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

ABSTRACT

Aggregate query processing over very large datasets can be slow and prone to error due to dirty (missing, erroneous, duplicated, or corrupted) values. To address the speed issue, there has lately been a resurgence of interest in sampling-based approximate query processing, but this approach further reduces answer quality by introducing sampling error. In this paper we explore an intriguing opportunity that sampling presents, namely, that when integrated with data cleaning, sampling actually improves answer quality. Data cleaning requires either domain-specific software (which can be costly and time-consuming to develop) or human inspection. The latter is increasingly feasible with crowdsourcing but can be highly inefficient for large datasets. Our approach requires cleaning only samples from the dataset to process aggregate numerical queries such as average, sum, count, variance, geometric mean, and product. We derive confidence intervals as a function of sample size and show how our approach reduces error bias. We also apply allocation theory to optimize cleaning cost and answer quality for group-by queries. We provide a detailed evaluation of our approach using datasets and workloads from the TPC-H benchmark, an on-line citation index and a database of indoor sensor network values. Our 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.

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 to deal with the speed issue. It has been well studied in the database community since the 1990s [2,29], and has drawn renewed attention in recent big data research [3].

SIGMOD'14, June 22–27, 2014, Snowbird, Utah, USA.

By processing queries on a sample of data, SAQP can provide fast response time with approximate answers. However, answer quality is not only dependent on sampling error. The data itself may contain errors which could affect query results. According to an industry survey, more than 25% of the critical data in top companies contained significant data errors [48]. 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 [17]. Therefore, even running a query on the entire data may still not get an accurate query result, and SAQP further reduces answer quality by introducing sampling error.

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. 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 [21,33,54]. While crowdsourcing makes this increasingly feasible, it is still highly inefficient for large datasets [50].

To overcome this limitation, 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. To better manage cleaning effort and answer quality, we further explore how SampleClean can integrate with cleaning cost or result quality constraints. We formalize these as the cost-constraint problem and quality-constraint problem respectively, and give the optimal solution to these problems.

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 sam-

ple 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 formalize the cost-constraint and quality-constraint problems, and present the optimal solutions to them.
- 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 introduces sampling error and data error that commonly affect query results. Section 3 presents our new SampleClean framework. We describe RawSC and NormalizedSC in Section 4 and Section 5, respectively. Section 6 discusses the solutions to cost-constraint and quality-constraint problems. We describe the experimental results in Section 7. Section 8 covers related work and Section 9 makes the conclusion.

2. SAMPLING ERROR AND DATA ERROR

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
```

In this section, we will precisely characterize sampling and data errors. Throughout the section, we will refer to the following example query on a dataset of academic publications:

```
SELECT AVG(citation_count) FROM papers
WHERE pub_year>2000
```

which finds the average number of citations received by the publications published after the year 2000.

2.1 Sampling Error

There are many different ways to sample data; a data sample could be either created online during the query time [12,29,45,52] or built offline from past query workloads [2,3,5,10]. 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 [1,3,29], 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 [40]. 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.

(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).

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 attribute of the query (i.e. `pub_year`), there may be some tuples that *falsely* satisfy or dissatisfy the predicate, leading to an incorrect result. In Figure 1(a), the first paper t_1 also has condition error since it was published after the year 2000 and should satisfy the predicate. Note that this type of error can also be extended to errors in `group-by` attributes (refer to Section 6 for details).

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}.

Value error and condition error are both caused by the incorrect attribute values of the dirty data. Many data-cleaning techniques, such as outlier detection [28,32] or rule-based approaches [15,20], have been proposed to solve this problem. For example, Fan et al. [21] proposed editing rules, master data and user confirmation to correct attribute values, and they proved that their approaches can always obtain reliable cleaning results.

There are also many studies in dealing with the duplication error (see [19] for a survey). Particularly, recent work explores the use of crowdsourcing to solve this problem. For example, Wang et al. [50] proposed a hybrid human-machine framework, which first adopts machine-based techniques to filter obviously non-duplicate tuples for each tuple, and then utilizes the crowd to check the remaining ones.

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.

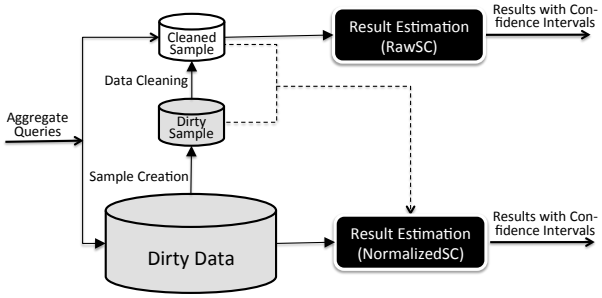

Figure 2: SampleClean framework.

3. SAMPLECLEAN FRAMEWORK

In this section, we present SampleClean, a novel framework that cleans only a sample of data and then processes aggregate queries based on the cleaned sample. 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. For each tuple in the sample, the data-cleaning technique corrects the attribute values of the tuple, and finds the number of duplicates for the tuple from the dirty data. For example, consider a dirty sample, $S = \{t_1, t_2, \cdots, t_7\}$ from the table 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. Since our framework is independent of how samples are cleaned, in this paper we take data cleaning as a black box, and assume that it always gives us correct cleaning results.

Next, SampleClean uses the cleaned sample to answer 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 4.

An alternative estimation approach is to estimate by how much the dirty data differs from the cleaned data. We can estimate the difference based on comparing the dirty and cleaned sample, and then subtract this estimate from a query result on the dirty data. We describe this alternative approach, called NormalizedSC, and compare its performance with RawSC in Section 5.

To facilitate a flexible trade-off between data-cleaning cost and answer quality, SampleClean also allows users to specify a result-quality or cleaning-cost constraint for their aggregate queries. 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? We discuss how to address these problems in Section 6.

4. RawSC APPROACH

In this section, we present the RawSC estimation approach. RawSC takes a clean sample of data as input, runs an aggregate query directly on the clean sample, and derives a confidence interval around this result. If we assume a perfect cleaning process, this technique gives an unbiased estimate of the true result.

4.1 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 (see [22] 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.
- $K_{\texttt{pred}}$: 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`: $\phi(t) = \texttt{Predicate(t)} \cdot N$
- `sum`: $\phi(t) = \texttt{Predicate(t)} \cdot N \cdot t[a]$
- `avg`: $\phi(t) = \texttt{Predicate(t)} \cdot \frac{K}{K_{\texttt{pred}}} \cdot t[a]$

[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.

4.2 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:

- `Correct`(t): for the tuple t return a tuple with correct attribute values
- `Numdup`(t): for the tuple t return the number of duplicates that exist in the population P

4.2.1 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:

$$\phi_{\texttt{clean}}(t) = \phi\left(\texttt{Correct}(t)\right). \tag{4}$$

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.

4.2.2 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 $\frac{3}{2}$.

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 cancel out 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 cancel out 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 [22] 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:

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

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 $\phi_{clean}(S)$.

4.2.3 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`: $\phi_{\texttt{clean}}(t) = d \cdot \frac{\phi(\texttt{Correct}(t))}{\texttt{Numdup}(t)}$

We plug $\phi(\cdot)$ (as described in Section 4.1) 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:

Table 1: $\phi_{\texttt{clean}}(\cdot)$ for `count`, `sum`, and `avg`

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

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:*

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

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. Being a `mean` of a uniform random sample of $P_{\texttt{clean}}$, this is an unbiased estimate. See [22] 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.*

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 $r = \frac{K}{K_{\text{pred}}}$, 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.

		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.

5. NormalizedSC APPROACH

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.

5.1 Estimating Bias

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

$$f(P) = \frac{1}{N}\sum_{t\in P}\phi(t) \qquad f(P_{\texttt{clean}}) = \frac{1}{N}\sum_{t\in P}\phi_{\texttt{clean}}(t) \tag{5}$$

If we solve for ϵ, we find that:

$$\epsilon = \frac{1}{N}\sum_{t\in P}\Big(\phi(t) - \phi_{\texttt{clean}}(t)\Big) \tag{6}$$

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 [22] for a detailed proof. □

Consider the Example 3 discussed in the previous section.

EXAMPLE 4. *Based on the definition of $\phi(\cdot)$ in Section 4.1, 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 $\phi_{\texttt{clean}}(\cdot)$ (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 $(f(P) - \mu_q) \pm \lambda\sqrt{\frac{\sigma_q^2}{K}}$.

5.2 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 $\pm\lambda\sqrt{\frac{\sigma_q^2}{K}}$ and for RawSC it is $\pm\lambda\sqrt{\frac{\sigma_c^2}{K}}$. 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 [22] for details.

6. CONSTRAINED GROUP-BY QUERIES

In this section, we discuss how to enforce quality and cost constraints on the queries by setting the size of our sample. In this work, we define the cleaning cost as the number of cleaned tuples. Our model could support more complex costs to model processor time in algorithmic cleaning or human costs for crowdsourced cleaning. We defer handling general costs for future work.

If the query does not have a `group-by` clause, handling constraints is trivial. For a query with a cleaning-cost constraint, we should draw and clean a uniform sample of the maximum size. For a query with a result-quality constraint, we use the results in Section 4 and 5 to solve for K that meets our desired quality.

When there are multiple populations, as in `group-by` queries, we need to allocate samples to each of the groups. In Sections 6.1 and 6.2, we discuss some of the theory behind satisfying `group-by` queries. Using this theory, we present the optimal solutions to the quality-constraint and cost-constraint problems.

6.1 Confidence Intervals For Group-By Queries

To address `group-by` queries (with multiple populations $\{P_1, P_2, \cdots, P_M\}$), we can simply sample from each group independently and then apply our algorithms. In order to better balance the estimate results of different populations, we set the maximally sized confidence interval to quantify the uncertainty in the group-by query: $\max_i e_i$, where e_i denotes the size of confidence interval for the i-th population.

When there are errors in the `group-by` attributes, sampling independently from each of the multiple populations is not possible. In this case, we can sample uniformly from a union of all the populations, and then treat the `group-by` clause as a predicate. We can then return the maximally sized confidence interval as the confidence interval of our

estimate, and apply the constraint handling scheme for a single population described above.

However, many practical queries do not have errors in the `group-by` attributes, and we can sample directly from each population. For example, in one of our experimental datasets, we found that while sensor readings were often incorrect, their reported timestamp was accurate. In these datasets, we can minimize the additional effort by solving an optimal allocation problem.

6.2 Handling Constraints

Recall that our framework handles two types of constraints: quality and cleaning-cost. A quality constraint problem is defined as: given an error tolerance, return a result with a confidence interval of at most that size with the least possible cost. Similarly, the cost constraint problem is defined as: given a cleaned sample size, return a result with the highest quality.

6.2.1 Sample Allocation Theory

Before we discuss the constraints, we discuss optimal allocations of samples in `group-by` queries. Since we define the total confidence interval as the maximally sized confidence interval, and in Sections 4 and 5, we return intervals of the form $\pm\lambda\frac{\sigma}{\sqrt{k}}$, then the optimal allocation problem is formally:

DEFINITION 1. *Given a set of M populations $\{P_1, P_2, ..., P_M\}$, population variances $\{\sigma_1^2, \sigma_2^2, ..., \sigma_M^2\}$, and a budget B, the optimal allocation problem is:*

$$\min_K \max\left(\frac{\sigma_1^2}{k_1}, \frac{\sigma_2^2}{k_2}, ..., \frac{\sigma_M^2}{k_M}\right)$$

$$\textit{subject to: } \sum_i^M k_i \leq B$$

$$\forall i : k_i \leq |P_i|$$

This problem is a convex Geometric Program, and can be readily solved with many different techniques/solvers. Problems of this form have been well studied in stratified sampling theory [35], e.g. Neyman allocation.

6.2.2 Cost Constraint Problem

The solution to Definition 1, gives a clear way to solve a cost constraint problem. Suppose, we are given a budget of B samples to clean. We first clean a small constant number of samples c for each population. Then, we apply the result estimation approach, which will return a confidence interval of the form $\pm\lambda\frac{\sigma_i}{\sqrt{c}}$. With the σ_i from the confidence intervals, we solve the optimization problem for the remaining $B - cM$ samples allocating them optimally.

6.2.3 Quality Constraint Problem

The solution to the quality constraint problem follows directly from the fact that to satisfy $max(...) \leq q$, every argument must be no larger than q, where q is the quality constraint. Therefore, like before we first clean a small constant number of samples c for each population to estimate the variance. Then, for each population i, we solve for k_i from $\lambda\frac{\sigma_i}{\sqrt{k_i + c}} \leq q$.

7. 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.

7.1 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.

7.1.1 TPC-H Dataset

We generated a 1GB TPC-H benchmark[2] 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.

7.1.2 Microsoft Academic Search Dataset

Microsoft maintains a public database of academic publications[3]. 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. We cleaned the data as best as possible by applying our own domain knowledge to remove the duplicates and mis-attributions. The characteristics of this dataset are shown in Table 2.

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

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

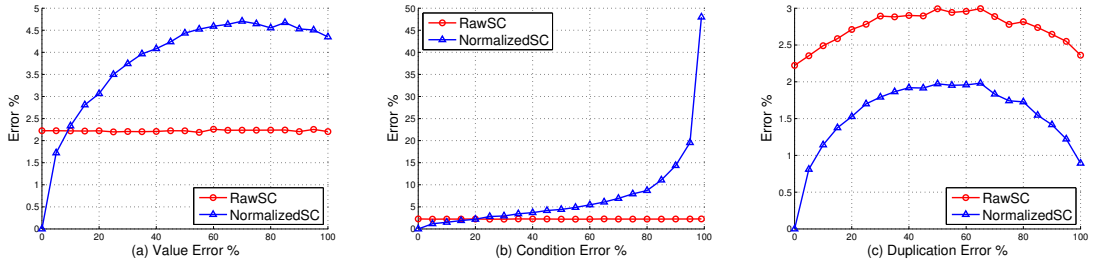

Figure 4: Varying the amount of each type of error on the TPC-H dataset and measured the performance of NormalizedSC vs. RawSC.

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	95	71.79%	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 402 respectively). Jeffery Ullman had a comparatively larger number of duplicated papers (182).

7.1.3 *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[4]. 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 [32].

7.2 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 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.

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

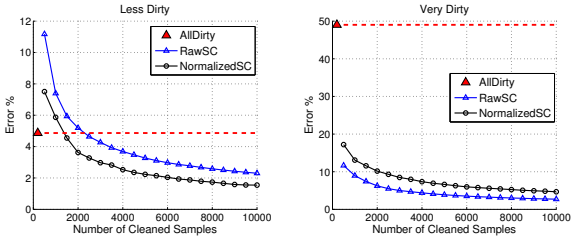

Figure 5: Comparison of the convergence of the methods on two TPC-H datasets with simulated errors (30% value, 10% condition, 20 % duplication) and (3% value, 1% condition, 2% duplication). On the dataset with larger errors, we find that RawSC gives a narrower confidence interval, and on the other NormalizedSC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

7.3 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 $\frac{1}{\sqrt{K}}$ with respect to 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.

We also compare our approaches with AllDirty. We can see both RawSC and NormalizedSC can achieve a better result by cleaning only a small number of samples. Section 7.5 contains a more detailed comparison with AllDirty and AllClean.

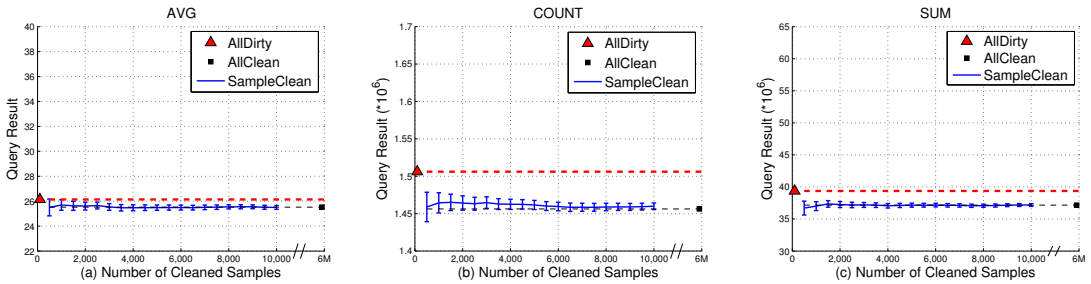

Figure 6: TPC-H evaluation with 3% Value Error, 1% Condition Error, 2% Duplication Error. Our technique can efficiently estimate on datasets with a small number of errors due to the trade-off between RawSC and NormalizedSC.

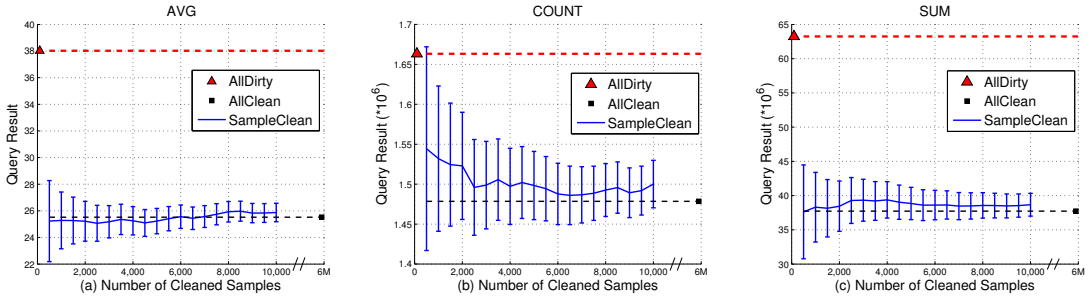

Figure 7: TPC-H evaluation with 30% Value Error, 10% Condition Errors, 20% Duplication Error. We find that even for a small number of cleaned samples, we can return results with high confidence. We are able to achieve 5% accuracy after cleaning only 0.08% of the total samples. High rates of certain types of random errors, such a duplication, may start to cancel out bias. For example, if all of the tuples are duplicated exactly once then there is no need to clean duplicates.

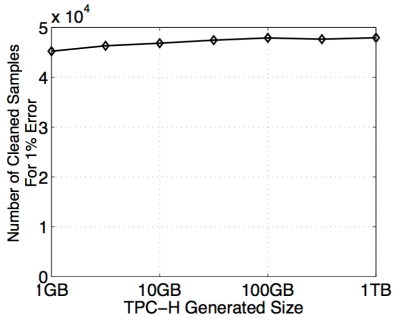

Figure 8: Estimating applied to TPC-H (30%,10%,20%) dataset of different sizes. The number of cleaned tuples needed to achieve a certain accuracy does not increase with the size of the dataset.

7.4 Scalability of Cleaning Cost

In our RawSC and NormalizedSC approaches, 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 8, 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.

7.5 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 7.1.1), we uniformly sampled from the dataset. Our results in Figure 6 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 7). On this dataset AllDirty differs from AllClean by 52% in the `avg` function. The results in Figure 7 show that RawSC 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 estimation method and RawSC was more accurate on one in the presence of larger errors.

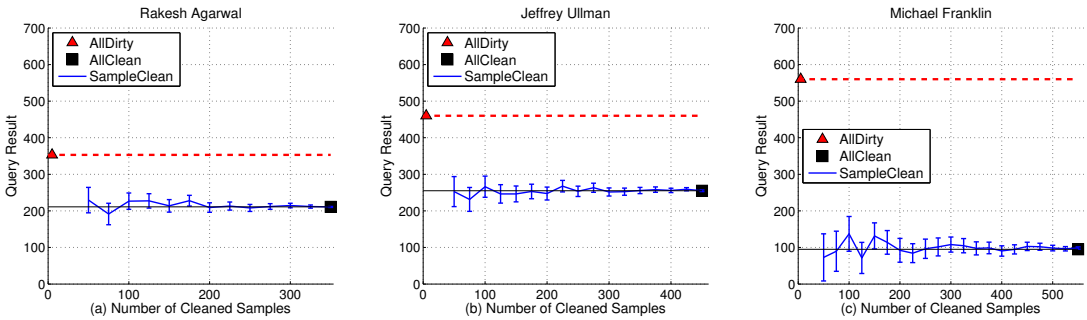

Figure 9: The output of our result estimation framework for each author. The dataset was particularly dirty and cleaning only 50 tuples per author was sufficient to outperform an aggregation of the entire dataset.

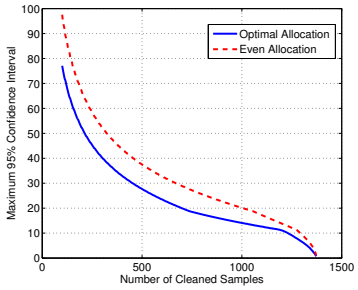

Figure 10: Evaluating the optimal cleaning-cost allocation for a `group-by` query. We can estimate within ± 30 for less than 500 total cleaned samples. The even allocation results in larger errors especially when the cleaning budget is small.

7.6 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 7.1, 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.

7.6.1 *Microsoft Academic Search*

We evaluated a cost constraint `group-by` count query on this dataset. We can see that in Table 2 AllDirty returns an incorrect ranking of the paper counts of the three authors. We designed this experiment to be illustrative of assessing the confidence of decisions based on dirty data.

In Figure 9, we show how our approach performs for each author. 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 estimates of Rakesh Agarwal's and Jeffery Ullman's papers overlap for a small number of cleaned samples.

We quantify this uncertainty by fixing the number of total cleaned samples across all of the authors. We can allocate cleaning cost to each of the authors based on the technique in Section 6.2; solving the optimization program with CVX[5]. The results in Figure 10 shows how the error of the most uncertain estimate varies with the total number of cleaned samples (under optimal allocation). Figure 10 also compares our optimal allocation to an even allocation, and we found that to achieve a confidence interval of ± 30 publications the optimal allocation required 484 cleaned tuples while the even allocation required 697. Using these optimal allocations, we simulated a 1000 trials for each quality constraint and counted the number times we returned an incorrect ordering. For example, we can return the correct ranking with 95% probability after cleaning only 166 total samples. To achieve a correct ranking with 99% probability, we require 276 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]http://cvxr.com/cvx/

7.6.2 *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 [32] and applied a variety of queries to illustrate how different types of value and condition errors can affect results (Figure 11).

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 $\pm 1.5\%$ and $\pm 15.2\%$ 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.

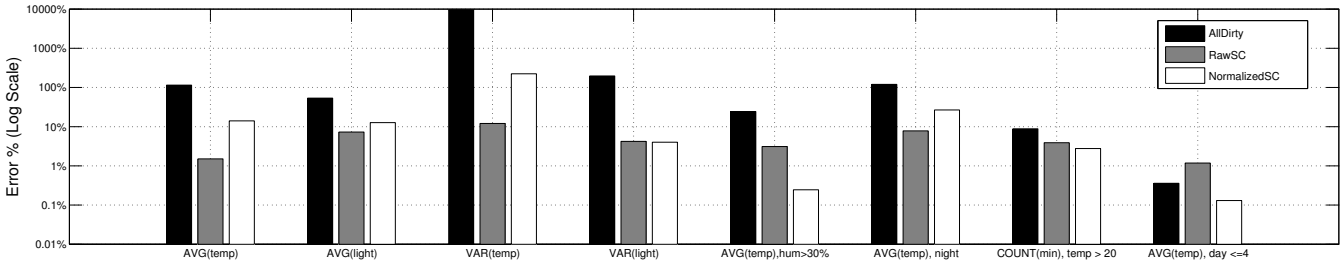

Figure 11: Queries applied to the sensor dataset. We applied a set of queries with a fixed sample (same for all queries) of 500 clean records. We compare relative error of AllDirty, RawSC, and NormalizedSC with respect to AllClean. The error percentage is shown on a log-scale. For all of these example queries, we are able to achieve a relative error of less than $\pm 10\%$ even when the data error is orders of magnitude higher.

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.

8. RELATED WORK

Approximate Query Processing: SAQP has been studied for more than two decades [14,23]. Many SAQP approaches [2,3,5,10,12,29,45,47,52] 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 [14]. 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 [15, 20], outlier detection [16,28], and duplicate detection [19]. In order to ensure reliable cleaning results, most of these techniques require human involvement [21,33,50,54]. 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 [7,26,35,46,49,51]. 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 [30]. For a detailed survey of different types of statistical bounds refer to [25]. 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 [38] for a survey of this theory. The stochastic process literature also discusses problems in unbiased estimation from samples [31].

Distinct Count Estimation: Distinct count estimation has been an open problem in stream processing and database research [6,8,13,24]. Similar to our analysis, some have argued that simply removing duplicates in a sample of data does not work [9]. In the storage literature, similar weighted average techniques have been applied for global file duplication rate estimation [27] 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 [39,43]. Further relevant work includes sensitivity analysis of set functions [34,41] and statistical information theory [37].

9. 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. Moreover, SampleClean can optimally satisfy user-specified cleaning-cost or

result-quality constraints. 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. Firstly, our framework can return unbiased query results by assuming a perfect cleaning process on the sample. While we can rely on domain experts to carefully clean the sample data, it is still hard to guarantee that no error exists in the cleaned sample. Thus, it is desirable to study how to estimate the accuracy of data-cleaning techniques and return unbiased results even with an imperfect cleaning process. 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. 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.

10. 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. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla. On synopses for distinct-value estimation under multiset operations. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 199–210. ACM, 2007.

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

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

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

[12] 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.

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

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

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

[21] 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.

[22] Full version. `http://to-be-available?`

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

[24] 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.

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

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

[27] 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.

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

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

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

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

[32] 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.

[33] 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.

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

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

[36] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. A scalable bootstrap for massive data. *arXiv preprint arXiv:1112.5016*, 2011.

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

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

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

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

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

[42] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. Dremel: Interactive analysis of web-scale datasets. *PVLDB*, 3(1):330–339, 2010.

[43] 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.

[44] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, pages 841–848, 2001.

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

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

[47] 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.

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

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

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

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

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

[53] R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica. Shark: SQL and rich analytics at scale. In *SIGMOD Conference*, pages 13–24, 2013.

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