiased with respect to AllClean. Consequently, we found that even if our cleaning technique can clean only 10% of the errors, we still only have to sample 2,000 tuples to achieve a more accurate result than AllDirty. Furthermore, we can take advantage of both RawSC and NormalizedSC. If a technique changes a lot of the tuple values, then RawSC gives us a more precise estimate. However, if a technique keeps most of the values the same NormalizedSC is more precise.

5.7 Evaluation on Real Data

The following sections contain experiments on two real datasets: Microsoft Academic Search and the Intel Lab Sensors. As described in Section 5.1, they contain different types of errors: the sensor dataset consists of predominantly value and condition errors, and the publication dataset consists of condition and duplication errors.

5.7.1 *Microsoft Academic Search*

We designed this experiment to be illustrative of assessing the confidence of decisions based on dirty data, by trying to get an accurate publication count for the three authors. We can see that in Table 2 AllDirty not only returns incorrect values but also returns an incorrect ranking of the three au-

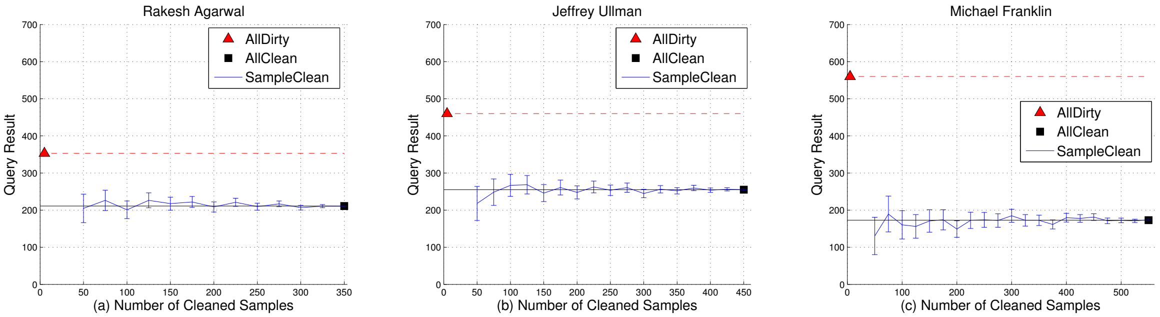


Figure 10: The output of our result estimation framework for each author. The dataset was particularly dirty and cleaning only 50 tuples per author was sufficient to outperform an aggregation of the entire dataset.

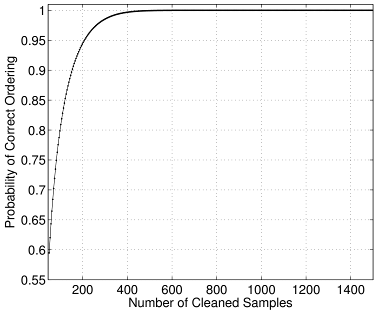


Figure 11: Even though AllDirty always returns an incorrect ranking, we can return the correct ranking with 95% probability after cleaning only 210 total samples. To achieve a correct ranking with 99% probability, we require 326 samples to be cleaned.

thors. This is an example of a dataset where the data cleaning is quite well defined, we can apply our domain expertise in database publications to reason about each tuple in the dataset, but cleaning is very time consuming. In Figure 10, we show how our approach can budget data cleaning for each author up-to a desired query quality. We clearly see that our estimates are centered around the AllClean value and even with the confidence intervals outperform aggregations of the full dirty data.

However, we do notice that the confidence intervals for the estimated paper counts overlap for a small number of cleaned samples. We evaluated how many tuples would need to be cleaned to get an accurate ranking using a simulation; that is if we re-ran the same experiment with the same sample size what is the probability our ordering of the authors is accurate. For each author, we treat the estimate of their paper count as a normally distributed random variable. From these distributions, we sample 10000 paper counts and calculate the empirical probability of an incorrect ordering. In Figure 11, 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.

5.7.2 Sensor Dataset

The sensor dataset consists of readings from inexpensive battery-operated sensors. The failure mode of these sensors is returning incorrect readings, particularly when the available battery power is low. We limited the dataset to a single month where the readings from the beginning of the month were largely accurate, while the readings from later in the month became increasingly inaccurate. We cleaned 500 samples (1.12%) of the dataset using the algorithm described in [35] and applied a variety of queries to illustrate how different types of value and condition errors can affect results (Figure 12).

Like the TCP-H results, this experiment shows how having both RawSC and NormalizedSC helps us estimate accurately in different error distributions. Surprisingly, we found that different queries on the same dataset may have very different error characteristics making the choice between RawSC and BiasCorrect very important. From this experiment, we can also see the interactive latency capabilities of SampleClean. Since we cleaned the 500 sampled tuples, subsequent RawSC queries only need to operate on the cleaned sample. This demonstrates how RawSC can give a very fast response time on large datasets as it only has to process the tuples in the cleaned sample.

The first two queries `avg`(temp) and `avg`(light) illustrate the severity of the sensor errors. Simply taking an average over the dirty data results in an estimate that differs from AllClean by about 100%. However, for the cost of cleaning only 500 samples, we are able to achieve a confidence interval of ±1.5% and ±15.2% respectively. Due to the nature of the sensor errors, a similar, but more dramatic, improvement is seen for `var`(temp) and `var`(light). Random errors such as ones generated by electronic sensors tend to increase variance of the data, and we can see that our estimate for `var`(temp) is three orders of magnitude closer to AllClean.

We also evaluated the queries with predicates. In the first query, we looked at the average temperature when the humidity was above 30%. For this query, there were both value and condition errors. We also looked at the average temperature when the predicate attribute (time) was accurate even during sensor failures. We found that our technique works well in both cases giving significant improvements over aggregations of the dirty data.

Finally, we demonstrate a query where the errors are relatively small. We counted the number of readings where the temperature was above some threshold. We found that the dirty aggregation was not very much worse than the confidence intervals returned by our method. Furthermore, we also experimented with the average temperature over the first four days when most of the sensors were working. In this query, AllDirty differed from AllClean by less than 1%. In fact, AllDirty outperforms RawSC for 500 cleaned samples. However, due to our tradeoff between RawSC and NormalizedSC, we are still able to improve on already good estimates.

6. RELATED WORK

Approximate Query Processing: SAQP has been studied for more than two decades [16,26]. Many SAQP approaches [2,3,5,11,14,32,47,50,57] were proposed, aiming to

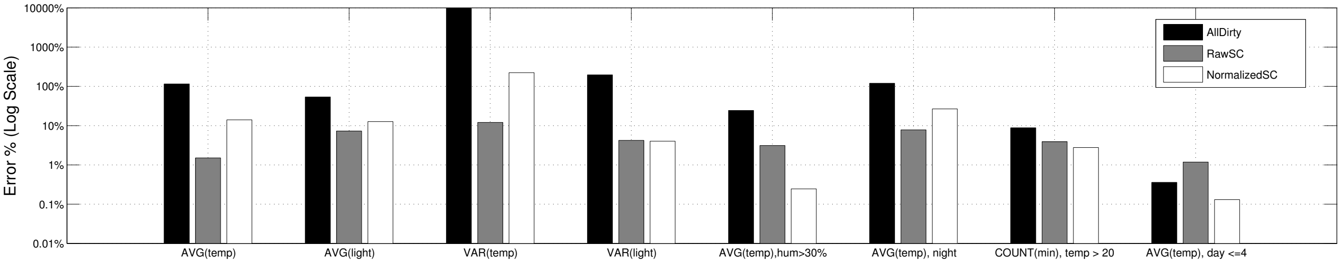


Figure 12: Queries applied to the sensor dataset. We applied a set of queries with a fixed sample (same for all queries) of 500 cleaned tuples. We compare the error percentage (log-scale) of AllDirty, RawSC, and NormalizedSC with respect to AllClean. For all of these example queries, we are able to achieve a relative error of less than ±10% even when the data error is orders of magnitude higher.

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 [17, 23], outlier detection [18,31], filling missing values [48,52], and duplicate detection [9,12]. In order to ensure reliable cleaning results, most of these techniques require human involvement [24,36,54,55,58]. In our paper, 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.

Result Estimation and Sampling: Estimating aggregate statistics of populations from samples has been well studied in the field of surveying [7,29,38,49,53,56]. These works explore different types of sampling, error characterizations, bias-variance trade-offs, and confidence intervals. The theoretical foundation of surveying is statistical bounding of functions of independent random variables (e.g., samples with replacement). This field includes distribution-free bounds such as Markov/Chebyshev/Hoeffding bounds, asymptotic bounds such as the Central Limit Theorem (CLT), and empirical testing such as Bootstrapping [33]. For a detailed survey of different types of statistical bounds refer to [28]. Due to the CLT's strong guarantees (unbiased sample estimates and normalcy), as in our work, it is widely applied in the analysis of sampling schemes. The more general study of sample estimators that are unbiased and have asymptotically normal distributions (like the CLT) is called U-Statistics, refer to [41] for a survey of this theory. The stochastic process literature also discusses problems in unbiased estimation from samples [34].

Distinct Value Estimation: Distinct value estimation has been an open problem in stream processing and database research [6,8,15,27]. Similar to our analysis, some have argued that simply removing duplicates in a sample of data does not work [10]. In the storage literature, similar weighted average techniques have been applied for global file duplication rate estimation [30] based on the knowledge of duplication rates within a sample. This work can be seen as a simplified version of our problem only answering a count query with only duplication errors. Techniques similar to our duplicate reweighting scheme have been studied in estimating from non-uniform samples [4], and is also similar to the acceptance ratio used sampling algorithms such as the Metropolis-Hastings Algorithm and Rejection Sampling [42,45]. Further relevant work includes sensitivity analysis of set functions [37,44] and statistical information theory [40].

7. CONCLUSION AND FUTURE WORK

In this paper, we explore using sampling, integrated with data cleaning, to improve answer quality. We propose SampleClean, a novel framework which only requires users to clean a sample of data, and utilizes the cleaned sample to obtain unbiased query results with confidence intervals. We identify three types of data errors (i.e., value error, condition error and duplication error) that may affect query results, and develop NormalizedSC and RawSC to estimate query results for the data with these errors. Our analysis and experiments suggest that SampleClean, which returns the better result between NormalizedSC and RawSC, is robust to different magnitudes and rates of data errors, and consistently reports good estimate results. Our experiments on both real and synthetic data sets indicate that SampleClean only needs to clean a small sample of the data to achieve accurate results, and furthermore the size of this sample is independent of the size of the dataset. In particular, RawSC, which processes queries only on the cleaned sample, not only makes the query processing more scalable, but surprisingly may provide higher quality results than an aggregation of the entire dirty data (AllDirty).

To the best of our knowledge, this is the first work to marry data cleaning with sampling-based query processing. There are many research directions for future exploration.

Constrained Queries: Now that we have quantified a tradeoff between cleaning costs and result quality, we can explore query results where users can specify a cost or quality constraint. For example, users may want to know that given a cleaning budget, what is the best result quality they can achieve? Or, given a quality constraint, how many samples they need clean to meet the constraint? In these constrained queries, we aim to answer with an optimal cost or most accurate result to meet the constraints.

Uncertain Cleaning Results: Our framework can return unbiased query results with respect to AllClean for a variety of different data cleaning approaches. We are also interested in how we can incorporate uncertain or probabilistic cleaning processes into this framework. For example, given a dirty record, could the data cleaning module specify a set of ranges for each attribute? We are interested in what guarantees, if any, we can achieve in such settings.

Complex SQL Queries: Another important avenue of future work is to extend our framework to support more complex SQL queries such as join and nested SQL queries. There are some straightforward methods to implement these queries. For example, we can materialize the join result as a single table, and then apply our framework to the materialized table. But this could be very costly for large datasets, thus we need to explore more efficient implementations. For a larger set of queries, it may not be possible to estimate their results with the CLT. Thus, exploring empirical estimation approaches (such as bootstrapping) to our framework is another interesting future direction.

Sample Maintenance: Finally, in real applications users may create multiple samples from the data. Maintaining these samples for data updates is also very challenging and needs to be investigated.

Acknowledgements. The authors would like to thank Sameer Agarwal, Bill Zhao, and the SIGMOD reviewers for their insightful feedback. This research is supported in part by NSF CISE Expeditions Award CCF-1139158, LBNL Award 7076018, DARPA XData Award FA8750-12-2-0331, the European Research Council under the FP7, ERC MoDaS, agreement 291071, and by the Israel Ministry of Science, and gifts from Amazon Web Services, Google, SAP, The Thomas and Stacey Siebel Foundation, Apple, Inc., Cisco, Cloudera, EMC, Ericsson, Facebook, GameOnTalis, Guavus, Hortonworks, Huawei, Intel, Microsoft, NetApp, Pivotal, Samsung, Splunk, Virdata, VMware, WANdisco and Yahoo!.

8. REFERENCES

[1] S. Acharya, P. B. Gibbons, and V. Poosala. Congressional samples for approximate answering of group-by queries. In *SIGMOD Conference*, pages 487–498, 2000.

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

[3] 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.

[4] A. Aldroubi. Non-uniform weighted average sampling and reconstruction in shift-invariant and wavelet spaces. *Applied and Computational Harmonic Analysis*, 13(2):151–161, 2002.

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

[6] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer, 2002.

[7] V. Barnett. Sample survey. *Principles and method*, 3, 1991.

[8] K. S. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla. On synopses for distinct-value estimation under multiset operations. In *SIGMOD Conference*, pages 199–210, 2007.

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

[10] M. Charikar, S. Chaudhuri, R. Motwani, and V. R. Narasayya. Towards estimation error guarantees for distinct values. In *PODS*, pages 268–279, 2000.

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

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

[13] P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Trans. Knowl. Data Eng.*, 24(9):1537–1555, 2012.

[14] 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.

[15] J. Considine, F. Li, G. Kollios, and J. W. Byers. Approximate aggregation techniques for sensor databases. In *ICDE*, pages 449–460, 2004.

[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] 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.

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

[19] DataWrangler. `http://vis.stanford.edu/wrangler`.

[20] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, pages 469–478, 2012.

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

[22] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

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

[24] 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.

[25] Full version. `http://goo.gl/SPNP1R`.

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

[27] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *VLDB*, pages 311–322, 1995.

[28] G. J. Hahn and W. Q. Meeker. *Statistical intervals: a guide for practitioners*, volume 328. Wiley. com, 2011.

[29] M. H. Hansen. Some history and reminiscences on survey sampling. *Statistical Science*, 2(2):180–190, 1987.

[30] D. Harnik, O. Margalit, D. Naor, D. Sotnikov, and G. Vernik. Estimation of deduplication ratios in large data sets. In *MSST*, pages 1–11, 2012.

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

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

[33] D. V. Hinkley. Bootstrap methods. *Journal of the Royal Statistical Society. Series B*, pages 321–337, 1988.

[34] J. Jacod and A. N. Shiryaev. *Limit theorems for stochastic processes*, volume 288. Springer-Verlag Berlin, 1987.

[35] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative support for sensor data cleaning. In *Pervasive*, pages 83–100, 2006.

[36] 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.

[37] S. Jukna. Analysis of boolean functions. In *Boolean Function Complexity*, pages 55–77. Springer, 2012.

[38] G. Kalton. *Introduction to survey sampling*, volume 7. SAGE Publications, Incorporated, 1983.

[39] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1):484–493, 2010.

[40] S. Kullback. *Information theory and statistics*. Courier Dover Publications, 1997.

[41] J. Lee. *U-statistics: Theory and Practice*. CRC Press, 1990.

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

[43] S. L. Lohr. *Sampling: design and analysis*. Cengage Learning, 2010.

[44] C. McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

[45] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21:1087, 1953.

[46] OpenRefine. `http://openrefine.org`.

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

[48] H. Park and J. Widom. Crowdfill: Collecting structured data from the crowd. Technical report, Stanford University.

[49] C.-E. Sarndal, B. Swensson, and J. H. Wretman. *Model assisted survey sampling*. Springer, 2003.

[50] 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.

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

[52] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar. Crowdsourced enumeration queries. In *ICDE*, pages 673–684, 2013.

[53] R. Valliant, A. H. Dorfman, and R. M. Royall. *Finite population sampling and inference: a prediction approach*. Wiley New York, 2000.

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

[55] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD Conference*, pages 229–240, 2013.

[56] H. Weisberg. *The Total Survey Error Approach: A Guide to the New Science of Survey Research*. University of Chicago Press, 2009.

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

[58] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *PVLDB*, 4(5):279–289, 2011.