PrivateClean: Data Cleaning and Differential Privacy

Sanjay Krishnan, Jiannan Wang*, Michael J. Franklin, Ken Goldberg, Tim Kraska†
UC Berkeley, * Simon Fraser University, †Brown University
{sanjaykrishnan franklin, goldberg}@berkeley.edu †tim_kraska@brown.edu *jnwang@sfu.ca

ABSTRACT

Recent advances in differential privacy make it possible to guarantee user privacy while preserving the main characteristics of the data. However, most differential privacy mechanisms assume that the underlying dataset is clean. This paper explores the link between data cleaning and differential privacy in a framework we call PrivateClean. PrivateClean includes a technique for creating private datasets of numerical and discrete-valued attributes, a formalism for privacy-preserving data cleaning, and techniques for answering `sum`, `count`, and `avg` queries after cleaning. We show: (1) how the degree of privacy affects subsequent aggregate query accuracy, (2) how privacy potentially amplifies certain types of errors in a dataset, and (3) how this analysis can be used to tune the degree of privacy. The key insight is to maintain a bipartite graph relating dirty values to clean values and use this graph to estimate biases due to the interaction between cleaning and privacy. We validate these results on four datasets with a variety of well-studied cleaning techniques including using functional dependencies, outlier filtering, and resolving inconsistent attributes.

1. INTRODUCTION

It is often beneficial for organizations to share user data with third parties or publish publicly available datasets. For example, the anonymous rating data shared as a part of the Netflix challenge engaged the community in developing several new recommendation algorithms [2]. While the success stories are numerous and ever growing in number, preserving the privacy of users is a fundamental obstacle [1]. Over the last few years, there have been several examples where ostensibly private data are linked to individuals through clever data mining techniques [9,32,48]. Thus, it is important to have privacy mechanisms with formal guarantees about resilience to attack without sacrificing too much utility.

Fortunately, recent advances in differential privacy make it possible to guarantee privacy against a wide range of linkage attacks while preserving the main characteristics of the data [11]. However, for reliable downstream query processing, existing models assume that the underlying data are "clean". However, raw data often requires extensive pre-processing including the extraction of named entities from textual fields [8], imputation of missing values [36], and resolution of inconsistent attributes [39]. Consequently, the burden is on the data provider to ensure that the data are completely clean before making it private, and this cost might even discourage her from sharing the data in the first place. In cleaning, the data provider has to account for every possible downstream analysis scenario, each of which may have different data quality requirements and notions of "clean".

This paper explores a differential privacy model that is compatible with several common cleaning operations and result estimation for aggregate queries after cleaning. This crucially removes much of the cleaning burden from the data provider, and analysts can clean the data as required. We propose a framework called PrivateClean, which includes a technique for creating private datasets of numerical and discrete-valued attributes, and answering `sum`, `count`, and `avg` queries after cleaning the private relation. PrivateClean supports cleaning operations that can be represented as deterministic user-defined functions over the discrete attributes that: (1) transform existing attributes, (2) create new attributes, and (3) resolve disparate attribute values to a single canonical representation. We further provide a lower bound on the amount of data required to detect the original data errors as a function of the degree of privacy and how this degree of privacy affects subsequent aggregate query accuracy.

To illustrate how this works, consider collecting college course evaluations consisting of a student's satisfaction on a scale from *1-5*, major, and some other demographic data (Figure 1a). Even

(a) Original Table

id	Major	Satisfaction
1	**Mechanical E.**	4
2	**Mech. Eng.**	3
3	Electrical Eng.	1
…	…	…
100	Electrical Eng.	5

(b) Private Table [Randomize Majors]

id	Major	Satisfaction
1	Mechanical E.	4
2	Mech. Eng.	3
3	Mech. Eng.	1
…	…	…
100	Electrical Eng.	5

(c) Fix Inconsistencies

id	Major	Satisfaction
1	Mech. Eng.	4
2	Mech. Eng.	3
3	Mech. Eng.	1
…	…	…
100	Electrical Eng.	5

(d) Query Result Estimation

AVG	Major
3.2	Mech. Eng.
4.9	Electrical Eng.
…	…

Figure 1: Randomly permuting a random fraction of major values can preserve the privacy of students in rare majors (Student 3 has plausible deniability). The resulting randomized table is still compatible with subsequent data cleaning (i.e., resolving inconsistencies).

SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-3531-7/16/06...$15.00
DOI: http://dx.doi.org/10.1145/2882903.2915248

with guarantees of anonymity, students may be reluctant to provide truthful responses as it might be possible to identify an individual based on a combination of the demographic data and the major. PrivateClean extends the well-studied *Randomized Response* [45] technique where each student is given plausible deniability: (1) she flips a coin before the evaluation, (2) if heads, she reports her major/demographics truthfully, and (3) if tails, she selects an answer at random. Clearly, the likelihood of tails (p) vs. heads ($1 - p$) controls the risk of identification (Figure 1b). On the other hand, increasing p (the degree of privacy) also decreases the accuracy of the analysis (i.e., the estimated average satisfaction for Mechanical Engineers).

This principle can be used to randomize any table by replacing discrete-valued attributes with probability p with another value in the domain uniformly at random; which we formalize as *Generalized Randomized Response* (GRR). The interesting technical challenges happen in query processing after randomization. By design, increasing the randomization parameter p makes rare values more common which affects the selectivity of queries and user-defined cleaning. We derive an estimate for the change in selectivity and compensate for the bias that this induces in query results. To do so, we have to maintain provenance of the values in the relation before and after the user-defined transformations using a bipartite graph (mapping dirty to clean values).

To summarize the contributions of PrivateClean:

- PrivateClean provides a mechanism for generating differentially private datasets on which the analyst can apply cleaning operations. PrivateClean maintains the necessary information to estimate query results on the cleaned private dataset. (Section 4).
- We show that directly applying `sum`, `count`, and `avg` queries with predicates produces a biased estimate where the bias is proportional to the skew in the dataset $\tilde{O}(privacy \cdot skew)$ (i.e., the selectivity in terms of distinct values vs. the selectivity in terms of actual records). We derive a corrected estimator that compensates for the bias.
- This bias is exacerbated by data cleaning operations that merge attribute values $\tilde{O}(privacy \cdot (skew + merge))$. Consequently, we leverage a graph-based provenance model to estimate and additionally compensate for the data cleaning.
- We evaluate the analysis on two synthetic and two real datasets.

2. PRIVATE CLEANING: 3 APPROACHES

This section introduces a running example to motivate private data cleaning, and discusses different approaches to clean the example dataset.

2.1 Motivating Example

M-CAFE is an online application for collecting college course evaluations [49]. Data are collected in the following relation:

$$R(id, major, section, instructor, score)$$

with a primary key id, a string attribute major, a section number, the instructor's name instructor, and a continuous numerical score for the instructor (0-5). For this example dataset, the section attribute uniquely determines the instructor with the functional dependency:

$$R[section] \mapsto R[instructor]$$

Even if the dataset is anonymous (i.e., removing the id attribute), it is not formally private. One might be able to identify individuals with rare major values, and consequently, may penalize those who left a low score. However, since these evaluations are collected via student input, they are susceptible to inconsistencies. For example, the major attribute might have alternative representations of the same logical value "Mechanical Engineering" and "Mech. Eng.". If the instructor wants to know the average score of Mechanical Engineers, she will have to resolve this inconsistency before analysis.

Consider the scenario, where we are providing this online course evaluation platform as a hosted service. We want to release the dataset to the course instructors for analysis, but we also have to provide privacy guarantees for the students. These data are potentially inconsistent, and the following describes three approaches to facilitate reliable analysis on private data. For clarity, we will use the term *provider* for the owner of the data and *analyst* to describe the recipient of the private data.

2.2 Clean First

One solution is to fully clean the dataset before making it private. However, data cleaning is often time consuming [21]. It may not be feasible to clean all of the errors before releasing a private version. Furthermore, it can be impractical or even impossible to anticipate how analysts will clean the data. For example, in our course evaluations example, suppose one student wrote "Mechanical Engineering and Math" as her major. For some types of analysis, it may be acceptable to resolve this inconsistency to "Mechanical Engineering", but there may be other cases where analysts want to process students with double majors differently. Additionally, the query workload provides valuable information about how to budget cleaning effort. For example, to calculate the average score for all students, one does not need to resolve this problem in the first place. Therefore, such transformations are best left to the discretion of the analyst.

2.3 Encryption

Another approach is homomorphic encryption that allows for data processing directly on the encrypted data [38]. The provider can encrypt all of the data and give the encrypted copy to the analyst. Essentially, homomorphic encryption is a domain transformation that obscures the values in a database. While homomorphic encryption preserves the frequencies of the distinct values, it does not preserve their actual values. In the running example, this leads to the natural issue where "Mechanical Engineering" and "Mech. Eng." are transformed to some non-human interpretable value (e.g., 0x86EF001 and 0x211415). This poses a problem for an analyst who wants to merge "Mechanical Engineering" and "Mech. Eng.". For cryptographic transformations, it is provably hard to figure out which of the encrypted values correspond to "Mechanical Engineering" and "Mech. Eng.".

2.4 PrivateClean: Solution Overview

To motivate PrivateClean, let us consider the simpler case where there are only two attributes $R(major, score)$. Suppose, we want to release a provably private relation R for analysis by the course instructors. Let M be the set of all majors, PrivateClean applies the following privacy mechanism: with probability p replace $r[major]$ with a random value from M chosen uniformly. Intuitively, if the relation is large enough (precisely characterized in Theorem 2), it is likely that the rare majors will be assigned to at least a few other students; affording some level of ambiguity. PrivateClean also needs to randomize the score attribute. Consider the case where it is known that all Civil Engineering majors gave a score of 1. Exploiting this correlation might de-privatize the result, and

PrivateClean will have to add statistical noise to all numerical attributes.

Intuitively, as the degree of privacy increases, the private dataset tends towards a uniform distribution (i.e., most ambiguous). This overestimates the prevalence of rare values, and underestimates of the prevalence of common values. Consequently, the effects of data skew get amplified by privacy, and we derive an analytic formula to compensate for this effect based on query selectivity. However, in the presence of data transformations, the original granularity of randomization may not be apparent (i.e., merging values together), and to this end, PrivateClean maintains provenance to estimate query results.

3. PROBLEM SETUP

This section describes the basic formalism for PrivateClean.

3.1 The Database Provider

A trusted provider wants to release a private relation for analysis by an untrusted analyst. Let R be the original relation, i.e., not private, destined for analysis. We consider relations of numerical attributes (real-valued or integer-valued) and discrete valued attributes (any data type). Let $A = \{a_1, ..., a_l\}$ be the set of numerical attributes, and $D = \{d_1, ..., d_m\}$ be the set of discrete-valued attributes. We assume that all user-defined data cleaning operations occur on the discrete attributes of the relation.

3.1.1 Local Differential Privacy

Given the relation R, the provider wants to release a private version of R denoted by V. We will analyze PrivateClean as a form of differential privacy [11]. A stronger variant of differential privacy, local differential privacy, is a measure of randomness in row-by-row randomized transformations. The level of privacy is parameterized by ϵ, where smaller values imply a greater degree of privacy. Intuitively, ϵ is a measure of ambiguity in a record's true value given observation of only the private value. The key consequence of Differential Privacy is that no deterministic function f applied to V can ever reduce the degree of privacy (i.e., increase ϵ).

Suppose, we are given a randomized algorithm $M : R \mapsto V$ which maps rows of R to a relation of the same schema V. We observe a result of applying this algorithm to some (unknown) row in R and call this $r_{private}$. For every element in $r_{private} \in V$, the probability of observing $r_{private}$ given an input row r is denoted by $P[r_{private} \mid r]$. M is said to be ϵ-local differentially private if for all records $r \in R$:

$$\max_{r,r'} \frac{P[r_{private} \mid r]}{P[r_{private} \mid r']} \leq \exp(\epsilon)$$

In other words, given some observed randomized result, ϵ is a bound on the ratio between the most likely input and the least likely input.

3.2 The Analyst

The analyst is given the private relation V with the same schema as R, and the analyst can apply user-defined data cleaning operations to the discrete attributes of V.

3.2.1 Data Cleaning Model

We model data cleaning operations as deterministic user-defined functions over the discrete attributes in each row. The user-defined function takes any projection g_i of D as input and returns a unique output (or output tuple), and we denote the set of all projections as $proj(D)$. Regardless of the analysts actions, privacy is preserved; even if those actions do not conform to the proposed projection/user-defined function cleaning model. However, if the analyst violates this model and attempts other types of data cleaning operations, result estimation is no longer guaranteed with PrivateClean. Formally, for each projection $g_i \in proj(D)$, a *local cleaner* can take three actions:

$\mathbf{Extract}(g_i)$ Creates a new discrete attribute d_{m+1} based the result of applying $C(v[g_i])$ for all $v \in V$. C is an arbitrary deterministic user-defined operation.

$\mathbf{Transform}(g_i)$ Replaces the values in the projection g_i with the result of applying $C(v[g_i])$ for all $v \in V$:

$$v[g_i] \leftarrow C(v[g_i])$$

$\mathbf{Merge}(g_i, Domain(g_i))$ Replaces the values in the projection g_i with some other value in the domain $Domain(g_i)$ with an arbitrary user-defined choice $C(v[g_i], Domain(g_i))$ for all $v \in V$:

$$v[g_i] \leftarrow C(v[g_i], Domain(g_i))$$

The analyst can apply an arbitrary composition of these local cleaners to the data:

$$C(\cdot) = C_1 \circ C_2 \circ ...C_k$$

These cleaners do not need to be known in advance and can be constructed through data exploration of V. The key point is that the analyst can only clean and query the private relation V. The following are examples of cleaners that can be implemented by PrivateClean:

EXAMPLE 1 (FIND-AND-REPLACE). *The analyst wants to count the number of "Electrical Engineering and Computer Sciences" majors. Suppose* major *attribute has alternative representations of the same logical value "Electrical Engineering and Computer Sciences" and "EECS". The find-and-replace operation "Electrical Engineering and Computer Sciences -> EECS" can be represented with a* $\mathbf{Merge}$ *operation.*

EXAMPLE 2 (FUNCTIONAL DEPENDENCY). *The analyst wants to compare scores across different instructors. However, suppose the functional dependency $V[section] \mapsto V[instructor]$ is violated, namely there are two students with the same section number but with different instructor names. The analyst can first apply a standard functional dependency repair algorithm (e.g., [6]), which will generate a series of updates to the records in $v_i \in V$:*

$$v_i : (section, instructor) \mapsto (section', instructor')$$

Each of these updates can be represented as a $\mathbf{Transform}$ *operation, and thus can be supported in PrivateClean.*

3.2.2 Query Processing

Let us call the resulting private relation after cleaning V_{clean}. By nature, differentially private relations preclude point queries since any given record might be randomized, but since they can answer aggregate queries since they preserve the main characteristics of the dataset. We explore a class of queries that can be answered with close-form Central Limit Theorem confidence intervals. We consider `sum`, `count`, and `avg` queries that aggregate over a single numerical attribute predicated on a single discrete attribute:

```
SELECT agg(a) FROM R WHERE cond(d)
```

where a is a numerical attribute, and d is a single discrete attribute. However, this query class is not a fundamental limitation of the technique, and we discuss extensions in Section 10.

We define accuracy with respect to the hypothetical scenario where the same sequence of cleaning operations $C(\cdot)$ was applied directly to R (non-private) resulting in $R_{clean} = C(R)$. For an aggregate query f,

$$error = |est(V_{clean}) - f(R_{clean})|$$

Bounded confidence means, that for any level of privacy, we can give a bound on $error$. Note that this definition does not consider the fact that a private relation may be harder to clean (in terms of human effort in detecting and reasoning about errors) or require more cleaning (in terms additional constraints). We treat the cleaning operations defined on the private relation by the user as ground truth and estimate the result as if those same operations were applied to the original relation.

3.3 Problem Statements

PrivateClean needs to address two issues, constructing provably private relations and query processing.

Private Relation Creation: Given a relation R with corruptions in its discrete attributes, and a privacy parameter ϵ, find a ϵ-local differentially private relation $V = T(R)$ can be cleaned with Merge, Transform, and Extract operations.

Query Processing: Let V be the private relation, $C = C_1 \circ ... \circ C_k$ be an arbitrary composition of the aforementioned data cleaning operations, and $V_{clean} = C(V)$. Let $R_{clean} = C(R)$ be the hypothetical dataset where the same transformations $C(\cdot)$ are applied to R. For an aggregate function f (`sum`, `count`, `avg`) with single attribute predicates, estimate the value of $f(R_{clean})$, using only the private cleaned relation V_{clean}.

4. GENERATING PRIVATE RELATIONS

This section presents a technique to generate private relations that allow for data cleaning.

4.1 Overview

In the generation phase, PrivateClean explores two new issues not considered prior randomized response models: (1) privacy for multiple columns and data types, and (2) analysis that describes the sufficient amount of data required before the results are meaningful. Multiple columns pose a challenge to differential privacy. This is because even one non-private column has the potential to de-randomize all of the other private columns. Even information that is not sensitive may correlate with sensitive information, and thus can be used to reverse privacy transformations. Consequently, we address (1) by proposing different randomization techniques for discrete and numerical attributes and applying this across all attributes. We address (2) using a worst-case analysis in a hypothetical highly skewed dataset, and analytically show the relationship between dataset size, the number of distinct values, and privacy.

4.2 Generalized Randomized Response

The key idea is to apply a randomized response to the partitioned discrete attributes and add random statistical noise to the numerical attributes. We propose the combination of these two "mechanisms" and call the combined technique Generalized Randomized Response (GRR)[1].

4.2.1 Discrete Attributes

For the discrete attributes, we formalize the intuition described in the introduction. Let d_i be a discrete attribute. Let $Domain(d_i)$ be the set of all the distinct values in $R[d_i]$. Then, for each d_i, we apply a randomized response mechanism:

$$r'[d_i] = \begin{cases} r[d_i] & \text{w.p } 1 - p_i \\ \mathbf{U}(Domain(d_i)) & \text{w.p } p_i \end{cases}$$

[1]All proofs are included in the Appendix

where $\mathbf{U}(\cdot)$ selects an element uniformly at random. Ignoring the other attributes, we can show that this transformation is ϵ-differentially private.

LEMMA 1 (RANDOMIZED RESPONSE MECHANISM). *The projection of each discrete attribute $\Pi_{d_i}(V)$[2] is ϵ-local differentially private, with $\epsilon = \ln(\frac{3}{p_i} - 2)$.*

PROOF SKETCH. The worst case is when there are only two values in the domain and all of the other entries in the database are one value except for one, based on the definition of differential privacy we can arrive at:

$$\epsilon = \ln(\frac{3}{p_i} - 2)$$

□

4.2.2 Numerical Attributes

However, in general, the randomized response model is not meaningful for numerical attributes. To randomize the numerical attributes, we apply a different differential privacy mechanism, the Laplace Mechanism [11]. The key idea is to add heavy-tailed, but zero-mean, statistical noise to every attribute. While this obscures outliers, the noise will average out when aggregating over a large enough dataset.

Formally, for each numerical attribute a_i,

$$r'[a_i] = \mathbf{L}(r[a_i], b_i)$$

where $\mathbf{L}(\mu, b)$ is the Laplace distribution:

$$\mathbf{L}(\mu, b) \sim \frac{1}{2b} \exp(\frac{|\, x - \mu \,|}{b})$$

The Laplace distribution is standard in the study of differential privacy. This distribution has several nice algebraic properties such as being analytically integrable, unlike the Gaussian distribution. Furthermore, it is also the maximum entropy distribution given a fixed mean value.

PROPOSITION 1 (LAPLACE MECHANISM). *The Laplace Mechanism is ϵ-local differentially private, with $\epsilon = \frac{\Delta_i}{b_i}$, where Δ_i is defined as the max difference between any two values in $\Pi_{a_i}(V)$.*

4.2.3 GRR Summary

To summarize, the GRR mechanism takes as input a set of numerical attributes A and a set of discrete attributes D. For each discrete attribute d_i there is a user-specified privacy parameter p_i which is its randomization probability, and for each numerical attribute a_i there is a parameter b_i which is the amount of Laplace noise to add. For each numerical attribute, we apply the Laplace mechanism with b_i, and for each discrete attribute partition, we apply randomized response with probability p_i. Using the composition lemma from Dwork et al. [11], we arrive at the following result:

THEOREM 1 (GENERALIZED RANDOMIZED RESPONSE). *For a set of numerical and discrete attributes, Generalized Randomized Response returns an ϵ locally differentially private mechanism with $\epsilon = \sum_{n_i} \epsilon_{n_i} + \sum_{d_i} \epsilon_{d_i}$, where ϵ_a is calculated per attribute.*

Interpretation: The basic intuition about Theorem 1 is that adding attributes (even private ones) decreases the privacy (by increasing ϵ. We can now clearly see why adding a non-randomized attribute $\epsilon_i = \infty$ de-privatizes all of the other attributes.

[2]Π denotes the standard projection relational operator.

Setting ϵ: So far, we have discussed ϵ as a given parameter from the provider. ϵ is a sum of the individual privacy factors for each attribute ϵ_i. These values can be uniformly allocated by dividing ϵ equally over the numerical attributes and the discrete attributes d_i. For numerical attributes, ϵ_i is controlled by b_i, and for discrete attribute partitions, it is controlled by p_i. Section 5 will show how to derive b_i and p_i based on the dataset size and desired query result accuracy. We formalize this into a parameter tuning algorithm listed in Appendix E.

4.3 Privacy Properties of GRR

If V is ϵ-local differentially private, then V_{clean} is ϵ differentially private, since any deterministic data manipulation is guaranteed to preserve privacy. This result is a direct consequence of Proposition 2.1 in Dwork et al. [11], which guarantees that differentially private relations are immune to post-processing.

In the discrete attributes, there is some probability that a rare value will be masked, that is, for all records that have the value the randomization will set it to a different one, and for all other records, the randomization does not choose this value. Intuitively, for a large enough database, this should be a rare occurrence. The database can regenerate the private views until this is true. Thus, we want to know how large the dataset needs to be before with high probability $(1-\alpha)$ all of the domain values are seen after applying GRR (i.e., the expected number of re-generations is $\frac{1}{1-\alpha}$).

THEOREM 2. *For all discrete attributes d_i, let N be the number of distinct tuples in $R[d_i]$. For $p > 0$, the dataset size S has to be larger than:*

$$S > \frac{N}{p} \log \frac{pN}{\alpha}$$

for the domain is preserved probability $1-\alpha$.

PROOF SKETCH. Assume domain value a is present once, and all other values are different. The probability that a is visible in the private relation:

$$\mathbf{P}[a] = 1 - p\frac{N-1}{N}[1-\frac{p}{N}]^{(S-1)}$$

To extend this bound for any given value, we apply a union bound (multiplying the probability by N):

$$\mathbf{P}[all] \geq 1 - p(N-1)[1-\frac{p}{N}]^{(S-1)}$$

The above formula can be found by solving for S and simplifying the logarithms. $\square$

Essentially, the number of distinct values needs to be on the order of $O(\frac{S}{\log S})$ to ensure that the entire domain is preserved with high probability. This result is related to the well known result of the Coupon Collector's problem [12].

EXAMPLE 3. *For the simplified course evaluation example $R(major, score)$, let us suppose p_i is 0.25 and there are $N = 25$ distinct majors in the dirty relation. We can plug these values into Theorem 2. To have $1-\alpha = 95\%$ confidence, we need approximately 391 records, and for $1-\alpha = 99\%$, we need 552 records.*

5. QUERY PROCESSING

This section describes query processing on private relations without considering cleaning. Direct query processing after Generalized Randomized Response has a bias because rare values start appearing increasingly frequently. Thus, we describe how to compensate for this bias and bound the result in confidence intervals.

5.1 Overview

Given a private relation V and its non-private base relation R, we define *error* in a query result as:

$$error = \mid f(R) - f(V) \mid$$

There are two reasons why a query on V may be erroneous: Laplace privacy in the numerical attribute adding error to the aggregate and randomized response privacy in the discrete attribute adding error to the predicate.

Let us first consider an aggregate without a predicate. Since, GRR adds zero-mean statistical noise to each numerical value, and consequently `sum`, `avg`, and `count` queries can simply be estimated as:

$$f(R) \approx f(V)$$

We call this estimator the Direct estimator that is unbiased for queries without predicates. As we will see, this argument does not quite hold when there is a predicate.

5.2 Quantifying the Bias

Now, let us consider the case when there is a predicate. Let the discrete attribute d be a member of a partition g_i which has N distinct values. Every deterministic predicate $cond(d)$ can be thought of as conditioning on a subset of the N distinct values. Suppose the predicate is querying l of the distinct values. The consequence of randomized response is that there are false positives (predicate falsely true) and false negatives (ignored records). If s is the true fraction of records that satisfy the predicate, the fraction of false positive records is:

$$\text{false positive} = (1-s)p\frac{l}{N} \quad (1)$$

and similarly, the fraction of false negative records is:

$$\text{false negative} = sp\frac{N-l}{N} \quad (2)$$

The key insight is that Equation 1 and Equation 2 are not the same, and thus, there is a bias. Accordingly, PrivateClean will estimate this bias from the observed private data and compensate for it.

PROPOSITION 2. *The Direct estimator introduces a bias w.r.t to the hypothetically cleaned non-private dataset of: $\tilde{O}(\frac{p(l-sN)}{N})$*

PROOF SKETCH. Since the Direct estimator sets two values equal, we can calculate the difference:

$$bias = (1-s)p\frac{l}{N} - sp\frac{N-l}{N}$$

This is the fraction of tuples not accounted for, and thus, up-to some scaling this is the bias. $\square$

What is interesting about this bias is that the term $\frac{(l-sN)}{N}$ is related to the skew in the dataset. l is the number of distinct values for which the predicate is true, s is the fraction of records selected by those distinct values. For uniform data distributions $\frac{l}{N} \approx s$. However, when this is not true, this bias is scaled by the privacy factor in the dataset. So essentially, the bias is $O(privacy \cdot skew)$.

5.3 Compensating For the Bias

To derive an estimator that compensates for this bias, there are four quantities of interest.

True Positive Probability: For a tuple t, given that the predicate is true on the original relation, the probability that predicate remains to be true on the private relation: $\tau_p = (1-p) + p\frac{l}{N}$

False Positive Probability: For a tuple t, given that the predicate is false on the original relation, the probability that predicate is set to be true on the private relation: $\tau_n = p\frac{l}{N}$.

True Negative Probability: For a tuple t, given that the predicate is false on the original relation, the probability that predicate remains to be false on the private relation: $\gamma_p = p\frac{N-l}{N}$.

False Negative Probability: For a tuple t, given that the predicate is true on the original relation, the probability that predicate is set to be false on the private relation: $\gamma_p = (1-p) + p\frac{N-l}{N}$.

It turns out that we can derive unbiased estimators parametrized by these deterministic quantities. Given a query, we know the number of distinct values, the selectivity of the predicate l, and the privacy parameter p. Consequently, these values do not change the statistical properties of the estimation.

5.4 COUNT Estimation

Let s be the fraction of records that truly satisfy the predicate. In the private relation, there will be a fraction s_p of records that nominally satisfy the predicate. We can relate s and p with the following function:

$$s_p = s\tau_p + (1-s)\tau_n$$

We can solve for s, resulting in:

$$s = \frac{s_p - \tau_n}{\tau_p - \tau_n}$$

Therefore, we arrive at the following estimator. Let $c_{private}$ (equivalent to $S \cdot s_p$) be the count on the private view. The estimate of the true count $\hat{c}$ is:

$$\hat{c} = \frac{c_{private} - S\tau_n}{(\tau_p - \tau_n)} \tag{3}$$

Confidence Intervals: To asymptotically bound this estimate, we need to recognize that s_p is actually a random variable. Based on the Central Limit Theorem, we can bound s_p in a confidence interval where S is the size of the relation:

$$s_p \pm z_\alpha \sqrt{\frac{(1-s_p)\cdot s_p}{S}}$$

We can substitute this random variable into the above expression for s, and recognizing that τ_n is deterministic and $1-p = \tau_p - \tau_n$, it follows that the estimate is bounded by:

$$\hat{c} \pm z_\alpha \frac{1}{1-p}\sqrt{\frac{(1-s_p)\cdot s_p}{S}}$$

The confidence interval allows a user to judge a range of possible true values for the `count` query.

Parameter Tuning: However, we may be interested in knowing the error over any `count` query given a level of privacy. Interestingly enough, we can calculate a bound for this independent of the dataset properties. Over all possible queries, $\sqrt{(1-s_p)\cdot s_p}$ is bounded by $\frac{1}{4}$. Thus, for all possible count queries, the error is bounded by:

$$error < z_\alpha \frac{1}{1-p}\sqrt{\frac{1}{4S}} \tag{4}$$

This bound can be used to tune the privacy parameter p.

$$p = 1 - Z_\alpha(\sqrt{\frac{1}{4S \cdot error^2}})$$

5.5 SUM Estimation

Estimating `sum` queries will be slightly more difficult. The key challenge is correlations in the dataset. If the numerical attribute is correlated with the discrete attribute, then false positives and false negatives can lead to different mixtures of data. It is also important to note that unlike `count`, the `sum` queries also query the numerical attributes with the Laplace randomization mechanism.

Let c_{true} be the true count, μ_{true} be the average value of the true records that satisfy the predicate, and μ_{false} be the average value of the records that do not satisfy the predicate. Using this notation, we can derive the following expression for the expected private sum h_p:

$$\mathbb{E}(h_p) = c_{true}\mu_{true}\tau_p + (S - c_{true})\mu_{false}\tau_n$$

$c_{true}\mu_{true}$ is the quantity that we want to estimate.

Unlike in the `count` query case, the challenge is that there are two unknown variables $c_{true}\mu_{true}$ and μ_{false}. Thus, the problem is that there is not enough information in this equation to solve for $c_{true}\mu_{true}$. To address this, we also calculate the sum of the complement as well, that is, we invert the predicate and run the query:

$$\mathbb{E}(h_p^c) = \gamma_n c_{true}\mu_{true} + (S - c_{true})\gamma_p \mu_{false}$$

This is defines a linear system of equations which we can solve for the true sum $h_{true} = c_{true}\mu_{true}$. The resulting estimator becomes:

$$\hat{h} = \frac{(1-\tau_n)h_p - \tau_n h_p^c}{(\tau_p - \tau_n)} \tag{5}$$

Confidence Intervals: The next goal is to bound this estimate in confidence intervals. Notice, that the estimate is a weighted average of h_p and h_p^c. Therefore, it suffices to bound $h_p + h_p^c$. Let μ_p be the mean value across all of the private data and σ_p^2 be its variance:

$$\hat{h} \pm z_\alpha 2 \cdot \sqrt{\frac{s_p(1-s_p)\mu_p^2 + \sigma_p^2}{S}}$$

We can turn this into an analytic bound like before over any `sum` query. Using the fact that $\sigma_p^2 = \sigma^2 + 2b^2$, $s_p(1-s_p) \le \frac{1}{4}$:

$$error \le z_\alpha \frac{1}{1-p}\sqrt{\frac{\mu}{S} + 4\frac{\sigma^2 + 2b^2}{S}} \tag{6}$$

Like before, this can be used to tune the privacy parameter p and b given a desired error.

5.6 AVG Estimation

It turns out that due to a ratio of random variables problem, the intuitive estimator for `avg` $\frac{\hat{h}}{\hat{c}}$ is biased. However, empirically, this bias is small and in fact such estimates are called conditionally unbiased [17]. Confidence intervals can be computed directly from the expressions above by taking the upper confidence interval of $\widehat{h}$ and dividing it by the lower confidence interval of $\widehat{c}$, and vice versa. To get an approximate analytic form for this interval, we can apply standard error propagation techniques [34]:

$$avg \approx \frac{\widehat{h}}{\widehat{c}} \qquad error \approx \frac{1}{\widehat{c}} \cdot \frac{error_{sum}}{error_{count}} \tag{7}$$

where err denotes the width of the confidence interval. The basic intuition is this differentiates the expression and linearizes $\frac{sum}{count}$, and it holds asymptotically.

EXAMPLE 4. *In the running example dataset, suppose p is 0.25, there are 25 distinct majors in the dirty relation, and 500 records. We are interested in counting the number of engineering majors (account for 10 out of the initial 25). Suppose, the private count was* 300. *Using the formula described in Section 5.4* $est = \frac{300 - 500 \times 10 \times .25 \times \frac{1}{25}}{.75} = 333.3$.

6. SINGLE-ATTRIBUTE CLEANING

The previous section described how to process queries on private relations, and this section will describe how to account for data cleaning in the query processing. The basic intuition is that the bias estimation in the previous section is no longer possible after data cleaning since the original distinct values are changed. To do so, we will have to keep track of provenance, i.e., the mapping between original values before data cleaning and the values after cleaning.

Recall, that the cleaning operations are deterministic user-defined operations over projections of the relation $g_i \in proj(D)$. There are two cases, single-attribute and multi-attribute, where the former is the problem where *all* cleaning operations happen on a single attribute $g_i = \{d_i\}$. The next section (Section 7) considers the multi-attribute case.

6.1 Overview

When the original relation was randomized, it was randomized on the dirty data. The key challenge in query processing after data cleaning is that data cleaning may change the number of distinct values and the selectivity of the predicate. The solution is to define a graph that keeps track of the provenance of each value in the domain and use this to find a predicate's selectivity l on the dirty data, and the number of distinct values in the dirty data N.

As in Section 5.2, where privacy amplifies the effects of skew, this discrepancy is also amplified. In other words, after cleaning l and N may change. Starting with the bias described in the previous section:

$$bias = \frac{p(l - sN)}{N}$$

However, since l and N change due to data cleaning, there is a second term accounting for the bias with respect to the new l' and N':

$$bias \leq \frac{p(l - sN)}{N} + p(\frac{l}{N} - \frac{l'}{N'})$$

The term $\frac{l}{N} - \frac{l'}{N'}$ measures the difference between the selectivity (w.r.t to the distinct values) of the query on the cleaned data vs. the original data. We call this term the "merge rate", as it describes how many of the distinct values in the predicate were merged due to data cleaning. Therefore, the resulting bias is:

$$bias = \tilde{O}(privacy \cdot (skew + merge))$$

6.2 Value Provenance Graph

We store a graph mapping dirty values to cleaned values for each attribute and use this graph to compensate for this bias. Let d_{clean} be the discrete attribute in the predicate on the cleaned private relation V_{clean}. Based on the model in Section 3.2.1, where the allowed operations are extract, merge, and transform, each d_{clean} is associated with exactly one attribute d_i. We need to understand the relationship between $Domain(V_{clean}[d_{clean}])$ and $Domain(V[d_i])$. Recall, that Section 3.2.1 makes an assumption of determinism, namely, given two records r and r', where $r[d_i] = r'[d_i]$, the operation has the same result.

Thus, the cleaning model defines a directed bipartite graph between the two domains:

$$Domain(V[g_i]) \mapsto Domain(V_{clean}[d_{clean}])$$

Let $L = Domain(V[d_i])$ be the set of distinct values of the private relation before data cleaning, and let $M = Domain(V_{clean}[d_{clean}])$ be the set of distinct values after data cleaning. Each $l \in L$ and each $m \in M$ defines a node in the bipartite graph. We add edges between L and M to represent the transformation made by a cleaner. Determinism implies a key property that this graph has many-to-one relationships but no one-to-many mappings; also implying that $|L| \geq |M|$. From a graph perspective, this graph is *fork-free* with un-weighted edges. This will be the key differentiating factor between the single-attribute case and the multiple-attribute in the next section.

6.3 Predicates as Vertex Cuts

Maintaining this graph gives us a convenient abstraction to count the number of edges affected by a predicate, and thereby, obtaining the original selectivity. Let $cond(d_{clean})$ be a predicate and each predicate can be expressed as a subset of the clean distinct values $M_{pred} \subseteq M$. Each M_{pred} will have a parent set L_{pred}, that is the set of vertices in L with an edge to exactly one $m \in M_{pred}$. Together $cut = (L_{pred}, M_{pred})$ defines a cut on the bipartite graph.

EXAMPLE 5. *Suppose we have a dirty* major *of four values "Civil Engineering", "Mechanical Engineering", "M.E", "Math". The cleaner maps the first three values to "Engineering", and the user's predicate queries "Engineering". L_{pred} is the set that contains "Civil Engineering", "Mechanical Engineering", "M.E". M_{pred} is singleton set with "Engineering".*

Let $l = |L_{pred}|$, $N = |L|$ and $r[d_i] \in L_{pred}$. We can then recompute the quantities from the previous section w.r.t to the dirty data. The estimate is parametrized by two values τ_p, τ_n, and we can calculate them as follows: The revised true positive probability and false positive probability are:

$$\tau_p = (1 - p) + p\frac{l}{N} \quad \tau_n = p\frac{l}{N}$$

The values can be plugged into the estimator described in the previous section. From a statistical estimation perspective p and l are deterministic values known to the query processing system. Therefore, τ_p, τ_n are deterministic values.

6.4 Efficiency: Linear Space, Linear Time

Of course, there is an additional space- and time-complexity overhead for materializing this graph and calculating the correction. However, we do not have to materialize the entire graph. Let $\hat{N}$ be the number of distinct values in the dirty relation affected by merge or transformation operations. Since the graph is fork-free the resulting cleaned relation will have strictly less than $\hat{N}$ values that link to those in the dirty relation. We only have to store the graph for this subset of vertices in M and L. Implemented with a hash map (cleaned value $\mapsto$ dirty value) , the space complexity of this graph is linear in $\hat{N}$, and the query complexity is linear in the distinct-value selectivity $l' = |M_{pred}|$.

PROPOSITION 3. *For an attribute that is individually cleaned with deterministic user-defined merge and transform operations. Let $\hat{N}$ be the total number of affected distinct values by these operations. The resulting provenance graph can be stored with $O(\hat{N})$ space and for a predicate with distinct-value selectivity l' can be queried in $O(l')$ time.*

7. MULTIPLE-ATTRIBUTE CLEANING

This section extends the graph cut formulation to the case when the cleaning is over multiple attributes. Since we only consider queries whose predicates are defined over single discrete attributes, it is sufficient to maintain a a single graph per-attribute. However, to understand why this case is more difficult consider the following example:

EXAMPLE 6. *Suppose g_i is (*section, instructor*). The private relation has rows "1, John Doe", "1, NULL", "2, NULL". The cleaner maps "1, NULL" to "1, John Doe" and "2, NULL" to "2, Jane Smith". The dirty distinct value $NULL$ has two possible cleaned values* `John Doe` *and* `Jane Smith`.

As illustrated in the example, the basic change from the previous section is that the provenance graph has forks, and as a result, the definition of l has to change.

7.1 Weighted Provenance Graph

As before, we need a graph to represent the relationship between $Domain(V_{clean}[d_{clean}])$ and $Domain(V[d_i])$, and we store one graph per discrete attribute. The cleaning model defines a directed bipartite graph between the two domains:

$$Domain(V[g_i]) \mapsto Domain(V_{clean}[d_{clean}])$$

Let $L = Domain(V[d_i])$ be the set of distinct values of the private relation before data cleaning, and let $M = Domain(V_{clean}[d_{clean}])$ be the set of distinct values after data cleaning. Each $l \in L$ and each $m \in M$ defines a node in the bipartite graph. We add edges between L and M to represent the transformation made by a cleaner. Let $V^{(l)}[d_i]$ denote the set of rows that have the distinct value l and $V^{(m)}_{clean}[d_i]$ denote the set of clean rows that have the distinct value m. Unlike before, we add a weight w_{lm} to each of the edges to represent the fraction of rows with the value l mapped to the value m:

$$w_{lm} = \frac{|V^{(m)}_{clean}[d_i]|}{|V^{(l)}[d_i]|}$$

In Example 6, this would assign a weight of 0.5 to both `NULL` $\mapsto$ `John Doe` and `NULL` $\mapsto$ `Jane Smith`. Intuitively, the weight captures the fraction of rows represented by the edge.

7.2 Predicate Probability Estimation

Let $cond(d_{clean})$ be a predicate and each predicate can be expressed as a subset of the clean distinct values $M_{pred} \subseteq M$. Each M_{pred} will have a parent set L_{pred}, that is the set of vertices in L with an edge to exactly one $m \in M_{pred}$. Together $cut = (L_{pred}, M_{pred})$ defines a cut on the bipartite graph. Before, we simply counted the number of vertices in L_{pred} to find l. Now, we need to account for the fraction of rows represented by the edges:

$$l = \sum_{l \in L_{pred}, m \in M_{pred}} w_{lm}$$

In the case, where the graph is fork-free this will reduce to the result in the previous section $w_{lm} \in \{0, 1\}$. Let $N = |L|$ and $r[d_i] \in L_{pred}$, then the revised true positive probability and false positive probability are:

$$\tau_p = (1-p) + p\frac{l}{N} \quad \tau_n = p\frac{l}{N}$$

The values can be plugged into the estimator described in the Section 5. This estimate is also asymptotically unbiased since the weights will tend to a fixed-point as the dataset grows in size.

7.3 Efficiency: Quadratic Space, Linear Time

Interestingly enough, the time- and space-complexity of the multi-attribute case is different from the single-attribute case. Since the graph is not fork-free the resulting attribute can have as many as $distinct(g_i)$ values that link to those in the dirty relation (follows from determinism). Let $\hat{N} \leq distinct(g_i)$ be the fraction of distinct tuple values affected by the cleaning operations. For the resulting graph implemented with a hash map (cleaned value $\mapsto$ dirty value), the space complexity of this graph is quadratic in $\hat{N}$, and the query complexity is linear in the distinct-value selectivity $l' = |M_{pred}|$.

PROPOSITION 4. *For multiple attributes cleaned with deterministic user-defined merge and transform operations. Let $\hat{N}$ be the total number of affected distinct values by these operations. The resulting provenance graph can be stored with $O(\hat{N}^2)$ space and for a predicate with distinct-value selectivity l' can be queried in $O(l')$ time.*

8. EXPERIMENTS

This section presents several experiments on real and synthetic datasets to evaluate PrivateClean. First, we show that the weighted estimator for PrivateClean is more accurate than the baseline Direct estimator, even when there are no data errors. We use a synthetic dataset and vary all of the parameters to show the tradeoffs. Next, we explore how privacy amplifies some types of data error and compare the estimates of Direct and PrivateClean. Finally, we present data cleaning and privacy applications on two real datasets.

8.1 Metrics and Setup

We implemented PrivateClean in Python 2.7 on a MacOSX i7 2.7GHZ, and the provenance algorithm was implemented using a standard Python dictionary stored in memory. To the best of our knowledge, there are no examples of a general purpose differentially private data cleaning framework in the literature. Consequently, the primary goal is to evaluate the proposed estimators for `sum`, `count`, and `avg`. We compare the following query processing approaches, both using GRR:

Direct: The Direct approach runs data cleaning on a private relation and then runs the query and returns the nominal result (no reweighting).

PrivateClean: The proposed approach in this paper including the data cleaning and weighted approximate query processing.

Error %: Let r be the result of running either PrivateClean or Direct. Let $r*$ be the result of running a query on a *cleaned non-private* dataset. The relative error % is $error = \frac{|r - r*|}{r*}$.

8.2 Datasets

We explore results on four datasets: Synthetic, TPC-DS, IntelWireless, and MCAFE.

Synthetic: This dataset includes a single numerical attribute $[0, 100]$ and a single categorical attribute $\{1, ..., N\}$. Both these attributes are drawn from a Zipfian distribution with scale parameter z. We vary N, z, and the privacy parameters to evaluate the estimation approaches. In particular, we use the synthetic dataset to validate the extremal behavior of PrivateClean and the regimes in which it performs well. In Appendix D, we list the default parameters. Unless otherwise noted, we fix the default parameters and vary one of the variables.

TPC-DS: We use this TPC-DS benchmark to construct constraint-based data cleaning use cases. We focus on a projection of one table: `customer_address(ca_city, ca_county, ca_state, ca_country)`. There are two data quality constraints on this table, one is a functional dependency:

$$[ca\_city, ca\_county] \mapsto [ca\_state]$$

There is also a matching dependency on country:

$$MD([ca\_country] \approx [ca\_country])$$

We introduce errors into the relation where we randomly replace *ca_state* and append one-character corruptions to *ca_country*. We solve for the constraints and implications using a standard repair algorithm [6]. Such algorithms are typically approximate or heuristic, and in this experiment, we measure accuracy with respect to ground truth (since the data cleaning itself may be fallible).

IntelWireless [29]: This dataset includes 2.3 million observations of sensor environment sensor measurements indexed by sensor id. It includes missing values and outliers. We simulate a task where we want to keep the sensor id private but still allow for cleaning of the outliers. This dataset is characterized by a small number of

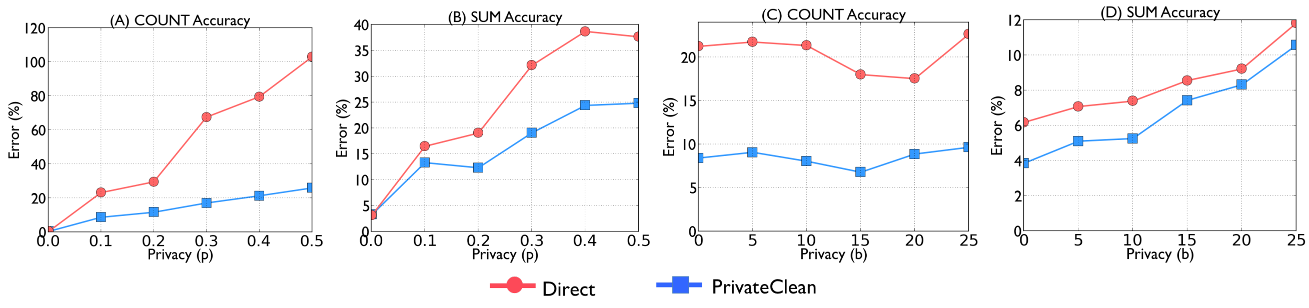

Figure 2: Relative query result error as a function of the privacy parameters (p,b). Even when there are no data errors, PrivateClean is more accurate than the Direct estimate because it accounts for data skew.

distinct discrete values (68) compared to the dataset size, which is a preferred regime of PrivateClean.

MCAFE [49]: This dataset contains 406 records of student course evaluations. We look at a numerical attribute of "enthusiasm" on a scale from 1-10, and the country code of the student. We want to keep the country codes private. We simulate a data cleaning scenario where the analyst has to merge country codes from nearby countries and remove missing values.

8.3 Synthetic Experiments

8.3.1 No Data Error

We first explore the significance of this improvement without data error and then show that the gains become more significant in the presence of data error.

Function of privacy: The two privacy parameters are p (privacy in the discrete value) and b (privacy in the numerical attribute). For a predicate with 10% selectivity and skew of $z = 2$, we first varied the categorical privacy factor p, while fixing the numerical privacy factor. Figure 2a shows how the `count` varies as a function of privacy. As predicted by the analysis the relationship is roughly linear. Even without data error, we find that when the categorical privacy factor p is high, PrivateClean has nearly a 5x reduction in query error in comparison to Direct. This is not surprising in a moderately skewed dataset given Lemma 2. Figure 2b shows how the `sum` varies as a function of privacy, and we see a similar linear relationship.

Figure 2c and Figure 2d show how the errors vary as a function of the numerical privacy factor. As expected `count` is not affected by this factor. However, there is an interesting relationship for `sum` (Figure 2d). Since the re-weighting does not affect the numerical privacy factor b, these gains w.r.t naive are diminished as the variance from the numerical privacy dominates.

Function of selectivity: The next question we explore is the how the query result error varies as a function of the selectivity. We fix the numerical and discrete privacy parameters and vary the query selectivity (defined in terms of the fraction of distinct values for which the predicate is true). Figure 3a and Figure 3b show the error % of `sum` and `count` queries for PrivateClean and Direct. PrivateClean is more accurate the Direct for low selectivity queries. The mathematical reason for this is subtle. The effects of data skew average out over more general predicates.

Function of skew: Next, we vary the skew of the data distribution by changing the Zipfian parameter z (Figure 4). PrivateClean is increasingly more accurate than Direct in skewed datasets. In fact,

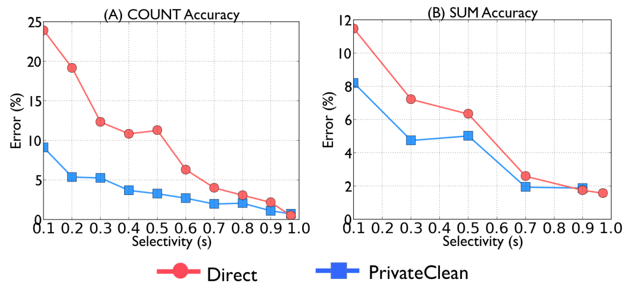

Figure 3: PrivateClean is more accurate than the Direct especially at lower selectivities.

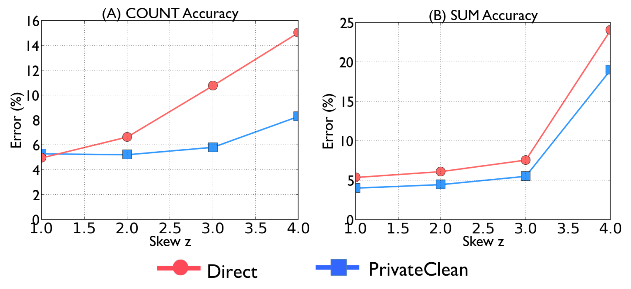

Figure 4: PrivateClean is more accurate than the Direct in skewed datasets

for count queries, there is no benefit to re-weighting results under a uniform distribution. This result is predicted by our analysis (Section 5.2).

8.3.2 Data Error

In the next set of experiments, we vary the frequency and type of data error.

Function of Error Rate: Next, we simulated data errors. For a random fraction of the distinct values mapped them to other distinct values. In Figure , we vary this fraction (error rate) while holding all of the other parameters fixed. Since Direct does not consider the provenance of the data, as the error rate increase its estimate is increasingly erroneous. On the other hand, PrivateClean keeps a constant relative error (the only error is due to the privacy factor) as the error rate is varied.

This experiment is also interesting in absolute terms. For both `sum` and `count`, PrivateClean provides a result with less than 10% error across all data error rates. This shows that in absolute

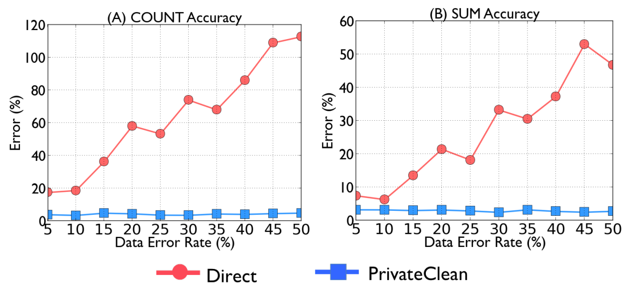

Figure 5: As the frequency of data errors increase, PrivateClean is increasingly more accurate than Direct.

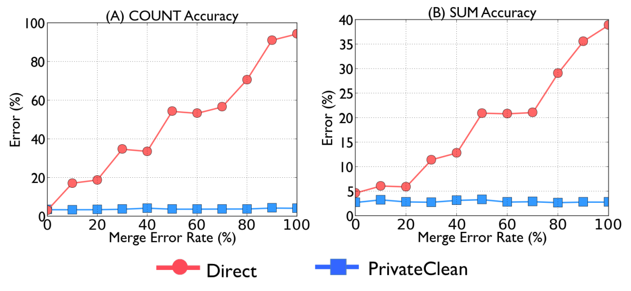

Figure 6: PrivateClean uses provenance to estimate the selectivity of queries on the original (uncleaned) data. Consequently, Direct is more sensitive to errors that "merge" distinct values.

terms the estimates from PrivateClean are useful. It also highlights a counter-intuitive property, where a query on the original dataset (no cleaning and no privacy) can be more erroneous than PrivateClean (cleaning and privacy). Privacy induces query result error while data cleaning mitigates error. This is a tradeoff that has been highlighted before in the SampleClean project [23,44].

Error Type: Of course, provenance is most valuable when the cleaned values are merged together. Next, we randomly map the distinct values to new random distinct values and other distinct values. We call this the merge rate and vary this in Figure 6. PrivateClean is more accurate than Direct when merge errors are more prevalent.

Cleaning Multiple Attributes : Figure 7 illustrates how PrivateClean to estimate results after cleaning multiple attributes. Unlike the single attribute case, we have to perform a cut on a weighted graph. We generated attribute two-attribute variant of the experiment in Figure 8.3.4a. We find that the weighted cut (PC-W) gives a more accurate result than no weighted edges (PC-U) and Direct.

8.3.3 Distinct Values

PrivateClean is designed for the setting where the number of distinct values in each discrete attribute relatively small in comparison to the dataset size. We proved a bound on the necessary dataset size to ensure that all values are visible in the private relation within probability $1 - \alpha$. While this bound was necessary for the theoretical derivation of estimates and confidence intervals, it turns out that PrivateClean empirically performs well even with datasets sizes far smaller than the bound would suggest. This experiment evaluates the performance of PrivateClean as a function of the distinct fraction $\frac{N}{S}$.

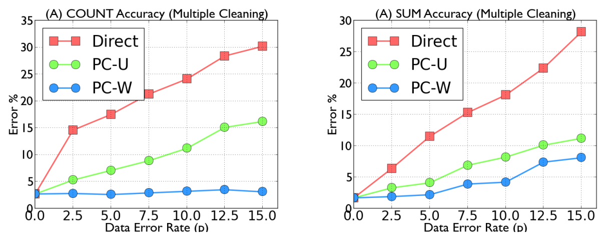

Figure 7: For cleaning operations that use information from multiple attributes, PrivateClean uses a cut on a weighted graph (PC-W) instead of an unweighted graph (PC-U).

We generated a dataset with a 5% error rate (keeping all other parameters fixed) and measure the accuracy of query results as a function of the distinct fraction. Figure 9(a-b) show the result for `sum` and `count` queries. In relative terms, PrivateClean outperforms Direct when the number of distinct values is less 50% of the dataset size. However, in absolute terms the accuracy of both methods is poor when the number of distinct values is high. Following from Figure 9, this means that as the number of distinct values increase, the relation can support (i.e., answer with the same accuracy) increasingly less selective queries.

8.3.4 Constraint-based Cleaning

Next, we evaluate PrivateClean on the constraint-based cleaning experiment with TPC-DS. We have two queries of interest that count records by the state and country:

```
SELECT count(1) FROM R GROUP BY ca_state
SELECT count(1) FROM R GROUP BY ca_country
```

In Figure 8a, we plot the results of PrivateClean as a function of the number of simulated corruptions to `ca_state`. These corruptions were fixed using a functional dependency on (`ca_city`,`ca_county`). Unlike before, this experiment illustrates the use of a heuristic data cleaning algorithm that does not perfectly clean the data, since it may not be possible to get a unique minimal repair. As the data error rate grows so does the error from PrivateClean, which we did not see in Figure a. As in previous experiments, we find that PrivateClean answers queries more accurately than Direct.

In Figure 8b, we plot the results of PrivateClean as a function of the number of simulated corruptions to `ca_country`. These corruptions were fixed using a matching dependency with a similarity metric using Edit Distance. Furthermore, matching dependencies can be resolved uniquely so the problems with the imperfect cleaning are not seen. We find that PrivateClean answers queries more accurately than Direct, and since this constraint merges values in the domain the difference between PrivateClean and Direct is more than in Figure 8a.

8.4 IntelWireless Experiments

Using the IntelWireless dataset, we simulate an analysis use case with PrivateClean. This dataset is a union of time series measuring environmental statistics (temperature, humidity, light) from 68 sensors (identified by a field `sensor_id`). Sensors that occasionally fail and during these failures their logs do not report a sensor id. As a result, in the `sensor_id` field there are missing or spurious values. Furthermore, during these failures statistics received from the sensor are untrustworthy. For data cleaning, we explored identifying and removing these log entries. We merged all of the spurious values to `null`.

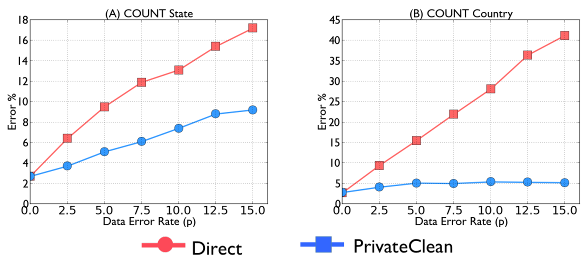

Figure 8: PrivateClean can also apply to constraint-based cleaning such using as functional dependencies and matching dependencies, and we present the results on `customer_address` table of TPC-DS.

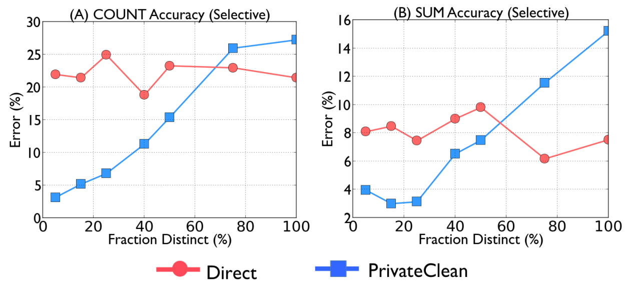

Figure 9: PrivateClean is sensitive to the number of distinct values. As the fraction of distinct values increases, the accuracy degrades. In fact, Direct becomes a more accurate estimator after a point.

We are interested in answering the following queries:

```
SELECT count(1) FROM R WHERE sensor_id != NULL
SELECT avg(temp) FROM R WHERE sensor_id != NULL
```

Figure 10 shows the query error for both queries as a function of privacy. We accordingly scale the numerical privacy parameter b such that both attributes have the same ϵ privacy parameter. As in the synthetic experiment, PrivateClean is more accurate than the Direct estimator. For reference in gray, we plot the error of the query result without data cleaning on the original dataset (without privacy). There is a counter-intuitive point at which queries on the cleaned private dataset are more accurate than queries on the dirty original dataset. Privacy introduces error while data cleaning simultaneously mitigates errors. In absolute terms, the error in these queries is relatively small $< 10\%$; demonstrating that cleaning and querying the private relation has practical utility.

8.5 MCAFE Experiments

The next experiment presents results on a much harder dataset of course evaluations. In this dataset, the number of distinct values is relatively high compared to the total number of records (21%). Consequently, the estimates for this dataset have a much larger error.

In this experiment, we demonstrate the flexibility of PrivateClean. The bipartite graph in PrivateClean can represent data transformations that go beyond traditional notions of data cleaning, such as merging together semantically similar distinct values. The MCAFE dataset has two attributes `score` and `country`. The vast majority of students are from the United States. We might be interested in aggregating students from all European countries together for comparison to the students from the United States. Like

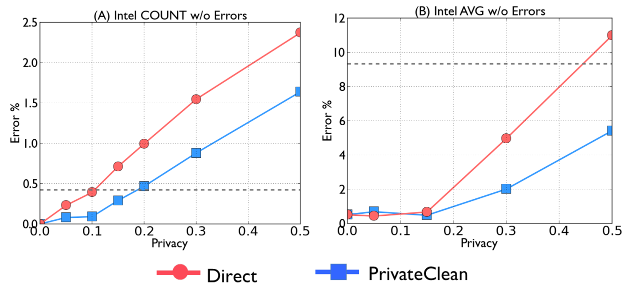

Figure 10: PrivateClean demonstrates higher accuracy in the Intel Wireless dataset validating previous results on a simulated datset.

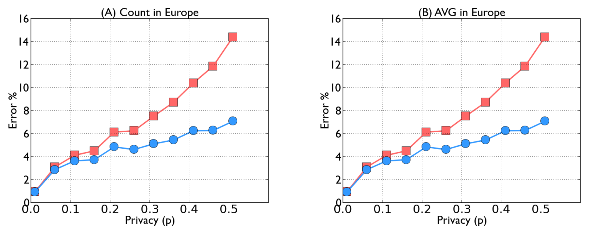

Figure 11: PrivateClean demonstrates higher accuracy on the MCAFE dataset on a data transformation task with a high fraction of distinct values.

data cleaning, this transformation is only possible if the analyst has access to the values in the private table (facilitated by GRR).

We measure the accuracy of the following queries as a function of privacy:

```
SELECT count(1) FROM R WHERE isEurope(country)
SELECT avg(score) FROM R WHERE isEurope(country)
```

The interesting point about these queries is that they cannot be answered on the untransformed data. Figure plots the error as a function of the privacy, and as in the synthetic experiments PrivateClean is more accurate than Direct. Both techniques return less than 10% query error on the private cleaned table.

9. RELATED WORK

Privacy and Data Cleaning: There have been a limited number of works that study the problem of data cleaning under privacy. However, these works do not consider differential privacy. Talukder et al. explore using cryptographic protocol to detect constraint violations in a database [43]. Similarly, Jaganathan and Wright explored using similar cryptographic protocols for secure data imputation [20].

Randomized Response: Even before the formalization of ϵ-differential privacy, statistical notions of data privacy have been well studied. Warner proposed the randomized response model in 1965 [45]. Some of the earliest work in the database community was on the topic of "data swapping" to increase privacy [40], where projections of a relation are randomly swapped. This model was the inspiration for the seminal work by Agarwal and Srikant on Privacy Preserving Data Mining (PPDM) [5]. This work led to several other results such as algorithms for classification on private data [10], statistical reconstruction of a dataset [18]. The contribution of our work, PrivateClean, is to: (1) formalize a similar mechanism using the new mathematical framework of differential privacy,

(2) understanding how data cleaning interacts with this model, (3) tight finite-sample guarantees for SQL aggregate query processing on private relations.

Privacy and Databases: In general, privacy is one of the fundamental subjects of research in the database community [19]. There are a number of surveys describing deterministic techniques for privacy preserving analytics [4,13]. These techniques include value aggregation, value suppression, and homeomorphic encryption [26,28,38,42].

Recently, the ϵ-differential privacy model [11] has renewed interest in privacy preserving analytics. McSherry proposed the seminal work of Privacy Integrated Queries (PINQ) [30]. PINQ proposed an algebra for describing privacy preserving data analytics, and ensuring the results of queries (primarily group by aggregates) would be kept differentially private. Similarly, the Airavat project studied this problem in the MapReduce setting [41]. The GUPT project explored the problem of setting ϵ (a "privacy budget") given desired accuracy guarantees [31]. Differential privacy also has been studied in a number of different contexts in databases including indexing and set-valued queries [7,27,37].

In PrivateClean, we explore the local model for ϵ differential privacy. The subtle problem of related work is that queries can only be issued a single time, and multiple runs can de-randomize a private result. On the other hand, in the local privacy model, we create a single private view of the dataset, which the analyst can query and clean an arbitrary number of times. Since the randomness is added up-front, the downside is that there are complex noise-propagation dynamics to consider. That said, PrivateClean only scratches the surface in the understanding of data error and privacy, and this is an important open problem that Getoor et al. described w.r.t entity resolution [15].

Approximate Query Processing: Approximate query processing (AQP) is a well studied problem [3,14,35] in which results of aggregate queries are approximated from samples, wavelets, or sketches. The goal of AQP is to achieve tight bounds on the results of the aggregate queries. The link between AQP and differential privacy has been studied before, and Xiao et al. [46] explore the differentially private implications of wavelet transforms. Likewise, the link between AQP and data cleaning has been studied, where Wang et al. and Krishnan et al. [23–25,44] show that samples of clean data can be used to estimate aggregate queries. PrivateClean tries to make the connection between AQP, data cleaning, and privacy. We also believe that PrivateClean will be increasingly relevant in human-in-the-loop systems as crowds are increasingly employed in data cleaning [16] and inadvertent data leaks can actually bias the crowds response [22].

10. EXTENSIONS

We highlight several points for discussion and opportunities for future work. Differential Privacy is best suited for aggregate query processing. In Section 3.2.2, we described how PrivateClean supports `sum`, `count`, `avg` queries over a single numerical attribute with predicates over a single discrete attribute. We discuss how PrivateClean can be extended to support different types of aggregate queries with more complex query structures.

Different Aggregates: PrivateClean applies additive independent Laplace random noise to each of the numerical attributes. In addition to having a mean value of 0, this noise also has a median value of 0. Therefore, it is relatively straightforward to extend PrivateClean to support `median` and `percentile` queries. Furthermore, since this noise is independent of the data, we can exploit the fact that $var(x + y) = var(x) + var(y)$ for independent random variables to also support `var` and `std` queries. For these new aggregates, calculating confidence intervals is a bit more challenging, and require an empirical method (e.g., [3,47]). However, in its current formulation PrivateClean cannot support `min` and `max` queries.

Aggregates over Select-Project-Join Views: PrivateClean can also be extended to estimate results for aggregate queries over SPJ views of differentially private relations. Since the randomization is added independently, it is easy to derive the query processing bias correction constants $\tau_n, \tau_p, \gamma_n, \gamma_p$ for such queries. For each column in the view, we essentially can calculate the constants and multiply them together, but there are some technicalities with maintaing the appropriate provenance that we defer to future work. To calcuate confidence intervals, we can use a technique like the one proposed in [33].

11. CONCLUSION

In this paper, we presented PrivateClean a framework for data cleaning and approximate query processing on locally differentially private relations. As far as we know this is the first work to marry data cleaning and local differential privacy. As a result, we hope that this initial exploration of this problem will set the stage for the exploration of more complex query processing and data cleaning models. We proposed a privacy mechanism called Generalized Randomized Response which we showed was compatible with data cleaning operations in the form of extraction, merging, and transformations. We analyzed the sufficient size of the data required for meaningful cleaning results. One of the key insights of this work is that privacy exacerbates problems of data skew, and it is crucial to estimate query selectivity for accurate query processing. However, after data cleaning apparent selectivity of a query may change, and we maintain a provenance graph to address this problem. We showed how this analysis can be inverted to select maximal privacy levels given some constraint on query accuracy. Our empirical evaluation validated the analytical results on both synthetic and real datasets.

Acknowledgements: We would like to thank Joseph Hellerstein, Joseph Gonzales, Dawn Song, and Cynthia Dwork for discussions about this work. This research is supported in part by NSF CISE Expeditions Award CCF-1139158, DOE Award SN10040 DE-SC0012463, DARPA XData Award FA8750-12-2-0331, NSERC Discovery Grant (No. 19579), and an SFU President's Research Start-up Grant (NO. 877335), and gifts from Amazon Web Services, Google, IBM, SAP, The Thomas and Stacey Siebel Foundation, Apple Inc., Arimo, Blue Goji, Bosch, Cisco, Cray, Cloudera, Ericsson, Facebook, Fujitsu, Guavus, HP, Huawei, Intel, Microsoft, Pivotal, Samsung, Schlumberger, Splunk, State Farm and VMware.

12. REFERENCES

[1] Exclusive: Apple ups hiring, but faces obstacles to making phones smarter. http://www.reuters.com/article/2015/09/07/us-apple-machinelearning-idUSKCN0R71H020150907.

[2] Netflix prize. http://www.netflixprize.com/.

[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*, 2013.

[4] C. C. Aggarwal and P. S. Yu. A general survey of privacy-preserving data mining models and algorithms. In *Privacy-Preserving Data Mining - Models and Algorithms*. 2008.

[5] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD*, 2000.

[6] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD*, 2005.

[7] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *PVLDB*, 4(11), 2011.

[8] Z. Chen and M. J. Cafarella. Integrating spreadsheet data via accurate and low-effort extraction. In *KDD*, 2014.

[9] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3, 2013.

[10] W. Du and J. Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *KDD*, 2003.

[11] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4), 2014.

[12] P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39(3), 1992.

[13] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4), 2010.

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

[15] L. Getoor and A. Machanavajjhala. Entity resolution: Theory, practice and open challenges. *PVLDB*, 5(12), 2012.

[16] D. Haas, S. Krishnan, J. Wang, M. J. Franklin, and E. Wu. Wisteria: Nurturing scalable data cleaning infrastructure. *PVLDB*, 8(12), 2015.

[17] P. J. Haas, J. F. Naughton, S. Seshadri, and A. N. Swami. Selectivity and cost estimation for joins based on random sampling. *J. Comput. Syst. Sci.*, 52(3), 1996.

[18] Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *SIGMOD*, 2005.

[19] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7), 2014.

[20] G. Jagannathan and R. N. Wright. Privacy-preserving imputation of missing data. *Data Knowl. Eng.*, 65(1), 2008.

[21] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise data analysis and visualization: An interview study. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12), 2012.

[22] S. Krishnan, J. Patel, M. J. Franklin, and K. Goldberg. A methodology for learning, analyzing, and mitigating social influence bias in recommender systems. In *RecSys*, 2014.

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

[24] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, T. Kraska, T. Milo, and E. Wu. Sampleclean: Fast and reliable analytics on dirty data. *IEEE Data Eng. Bull.*, 38(3), 2015.

[25] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: Interactive data cleaning while learning convex loss models. In *Arxiv: http://arxiv.org/pdf/1601.03797.pdf*, 2015.

[26] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*. IEEE, 2007.

[27] N. Li, W. H. Qardaji, D. Su, and J. Cao. Privbasis: Frequent itemset mining with differential privacy. *PVLDB*, 5(11), 2012.

[28] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *TKDD*, 1(1), 2007.

[29] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1), 2005.

[30] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9), 2010.

[31] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. E. Culler. GUPT: privacy preserving data analysis made easy. In *SIGMOD*, 2012.

[32] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008.

[33] S. Nirkhiwale, A. Dobra, and C. M. Jermaine. A sampling algebra for aggregate estimation. *PVLDB*, 6(14), 2013.

[34] G. W. Oehlert. A note on the delta method. *The American Statistician*, 46(1), 1992.

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

[36] H. Park and J. Widom. Crowdfill: collecting structured data from the crowd. In *SIGMOD*, 2014.

[37] S. Peng, Y. Yang, Z. Zhang, M. Winslett, and Y. Yu. Dp-tree: indexing multi-dimensional data under differential privacy. In *SIGMOD*, 2012.

[38] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Symposium on Operating Systems Principles, Cascais, Portugal*, 2011.

[39] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4), 2000.

[40] S. P. Reiss, M. J. Post, and T. Dalenius. Non-reversible privacy transformations. In *PODS*, 1982.

[41] I. Roy, S. T. V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel. Airavat: Security and privacy for mapreduce. In *NSDI*, 2010.

[42] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5), 2002.

[43] N. Talukder, M. Ouzzani, A. K. Elmagarmid, and M. Yakout. Detecting inconsistencies in private data with secure function evaluation. 2011.

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

[45] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309), 1965.

[46] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. Knowl. Data Eng.*, 23(8), 2011.

[47] K. Zeng, S. Gao, B. Mozafari, and C. Zaniolo. The analytical bootstrap: a new method for fast error estimation in approximate query processing. In *SIGMOD*, 2014.

[48] E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *WWW, Madrid, Spain*, 2009.

[49] M. Zhou, A. Cliff, A. Huang, S. Krishnan, B. Nonnecke, K. Uchino, S. Joseph, A. Fox, and K. Goldberg. M-cafe: Managing mooc student feedback with collaborative filtering. In *Learning@ Scale*. ACM, 2015.

APPENDIX

A. PROOFS

A.1 Proof of Lemma 1

We directly apply the definition of local differential privacy. Let $N =\mid Domain(g_i) \mid$:

$$\epsilon = \ln \frac{1 - p_i + p_i \frac{1}{N}}{p_i \frac{N-1}{N}}$$

The worst case is when there are only two values in the domain and all of the other entries in the database are one value except for one, then this gives us $N = 2$, plugging it in to the above formula:

$$\epsilon = \ln(\frac{3}{p_i} - 2)$$

A.2 Consequences of Differential Privacy

Lemma 2, Theorem 1, and Theorem 2 follow from well-established results in Differential Privacy (see [11]). We provide detailed references for the results here.

Proposition 1: This follows from Definition 3.3 in [11].

Theorem 1: This follows from Theorem 3.16 in [11].

Theorem 2: Proposition 2.1 in [11] shows that for any deterministic function applied to a differentially private result, differential privacy is preserved.

A.3 Proof of Theorem 3

For some domain value a, let us first start with the probability that a is visible in the private relation:

$$\mathbf{P}[a] = 1 - p\frac{N-1}{N}[1-\frac{p}{N}]^{(S-1)}$$

For all domain values, we can apply a union bound:

$$\mathbf{P}[all] \geq 1 - p(N-1)[1-\frac{p}{N}]^{(S-1)}$$

Setting $\mathbf{P}[all] = 1-\alpha$:

$$\alpha \leq p(N-1)[1-\frac{p}{N}]^{(S-1)}$$

With some algebra, we can show:

$$\log\frac{\alpha}{p(N-1)}\frac{1}{\log 1-\frac{p}{N}} \leq S$$

Using the inequality $\frac{x-1}{x} \leq \log x$, we can arrive at the bound presented in the paper.

A.4 Proof of Lemma 3

We start with the predicate false positive and false negative probabilities. The false positive probability is:

$$\mathbf{P}(\text{false positive}) = (1-s)p\frac{l}{N}$$

and the false negative probability is:

$$\mathbf{P}(\text{false negative}) = sp\frac{N-l}{N}$$

For `sum` and `count`, the bias is proportional to the predicate errors. Since the Naive estimator sets two values equal, we can calculate the difference:

$$bias = (1-s)p\frac{l}{N} - sp\frac{N-l}{N}$$

$$bias = \frac{p(l-sN)}{N}$$

This value will be scaled by a constant related to the dataset size:

$$\tilde{O}(\frac{p(l-sN)}{N})$$

B. ESTIMATOR NOTATION

- $c_{private}$: Count on the private data
- c_{true}: Count on the original data
- μ_{true}: Average on the true data
- S: Dataset size
- N: Number of distinct values in the attribute partition g_i that contains the predicate attribute.
- $h_{private}$: sum on the private data
- h_{true}: sum on the true data
- $h^c_{private}$: sum on the private data (with predicate negated).
- h^c_{true}: sum on the true data (with predicate negated).

C. ESTIMATOR DERIVATIONS

COUNT: To prove this, first, we formulate an unbiased estimator:

$$\mathbb{E}(c_{private}) = c_{true}\tau_p + (S - c_{true})\tau_n$$

$$\mathbb{E}(c_{private}) = c_{true}(\tau_p - \tau_n) + S\tau_n$$

$$\frac{\mathbb{E}(c_{private}) - S\tau_n}{(\tau_p - \tau_n)} = c_{true}$$

SUM: To prove this, first, we formulate an unbiased estimator:

$$\mathbb{E}(h_{private}) = c_{true}[(1-p)+p\frac{l}{N}]\mu_{true}+(S-c_{true})(p\frac{l}{N})\mu_{false}$$

We are interested in estimating the count which is $c_{true}\mu_{true}$:

$$\mathbb{E}(h_{private}) = [(1-p)+p\frac{l}{N}]c_{true}\mu_{true}+(S-c_{true})(p\frac{l}{N})\mu_{false}$$

The challenge is that there are two unknown variables $c_{true}\mu_{true}$ and μ_{false}, so we can apply a trick where we also calculate the sum over the complement and solve:

$$\mathbb{E}(h^c_{private}) = [p\frac{N-l}{N}]c_{true}\mu_{true}+(S-c_{true})(1-p+p\frac{N-l}{N})\mu_{false}$$

This is a linear system of equations, and solving it arrives at:

$$\begin{bmatrix} h_{private} \\ h^c_{private} \end{bmatrix} = \begin{bmatrix} (1-p)+p\frac{l}{N} & (S-c_{true})(p\frac{l}{N}) \\ p\frac{N-l}{N} & (S-c_{true})(1-p+p\frac{N-l}{N}) \end{bmatrix} \begin{bmatrix} c_{true}\mu_{true} \\ \mu_{false} \end{bmatrix}$$

$$c_{true}\mu_{true} = \frac{(1-p+p\frac{N-l}{N})h_{private} - p\frac{l}{N}h_{private}}{1-p}$$

With a little bit of algebra, we can arrive at:

$$c_{true}\mu_{true} = \frac{(N-lp)h_{private} - (lp)h^c_{private}}{(1-p)N}$$

D. SYNTHETIC EXPERIMENTS DEFAULT PARAMETERS

We list the default parameters in the synthetic experiment below. Unless otherwise noted, these are the parameters which we use. For all experiments, we create 100 random instances of the private database, and for each instance we run a randomly selected query. The plots show a mean value over the 100 random instances.

Table 1: Default parameters in the synthetic experiment

Symbol	Default Value	Meaning
p	0.1	Discrete privacy parameter
b	10	Numerical privacy parameter
N	50	Number of distinct values
S	1000	Number of total records
l	5	Distinct values selected by predicate
z	2	Zipfian Skew

E. PARAMETER TUNING ALGORITHM

Following from the analysis in the paper, we derive a tuning algorithm to set ϵ by setting p_i and b_i for each attribute.

Inputs: error the user gives us the desired maximum error in any count query on the relation with $1 - \alpha$ confidence intervals.

Output: For each numerical attribute j return b_j the numerical privacy parameter, and for each discrete attribute partition g_i return a p_i the discrete privacy parameter.

1. Let $p = 1 - z_\alpha \sqrt{\frac{1}{4S \cdot error^2}}$, where z_α is the Gaussian tail parameter (e.g, $95\% \approx 1.96$).
2. For each discrete attribute partition g_i, set p_i to p.
3. For each numerical attribute j, set $b_j = \frac{\Delta_j}{\ln \frac{3}{p} - 2}$, where Δ_j is the difference between the max and min values of that attribute.