DP-PQD: Privately Detecting Per-Query Gaps In Synthetic Data Generated By Black-Box Mechanisms

Shweta Patwa
Duke University
sjpatwa@cs.duke.edu

Danyu Sun
Duke University
ds592@cs.duke.edu

Amir Gilad
Hebrew University
amirg@cs.huji.ac.il

Ashwin Machanavajjhala
Duke University
ashwin@cs.duke.edu

Sudeepa Roy
Duke University
sudeepa@cs.duke.edu

ABSTRACT

Synthetic data generation methods, and in particular, private synthetic data generation methods, are gaining popularity as a means to make copies of sensitive databases that can be shared widely for research and data analysis. Some of the fundamental operations in data analysis include analyzing aggregated statistics, e.g., count, sum, or median, on a subset of data satisfying some conditions. When synthetic data is generated, users may be interested in knowing if their aggregated queries generating such statistics can be reliably answered on the synthetic data, for instance, to decide if the synthetic data is suitable for specific tasks. However, the standard data generation systems do not provide "per-query" quality guarantees on the synthetic data, and the users have no way of knowing how much the aggregated statistics on the synthetic data can be trusted. To address this problem, we present a novel framework named *DP-PQD (differentially-private per-query decider)* to detect if the query answers on the private and synthetic datasets are within a user-specified threshold of each other while guaranteeing differential privacy. We give a suite of private algorithms for per-query deciders for count, sum, and median queries, analyze their properties, and evaluate them experimentally.

PVLDB Reference Format:
Shweta Patwa, Danyu Sun, Amir Gilad, Ashwin Machanavajjhala, and Sudeepa Roy. DP-PQD: Privately Detecting Per-Query Gaps In Synthetic Data Generated By Black-Box Mechanisms. PVLDB, 17(1): 65 - 78, 2023.
doi:10.14778/3617838.3617844

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/sjpatwa/dp-pqd.

1 INTRODUCTION

For more than a decade, we have witnessed an abundance of data containing private and sensitive information and a growing interest in using this data for decision making and data analytics. Formal policies like the GDPR [19] and CCPA [5] require that the privacy of the individuals whose data is being used be maintained. Differential privacy (DP) [10] is the gold standard in offering mathematically rigorous bounds on privacy leakage while offering utility through multiple data releases, even in the presence of side information. Intuitively, DP guarantees that the output has a similar distribution whether an individual's data is used. It has been widely adopted by many organizations [2, 11] and leading companies [7, 15, 39].

Private data can be queried directly using designated DP mechanisms. However, the accuracy of the results depends heavily on the privacy budget, especially when multiple queries need to be answered. Furthermore, the results may be inconsistent with each other. A prominent alternative is using differentially-private *synthetic data generators (SDGs)* to produce a synthetic copy of the private data, which can be used repeatedly to answer multiple queries without spending additional privacy budget. Previous works have employed techniques from game theory [18, 21, 41], probabilistic graphical models [4, 24, 29, 31, 43], and deep learning [20, 37]. On the other hand, to generate new instances of datasets resembling properties of the given dataset, SDGs not satisfying DP have also become very popular alternatives in applications where it is appropriate to do so. Examples include SDGs using generative modeling [35] and deep learning techniques [34, 42]. Synthetic data offers advantages such as: (1) consistency in answering a large number of statistical queries, (2) preservation of desired correlations within data, and (3) concise representation of the private data that circumvents expending more privacy budget to answer queries.

While SDGs (DP or not) embody a promising approach for increasing the usability of private data, there may exist discrepancies between query results over the private and synthetic datasets. This work in particular focuses on analyzing aggregated statistics, e.g., count, sum, or median, on a subset of data satisfying some conditions, which form some of the most fundamental operations in data analysis. When synthetic data produced by a SDG is used in data analysis, users may be interested in knowing if their aggregated queries generating such statistics can be reliably answered on said data. However, the standard data generation systems do not provide "per-query" quality guarantees on the synthetic data, and the users have no way of knowing how much the aggregated statistics on the synthetic data can be trusted. We illustrate with an example below.

Example 1.1. *Let the private database D be a simplified version of the Adult database [9] with attributes:* `age`, `education`, `capital-gain`, `marital-status`, `occupation`, `relationship`, and `sex`. *Let D_s denote the synthetic copy of D from PrivBayes [43], which is a DP SDG. Also consider the following query q, where (*<a>, <b>*) is one of* $(0, 200)$, $(200, 400)$, $(400, 600)$, $(600, 800)$ *or* $(800, 1000)$*:*

```
SELECT COUNT(*) FROM D_s
WHERE capital-gain ≥ <a> AND capital-gain < <b>.
```

Figure 1 shows the output for the aforementioned values of <a> *and* <b>*. Bars with height* 0 *are not shown here. Note that the corresponding counts from D are private and we include them here only for reference.*

Proceedings of the VLDB Endowment, Vol. 17, No. 1 ISSN 2150-8097.
doi:10.14778/3617838.3617844

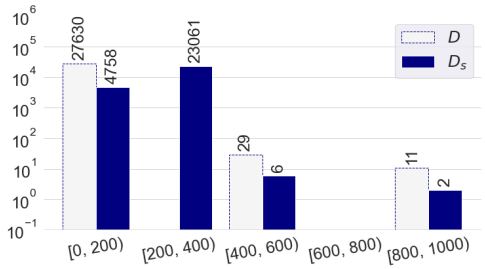

Figure 1: Histogram for attribute *capital-gain* **in** D **(in white, private and not visible to the user) and** D_s **(in navy blue, visible to the user) for bins given by** $(0, 200)$, $(200, 400)$, $(400, 600)$, $(600, 800)$ **and** $(800, 1000)$**.**

Suppose the user wants to know if the gap between $q(D)$ *and* $q(D_s)$ *is less than* $\tau = 200$. ***How can the user find out if their distance bound for query*** q ***is met without access to either the private data or the SDG that was used to produce the synthetic data?*** *In Figure 1, we see that when* (<a>, <b>) $= (400, 600)$*, the (true) answer is 'Yes', whereas when* (<a>, <b>) $= (0, 200)$*, the (true) answer is 'No'. In fact, the distance is* 114.36 *times* 200.

In this paper, we aim to build a *"Differentially-Private Per-Query Decider"*, which gives a *'Yes' (distance bound is satisfied)* answer if $|q(D) - q(D_s)|$ is smaller than a given distance bound $\tau > 0$, and a *'No' (distance bound is unmet)* answer otherwise, while satisfying DP with a given privacy budget $\epsilon > 0$. However, there are several challenges that one needs to address. *First*, we assume that we do not have access to the mechanism behind the SDG producing D_s. All the per-query decider can see is the synthetic data D_s generated by the SDG. This SDG may be one of the SDGs satisfying DP while outputting D_s using a separate privacy budget [4, 18, 20, 21, 24, 29, 31, 37, 41, 43], or a SDG that does not use DP [34, 35, 42]. The per-query decider should work with D_s generated by any SDG. *Second*, not only can the per-query decider not output the true 'Yes' or 'No' answers since $q(D)$ is private, but also it cannot give a deterministic answer because DP mechanisms must be randomized algorithms. *Third*, the per-query decider should have good accuracy – answering a random 'Yes'/'No' answer is trivially private but is not useful to the user. *Finally*, we aim to build a framework of per-query decider that can handle different types of standard aggregates, namely COUNT, MEDIAN, and SUM, which have different sensitivities on the input (private) data and will need different techniques.

Our Contributions

In this work, we propose a novel framework called *Differentially-Private Per-Query Decider (DP-PQD) to decide if the distance between* $q(D)$ *and* $q(D_s)$ *is less than a user-provided distance bound of* $\tau > 0$ *for a given query* q *and privacy budget* $\epsilon > 0$. We investigate the problem for COUNT, SUM, and MEDIAN queries under this framework (with optional predicates selecting a subset of the data), which are three fundamental aggregate operators used in data analysis. We make the following contributions.

(1) The DP-PQD framework (Section 3): We formally define the differentially-private per-query decider and introduce the notion of *effectiveness* of an algorithm to capture the range of input distance thresholds for which a per-query decider algorithm is expected to perform well.

(2) COUNT queries (Section 4): For COUNT queries, we present and analyze two approaches, one based on the Laplace Mechanism (LM) [13] that uses a DP noisy estimate of $q(D)$ to compare with $q(D_s)$, and the other is direct approach for answering 'Yes' or 'No' based on the Exponential Mechanism (EM) [32] using a carefully designed score function.

(3) SUM queries (Section 5): For SUM queries, we present and analyze three approaches. Two of them based on the Laplace Mechanism (LM) [13] and the recent *Race-to-the-Top* Mechanism (R2T) [8] use a DP noisy estimate of $q(D)$ to compare with $q(D_s)$. The third direct approach exploits the *Sparse Vector Technique* (SVT) [26] originally designed to detect when the first of a sequence of queries exceeds a given threshold to implement a per-query decider.

(4) MEDIAN queries (Section 6): For MEDIAN queries, we present and analyze two approaches: one uses a DP noisy estimate of the median query using EM, and the other is a new histogram-based DP mechanism that directly solves the problem using the LM.

(5) Experimental evaluation (Section 7): We have implemented the DP-PQD framework with all the above algorithms to evaluate our proposed solutions. We analyze the accuracy for 22 COUNT queries (with a range of tuple selectivity), 19 SUM queries (with a range of tuple selectivity and varying downward local sensitivities), and 19 MEDIAN queries (with different data distribution around the true median). One of the interesting observations is that the error of a DP per-query decider is not always monotonic in the privacy budget ϵ, and we explain why this happens.

2 PRELIMINARIES

In this section, we review some background concepts and present notations used in the rest of the paper.

2.1 Data and Queries

We are given a private database instance D that comprises a single relation with attributes $A_1, \ldots, A_d$. The domain of attribute A_i is given by $dom(A_i)$, which is categorical or integral.

In this paper, we consider three aggregate operators: COUNT, SUM, and MEDIAN, i.e., the corresponding aggregate queries in SQL with an optional WHERE clause take the following form:

- SELECT COUNT(*) FROM D WHERE φ,
- SELECT SUM(A_i) FROM D WHERE φ, and
- SELECT MEDIAN(A_i) FROM D WHERE φ.

In these queries, first the predicate φ (if the WHERE clause exists) is applied over all tuples in D, and then the aggregate is computed on tuples that satisfy φ. These queries output single real output value and belong to the class of *scalar queries*.

Example 2.1. *Recall the private database* D *and its synthetic copy* D_s *from Example 1.1. Consider the following queries:*

- q_1: SELECT COUNT(*) FROM D_s WHERE *age* > 30 AND *education* LIKE 'Masters'.

- q_2: SELECT SUM(*capital-gain*) FROM D_s WHERE *education* LIKE '12th'.
- q_3: SELECT MEDIAN(*capital-gain*) FROM D_s.

Query q_1 asks for the number of people with age above 30 and Master's degree, q_2 asks for the total capital-gain of people with 12-th grade education, and q_3 asks for the median of capital-gain over all people.

2.2 Differential Privacy

We use *Differential Privacy (DP)* [10] as the measure of privacy. We say that two databases D and D' are *neighbors* if they differ by a single tuple. This is denoted by $D \approx D'$.

Definition 2.2 (Differential Privacy [14]). *A randomized mechanism $\mathcal{M}$ is said to satisfy ϵ-DP if $\forall S \subseteq Range(\mathcal{M})$ and $\forall D, D'$ pair of neighboring databases, i.e., $D \approx D'$,*

$$Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(D') \in S]$$

Smaller ϵ gives stronger privacy guarantee.

Definition 2.3 (Global Sensitivity). *For a scalar query q, its global sensitivity is given by $\Delta q = \max_{D \approx D'} |q(D) - q(D')|$.*

Definition 2.4 (Downward Local Sensitivity). *For a scalar query q, its downward local sensitivity on database D is given by*

$$DS_{q,D} = \max_{D' \approx D, D' \subseteq D} |q(D) - q(D')|$$

Example 2.5. *Consider the private database D and sum query q_2 from Example 2.1.Suppose dom(capital-gain) is $\{0, 1, \ldots, 99999\}$, so $\Delta q_2 = 99999$ since the maximum change in the sum over all pairs of neighboring databases is the maximum value in the domain. On the other hand, given a database D, $DS_{q,D}$ equals the largest value in capital-gain from tuples in D with education equal to 12-th.*

Properties like composition [10] and post-processing [12] give a modular way to build complex DP mechanisms:

Proposition 2.6. *[10, 12] give the following:*

(1) **(Sequential composition)** *If $\mathcal{M}_i$ satisfies ϵ_i-DP, then the sequential application of $\mathcal{M}_1, \mathcal{M}_2, \cdots$, satisfies $\sum_i \epsilon_i$-DP.*
(2) **(Parallel composition)** *If each $\mathcal{M}_i$ accesses disjoint sets of tuples, then they together satisfy $\max_i \epsilon_i$-DP.*
(3) **(Post-processing)** *Any function applied to the output of an ϵ-DP mechanism $\mathcal{M}$ also satisfies ϵ-DP.*

Laplace mechanism (LM). The Laplace mechanism [13] is a common building block in DP mechanisms and is used to get a noisy estimate for scalar queries with numeric answers. The noise injected is calibrated to the global sensitivity of the query.

Definition 2.7 (Laplace Mechanism). *Given a database D, scalar query q (with output in $\mathbb{R}$), and privacy budget ϵ, the Laplace mechanism $\mathcal{M}_L$ returns $q(D) + v_q$, where $v_q \sim Lap(\Delta q/\epsilon)$.*

Its accuracy is given by the following theorem [14].

Theorem 2.8 (Accuracy of Laplace Mechanism [14]). *Given a database D and scalar query q (with output in $\mathbb{R}$). Let y be the output from running the Laplace Mechanism $\mathcal{M}_L$ on D and q with privacy budget ϵ. Then, $\forall \delta \in (0, 1]$,*

$$Pr\left[|q(D) - y| \geq \frac{\Delta q}{\epsilon} \ln \frac{1}{\delta}\right] \leq \delta$$

Example 2.9. *Consider the private database D and query q_1 from Example 2.1. Δq_1 equals 1 because the count can change by at most one on neighboring databases. Say the output of q_1 on D is n_1, a private quantity, and we want a DP estimate, $\tilde{n}_1$, for it. $\mathcal{M}_L$ returns n_1 plus noise sampled from $Lap(1/\epsilon)$. Also, $Pr\left[|n_1 - \tilde{n}_1| \geq \frac{1}{\epsilon} \ln \frac{1}{\delta}\right] \leq \delta$.*

Exponential mechanism (EM). For categorical outputs, the Exponential mechanism [32] is used with an appropriate *score function* that gives the utility of each element in the output space with respect to the given private database D. The likelihood of an element being returned as the output depends on its score.

Definition 2.10 (Exponential Mechanism). *Given a database D, range of outputs $\mathcal{R}$, real-valued score function $u(D, e)$ that gives the utility of $e \in \mathcal{R}$ with respect to D, and privacy budget ϵ, the Exponential mechanism $\mathcal{M}_E$ returns $e \in \mathcal{R}$ with probability $c \cdot e^{\frac{\epsilon u(D,e)}{2\Delta u}}$, where c is a positive constant and Δu is the global sensitivity of u.*

Example 2.11. *Recall the private database D and query q_3 from Example 2.1. We find a DP estimate for the median [6] by applying $\mathcal{M}_E$ with $u(D, e) = -|rank(e) - n/2|$, for $e \in dom(capital\text{-}gain)$. Δu is 1, and therefore, $\mathcal{M}_E$ returns e with probability $\propto e^{\epsilon u(D,e)/2}$.*

2.3 Synthetic Data Generators

We treat the *synthetic data generator (SDG)* used to produce D_s as a black box. Our framework DP-PQD can be used in conjunction with any synthetic data generator: a standard SDG (not satisfying DP) like [34, 35, 42] that typically takes D and some optional constraints as inputs, or an SDG satisfying DP (SDG_{DP}) that takes D and a privacy budget as input. An SDG_{DP} may or may not take a set of queries as input. For example, PrivBayes [43] does not take queries as input but works such as [18, 21, 24, 27, 29, 31, 41] do. For an SDG_{DP}, we assume that it has a privacy budget separate from the privacy budget ϵ for the per-query decider DP-PQD. DP-PQD takes as input the synthetic data D_s generated from any SDG, the private database D and a privacy budget ϵ, does not need to run the SDG again, and does not assume anything about how the SDG works.

3 THE DP-PQD FRAMEWORK

In this section, we present the DP-PQD (Differentially-Private Per-Query Decider) framework that intends to solve the following problem; the workflow of DP-PQD is given in Figure 2.

Definition 3.1 (Differentially-Private Per-Query Decider). *Given a private database D, synthetic database D_s for D from a black-box SDG, query q (*COUNT, SUM *or* MEDIAN*), distance bound $\tau > 0 \in \mathbb{R}$, and privacy budget $\epsilon > 0$, return whether $|q(D) - q(D_s)| < \tau$ while satisfying ϵ-DP. We call such a mechanism a* **differentially-private per-query decider**, *or simply a per-query decider, and denote it by $\mathcal{A}(D, D_s, q, \tau, \epsilon)$, or $\mathcal{A}(D)$ when D_s, q, τ, and ϵ are clear from context.*

To simplify notation, we will write o to denote the outcome of $\mathcal{A}(D)$ when $\mathcal{A}, D, D_s, q, \tau, \epsilon$ are clear from context. Here,

$$o = 1 \quad \equiv \quad \text{"Distance bound satisfied"}$$
$$o = 0 \quad \equiv \quad \text{"Distance bound unmet"}$$

In this paper, we investigate the following approaches for $\mathcal{A}(D)$: (1) spend ϵ to obtain a noisy DP estimate for $q(D)$, and compare it

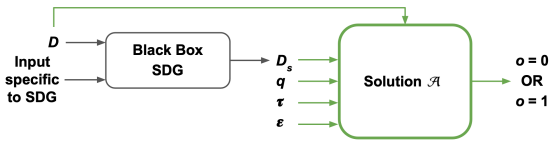

Figure 2: Workflow of DP-PQD

with $q(D_s)$ to check if the distance bound of τ is met (by instantiating Algorithm 1 with a suitable DP mechanism DPNoisy$(\cdot)$), and (2) design specialized algorithms to solve the problem without first estimating $q(D)$. A summary of our solutions is given in Table 4.

For convenience, we denote the desired interval for $q(D)$ as $\mathcal{I} := (l, r)$ (known to the user), and the absolute difference in query values of $q(D)$ and $q(D_s)$ as d_q (hidden from the user), i.e.,

$$\mathcal{I} := (l, r) = (q(D_s) - \tau, q(D_s) + \tau) \quad (1)$$

$$d_q := |q(D) - q(D_s)| \quad (2)$$

Since $q(D_s)$ is known to the user, but $q(D)$ is not, we envision the user choosing τ as a percentage value of $q(D_s)$ in practical applications. Here, $\mathcal{I} = (l, r) = (q(D_s) \cdot (1 - \tau), q(D_s) \cdot (1 + \tau))$. The τ value depends on how much error in $q(D_s)$ can be tolerated by the application using $q(D_s)$. For example, τ larger than 10% of $q(D_s)$ may introduce too much error in the downstream analysis. To present our techniques in Sections 4, 5, and 6 for a given query, we use τ as a constant, whereas in the experiments in Section 7 when we vary the queries, we use and vary τ as a percentage of $q(D_s)$ since the same value of τ may not be meaningful to all queries.

$\mathcal{A}$ satisfies ϵ-DP and like any randomized mechanism, incurs error in deciding if the distance bound is met (i.e. if $d_q < \tau$). We quantify error as the expectation of the event that $\mathcal{A}$ returns the wrong outcome on D defined as follows:

Definition 3.2 (Error). *Let $\mathcal{A}$ be a per-query decider for the given private database D, synthetic database D_s, query q, distance bound τ, and privacy budget ϵ. Then the error of $\mathcal{A}$ is given by:*

$$err_{\mathcal{A}}(q, D, D_s, \tau, \epsilon) = Pr[o = 1 \wedge d_q \geq \tau] + Pr[o = 0 \wedge d_q < \tau]$$

where the probability is over the randomness in $\mathcal{A}$.

In Definition 3.2, the first term corresponds to the *false positive rate* and the second term corresponds to the *false negative rate*. Intuitively, we expect larger error when d_q and τ are closer because the chance of the random noise injected by $\mathcal{A}$ causing the wrong outcome becomes higher.

Example 3.3. *Consider D, D_s, q from Example 1.1 where $(0, 200)$ is used as the range. Given that $\tau = 200$. Here, d_q (hidden from the user) equals 22,872 and is greater than τ, so the correct outcome of $\mathcal{A}$ is $o = 0$. Hence, a solution $\mathcal{A}$ makes a mistake if $o = 1$ and the error equals the probability that $\mathcal{A}$ returns $o = 1$.*

Next, we introduce a notion we call the *effectiveness* of a per-query decider $\mathcal{A}$. The idea is to derive a lower bound for τ such that $\mathcal{A}$ returns the correct outcome with probability at least $1 - \delta$ (for $0 < \delta < 1$) in two cases: (1) $q(D) = q(D_s)$, and (2) $q(D) \leq q(D_s) - 2\tau$ or $q(D) \geq q(D_s) + 2\tau$. The first condition ensures that $\mathcal{A}$ has high accuracy when the two query outputs match, and the second condition ensures that $\mathcal{A}$ has high accuracy also for very different query outputs. In the absence of the first case, an $\mathcal{A}$ that always returns $o = 0$ meets the condition, and in the absence of the second case, an $\mathcal{A}$ that always returns $o = 1$ meets the condition, but neither is a useful solution. We denote the lower bound by $\tau_{min}^{\mathcal{A},\delta}$ and call it the *effectiveness threshold* of $\mathcal{A}$ at δ.

Table 1: Table of Notations

Notation	Description
D, D_s	Private database and its synthetic copy
q	COUNT, SUM, or MEDIAN query
Δq	Global sensitivity of query q
$DS_{q,D}$	Downward local sensitivity of q on D (Sec. 5)
τ	Given upper bound on $\lvert q(D) - q(D_s)\rvert$
ϵ	Privacy budget
$\mathcal{I} = (l, r)$	$(q(D_s) - \tau, q(D_s) + \tau)$
$\mathcal{A}(D, D_s, q, \tau, \epsilon)$	ϵ-DP per-query decider
$o \in \{0, 1\}$	$\mathcal{A}$'s outcome on the given problem
$\tau_{min}^{\mathcal{A},\delta}$	Effectiveness threshold of $\mathcal{A}$ for $0 < \delta \ll 1$
GS_q	Upper limit on Δq for SUM query q (Sec. 5)

Definition 3.4 (Effectiveness). *Let $\mathcal{A}$ be a per-query decider for a given private database D, a synthetic database D_s, a query q, a distance bound τ, and a privacy budget ϵ. $\mathcal{A}$ is called* **effective** *at error probability $0 < \delta < 1$ if the following two conditions hold:*

(1) if $q(D) = q(D_s)$, $Pr[o = 1] \geq 1 - \delta$, and
(2) if $q(D) \notin (q(D_s) - 2\tau, q(D_s) + 2\tau)$, $Pr[o = 0] \geq 1 - \delta$.

The smallest value of τ that achieves the above is called the effectiveness threshold of $\mathcal{A}$ at δ and is denoted by $\tau_{min}^{\mathcal{A},\delta}$.

We give an upper bound on the effectiveness thresholds of each solution for COUNT and SUM queries (Section 4 and 5). We use effectiveness as proxy for error for queries with high sensitivities, like SUM queries. For MEDIAN queries, $q(D)$ shows up in the rank space in the analysis. This is an interesting direction for future work. Table 1 summarizes the notations used throughout the paper.

4 SOLUTIONS FOR COUNT QUERY

We propose two approaches for COUNT query q: (1) LM_{count} (that instantiates Algorithm 1 with the Laplace Mechanism (LM)) in Section 4.1, and (2) EM_{count} (that directly solves the problem using the Exponential Mechanism (EM)) in Section 4.2, and analyze their errors. We also derive upper bounds for their respective effectiveness thresholds. We give an error comparison in Section 4.2.

4.1 Laplace Mechanism-Based Approach

In our first algorithm, LM_{count}, we use the LM (Definition 2.7) to obtain a DP estimate for $q(D)$ and check if the noisy answer is less than τ away from $q(D_s)$. This is achieved by running GenericDecider (Algorithm 1) with the LM that adds noise from $Lap(\frac{1}{\epsilon})$ as DPNoisy (since $\Delta q = 1$ for COUNT queries). Since we post-process a DP estimate (Proposition 2.6), the following holds:

Observation 4.1. *LM_{count} satisfies ϵ-DP.*

We denote the noise injected by the LM (Definition 2.7) as v_q and analyze LM_{count}'s error next. We will frequently use the following properties of the Laplace distribution (with mean 0) [14]. For a Laplace random variable $v_q \sim Lap(\frac{1}{\epsilon})$ and for $t \geq 0$,

Algorithm 1: Basic approach using DP estimate of $q(D)$

Input : q - count/sum/median query, D - private database, D_s - synthetic database, τ - distance bound, ϵ - privacy budget, `DPNoisy` - any ϵ-DP mechanism to get a noisy estimate for $q(D)$, ϕ - any additional parameter(s) that `DPNoisy` takes.
/* *If* `DPNoisy` = *LM (Defn. 2.7), then* $\phi = \emptyset$
If `DPNoisy` = *EM (Defn. 2.10), then* $\phi = \{\mathcal{R}, u\}$
If `DPNoisy` = *R2T (Sec. 5.2), then* $\phi = \{GS_q, \beta\}$ */

Output : $o = 1$ if the desired distance bound from $q(D_s)$ is satisfied for $q(D)$, else $o = 0$.

Function `GenericDecider`$(q, D, D_s, \tau, \epsilon,$ `DPNoisy`$, \phi)$**:**
  if $-\tau <$ `DPNoisy`$(D, q, \epsilon, \phi) - q(D_s) < \tau$ **then**
    return $o = 1$ ("Distance bound satisfied");
  return $o = 0$ ("Distance bound unmet");

$$Pr\left[v_q \geq t \cdot \frac{1}{\epsilon}\right] = \frac{1}{2}e^{-t} \quad (3)$$

$$Pr\left[v_q \leq -t \cdot \frac{1}{\epsilon}\right] = \frac{1}{2}e^{-t} \quad (4)$$

$$Pr\left[|v_q| \geq t \cdot \frac{1}{\epsilon}\right] = e^{-t} \quad (5)$$

We next employ these equations to bound the error of LM_{count}. We give the full proof of Proposition 4.2 in the full version [36].

Proposition 4.2. *Given a private database D, synthetic database D_s,* `COUNT` *query q, distance bound τ, and privacy budget ϵ. Interval $\mathcal{I} = (l, r) = (q(D_s)-\tau, q(D_s)+\tau)$ (1). LM_{count} satisfies the following:*

(1) If $q(D) \leq l$ but $o = 1$, then $err(\cdot) \leq \frac{1}{2}e^{-(l-q(D))\epsilon} - \frac{1}{2}e^{-(r-q(D))\epsilon}$.

(2) If $q(D) \geq r$ but $o = 1$, then $err(\cdot) \leq \frac{1}{2}e^{-(q(D)-r)\epsilon} - \frac{1}{2}e^{-(q(D)-l)\epsilon}$.

(3) If $l < q(D) < r$ but $o = 0$, then $err(\cdot) = \frac{1}{2}e^{-(q(D)-l)\epsilon} + \frac{1}{2}e^{-(r-q(D))\epsilon}$.

We now give an upper bound for the effectiveness threshold.

Proposition 4.3. *Given a private database D, synthetic database D_s,* `COUNT` *query q, privacy budget ϵ, and error probability $0 < \delta < 1$, the effectiveness threshold of the per-query decider LM_{count} at δ (Definition 3.4) has the following upper bound: $\tau_{min}^{LM_{count},\delta} \leq \frac{1}{\epsilon}\ln\frac{1}{2\delta}$.*

We give the full proof of Proposition 4.3 in the full version [36].

4.2 A Direct Solution Using the EM

In our second algorithm, EM_{count}, instead of plugging in a DP estimate for $q(D)$ to reach a decision about the distance bound, directly returns whether $o = 1$ ("Distance bound satisfied") or not.

4.2.1 A straightforward EM-based per-query decider. We begin by describing a straightforward approach we call EM_{count}^{naive} that directly instantiates the EM (Definition 2.10) with the output range $\mathcal{R} = \{0, 1\}$ and the score function $u(D, D_s, q, \tau, o)$ given by:

$$\text{If } q(D) \in \mathcal{I} \begin{cases} u(D, D_s, q, \tau, o = 0) = 0 \\ u(D, D_s, q, \tau, o = 1) = 1 \end{cases}$$

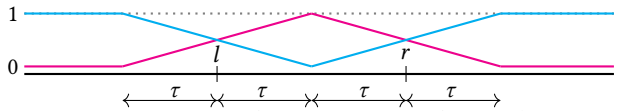

Figure 3: Score function u' for output $o = 0$ (in cyan) and $o = 1$ (in pink). Recall that $r - l = 2 \cdot \tau$.

$$\text{If } q(D) \notin \mathcal{I} \begin{cases} u(D, D_s, q, \tau, o = 0) = 1 \\ u(D, D_s, q, \tau, o = 1) = 0 \end{cases}$$

where $\mathcal{I} = (l, r) = (q(D_s) - \tau, q(D_s) + \tau)$ (from (1)). Intuitively, $u(\cdot)$ gives a non-zero score only to the correct outcome.

We give the full proof of Proposition 4.4 in the full version [36].

Proposition 4.4. *The global sensitivity of u is $\Delta u = 1$.*

The problem with EM_{count}^{naive} is that it suffers from high error as sensitivity 1 is too high. For instance, when $q(D) \in \mathcal{I}$ but $o = 0$, $Pr[o = 0] = c \cdot e^{\epsilon \times 0/2}$ and $Pr[o = 1] = c \cdot e^{\epsilon \times 1/2}$ (Definition 2.10), where c is a positive constant. Since there are only two outcomes

$$Pr[o = 0] = \frac{1}{1 + e^{\epsilon/2}} \quad (6)$$

$$Pr[o = 1] = \frac{e^{\epsilon/2}}{1 + e^{\epsilon/2}} \quad (7)$$

When $q(D) \in \mathcal{I}$, $o = 0$ is the wrong outcome and the error is given by (6). Similarly for $q(D) \notin \mathcal{I}$, error equals $1/(1 + e^{\epsilon/2})$.

In contrast, LM_{count} shows that lower error may be achieved. For example, when $q(D) = q(D_s)$ but $o = 0$ (case (3) in Proposition 4.2), LM_{count}'s error, $e^{-\tau\epsilon}$, is smaller than EM_{count}^{naive}'s error for values of ϵ and τ such as 0.1 and 8, respectively. We improve upon EM_{count}^{naive} by engineering a new score function with a smaller global sensitivity.

4.2.2 An improved EM-based per-query decider. To use the EM more reliably, we propose a new score function $u'(D, D_s, q, \tau, o)$:

$$\text{If } q(D) \notin (l - \tau, r + \tau) \begin{cases} u'(D, D_s, q, \tau, o = 0) = 1 \\ u'(D, D_s, q, \tau, o = 1) = 0 \end{cases} \quad (8)$$

$$\text{If } q(D) \in (l - \tau, q(D_s)] \begin{cases} u'(D, D_s, q, \tau, o = 0) = 1 - \frac{q(D)-(l-\tau)}{2\tau} \\ u'(D, D_s, q, \tau, o = 1) = \frac{q(D)-(l-\tau)}{2\tau} \end{cases} \quad (9)$$

$$\text{If } q(D) \in (q(D_s), r + \tau) \begin{cases} u'(D, D_s, q, \tau, o = 0) = \frac{q(D)-q(D_s)}{2\tau} \\ u'(D, D_s, q, \tau, o = 1) = 1 - \frac{q(D)-q(D_s)}{2\tau} \end{cases} \quad (10)$$

We illustrate u' in Figure 3. Note that u' outputs a value in the range $[0, 1]$ by definition, and not just in the set $\{0, 1\}$. It allows for a more gradual transition between scores of 0 and 1.

We give the full proof of Proposition 4.5 in the full version [36].

Proposition 4.5. *The global sensitivity of u' is $1/2\tau$.*

We refer to the algorithm that directly instantiates the EM (Definition 2.10) with $\mathcal{R} = \{0, 1\}$ and score function u' as EM_{count}. Since we post-process a DP estimate (Proposition 2.6), the following holds:

Observation 4.6. *EM_{count} satisfies ϵ-DP.*

We now give an upper bound for the effectiveness threshold (proof in the full version [36]).

Proposition 4.7. *Given a private database D, synthetic database D_s,* `COUNT` *query q, privacy budget ϵ, and error probability $0 < \delta < 1$, the effectiveness threshold of the per-query decider EM_{count} at δ (Definition 3.4) has the following upper bound: $\tau_{min}^{EM_{count},\delta} \leq \frac{1}{\epsilon}\ln\frac{1-\delta}{\delta}$.*

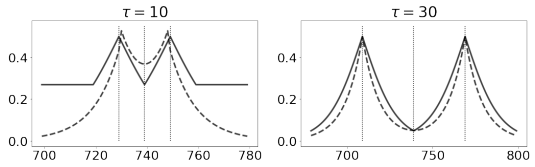

Figure 4: Error from LM_{count} (dashed) and EM_{count} (solid) as $q(D)$ is varied along the X axis, where D is the Adult dataset. The selection predicate is *relationship* `LIKE 'Unmarried' AND` *sex* `LIKE 'Male'`, $q(D_s) = 749, \epsilon = 0.1$ and $\tau = 10, 30$. The vertical lines mark the positions of $l, q(D_s)$, and r, respectively.

Comparison of LM_{count} and EM_{count}. Figure 4 depicts the error (along the Y axis) for fixed D_s, q, and ϵ but varying $q(D)$ values (along the X axis) when $\tau = 10$ (on the left) and $\tau = 30$ (on the right). We plot the error profile for LM_{count} following Proposition 4.2. Note that it gives upper bounds when $q(D) \leq l$ or $q(D) \geq r$. We plot the error profile for EM_{count} based on Definition 2.10 for the EM with score function u' ((8) to (10)). For $\tau = 10$ and $q(D) \in \mathcal{I}$, EM_{count}'s error is smaller. At $q(D) = q(D_s)$, LM_{count}'s error $e^{-\epsilon\tau}$ is larger than EM_{count}'s error $\frac{1}{1+e^{\epsilon\tau}}$ ($\epsilon\tau > 0$).

5 SOLUTIONS FOR SUM QUERY

We now discuss our solutions for SUM query on attribute A_i: (1) LM_{sum} (that instantiates Algorithm 1 with the Laplace Mechanism (LM)) in Section 5.1, (2) $R2T_{sum}$ (that instantiates Algorithm 1 with the *Race-to-the-Top* (R2T) mechanism [8]) in Section 5.2, and (3) SVT_{sum} (that directly solves the problem using the *Sparse Vector Technique* (SVT) [26]) in Section 5.3. We derive upper bounds for their effectiveness thresholds and then compare them.

Consider the SUM query q : `SELECT SUM(`A_i`) FROM` D WHERE φ, where φ denotes the predicate in the WHERE clause and is empty. Its global sensitivity Δq equals $\max dom(A_i)$ and is unbounded if the domain is unbounded. As done in previous work [3, 8, 23, 40], we assume a bound of GS_q on Δq and use it in the per-query deciders. Thus, we use GS_q as the global sensitivity in the analysis.

5.1 Laplace Mechanism-based Approach

Our algorithm LM_{sum} works similarly to LM_{count}. The only difference is that the scale of the Laplace distribution for noise ν_q is now GS_q/ϵ instead of $1/\epsilon$. LM_{sum} works by running `GenericDecider` (Algorithm 1) with the LM (Definition 2.7) as `DPNoisy`. Since we post-process a DP estimate (Proposition 2.6):

Observation 5.1. *LM_{sum} satisfies ϵ-DP.*

LM_{sum} suffers from the drawback that GS_q is often large in practice, resulting in high variance in ν_q. For example, A_i can represent incomes, distances, etc. and may contain large numbers. This can cause LM_{sum} to make mistakes with higher probability. We derive an upper bound for the effectiveness threshold below.

Proposition 5.2. *Given a private database D, synthetic database D_s, SUM query q with global sensitivity GS_q, privacy budget ϵ, and error probability $0 < \delta < 1$, the effectiveness threshold of the per-query decider LM_{sum} at δ (Definition 3.4) has the following upper bound: $\tau_{min}^{LM_{sum},\delta} \leq \frac{GS_q}{\epsilon} \ln \frac{1}{2\delta}$.*

Proof Sketch. The proof is the same as that of Proposition 4.3 (full proof given in the full version [36]). Since $\nu_q \sim Lap(GS_q/\epsilon)$, bounds (3)-(5) now use $\frac{GS_q}{\epsilon}$ than $\frac{1}{\epsilon}$. □

We now give an example to illustrate how $\tau_{min}^{LM_{sum},\delta}$ is computed. We will refer to this example again in Sections 5.2 and 5.3.

Example 5.3. *Let D be the database derived from IPUMS-CPS [17] with attributes age, sex, education and income-total, and D_s be its synthetic copy from an SDG. Consider the query q:* `SELECT SUM(`*income-total*`) FROM` D_s `WHERE` *age* ≤ 18. *Suppose $GS_q = 2M, DS_{q,D} = 9K$ and $\epsilon = 0.1$. For $\delta = 0.05$, we get $\tau_{min}^{LM_{sum},\delta} = \frac{GS_q}{\epsilon} \ln \frac{1}{2\delta} = 4.605 \times 10^7$.*

Observe that $\tau_{min}^{LM_{sum},\delta}$ is proportional to GS_q, so it is large in part due to GS_q being large. If GS_q was 2, then $\tau_{min}^{LM_{sum},0.05} = 46.052$. Thus, $\tau_{min}^{LM_{sum},\delta}$ highly depends on $dom(A_i)$ for sum, as expected.

5.2 R2T-based Approach

Our next algorithm, $R2T_{sum}$, uses a state-of-the-art DP mechanism for sum queries called *Race-to-the-Top* (R2T) [8] to obtain a DP estimate for $q(D)$. R2T constructs $\log(GS_q)$ number of queries (described below) with global sensitivities bounded by $2, 4, 8, \ldots, GS_q$. Then it computes a noisy estimate for each query using the LM (Definition 2.7) and returns the max of the largest noisy estimate and 0 (for non-negative integer-valued domain).

In $R2T_{sum}$, the j-th query (for $j = 1$ to $\log(GS_q)$) works by running the original SUM query q on a truncated database constructed by removing tuples in D with A_i value greater than $t_j = 2^j$. The output is denoted by $q(D, t_j)$. In the full version [36], we show that this choice of *truncation function* for constructing queries satisfies the required properties as stated in [8].

R2T [8] computes the noisy estimates for the constructed queries as follows (for a given parameter $0 < \beta < 1$, which corresponds to the confidence bound as equation (13) below will show):

$$\tilde{q}(D, t_j) = q(D, t_j) + Lap\left(\frac{t_j}{\epsilon/\log(GS_q)}\right) - \frac{t_j}{\epsilon/\log(GS_q)} \ln \frac{\log(GS_q)}{\beta} \quad (11)$$

where $t_j = 2^j$, $j = 1$ to $\log(GS_q)$. The final estimate for $q(D)$ is:

$$\tilde{q}(D) = \max\{\max_j \tilde{q}(D, t_j), q(D, 0)\} \quad (12)$$

In our setting, the downward local sensitivity $DS_{q,D}$ (Definition 2.4) is the largest A_i value from tuples in D that satisfy the WHERE condition φ in q, because removing the matching tuple gives the worst-case change for $D \approx D', D' \subseteq D$. The following result from [8] holds (discussed further in the full version [36])

$$Pr\left[q(D) \geq \tilde{q}(D) \geq q(D) - 4\log(GS_q) \ln\left(\frac{\log(GS_q)}{\beta}\right)\frac{DS_{q,D}}{\epsilon}\right] \geq 1 - \beta \quad (13)$$

In other words, the probability that the noisy estimate $\tilde{q}(D)$ for $q(D)$ in $R2T_{sum}$ lies in the range $[q(D) - 4\log(GS_q) \ln\left(\frac{\log(GS_q)}{\beta}\right) \cdot \frac{DS_{q,D}}{\epsilon}, q(D)]$ is at least $1 - \beta$.

To summarize, $R2T_{sum}$ works by running `GenericDecider` (Algorithm 1) with R2T [8] (and the aforementioned truncation function) as `DPNoisy`, with parameters GS_q and $0 < \beta < 1$ for confidence bound in (13). Since we post-process a DP estimate (Proposition 2.6), the following holds:

Observation 5.4. *$R2T_{sum}$ satisfies ϵ-DP.*

We give an upper bound on the effectiveness threshold of $R2T_{sum}$ using (13) (proof in the full version [36]).

Proposition 5.5. *Given a private database D, synthetic database D_s, SUM query q with global sensitivity GS_q and downward local sensitivity $DS_{q,D}$, privacy budget ϵ, and error probability $0 < \delta < 1$, the effectiveness threshold of the per-query decider $R2T_{sum}$ at δ (Definition 3.4) has the following upper bound: $\tau_{min}^{R2T_{sum},\delta} \leq 4\log(GS_q)\ln\left(\frac{\log(GS_q)}{\delta}\right)\frac{DS_{q,D}}{\epsilon}$.*

We now illustrate how to compute $\tau_{min}^{R2T_{sum},\delta}$ for Example 5.3.

Example 5.6. *Recall the setting in Example 5.3. As a result,*

$$\tau_{min}^{R2T_{sum},\delta} = 4\log(GS_q)\ln\left(\frac{\log(GS_q)}{\delta}\right)\frac{DS_{q,D}}{\epsilon} = 4.549\times 10^7$$

which is less than $\tau_{min}^{LM_{sum},\delta} = 4.605\times 10^7$.

Observe that $\tau_{min}^{R2T_{sum},\delta}$ is proportional to $\log(GS_q)\log\log(GS_q)\cdot DS_{q,D}$, so it is large in part due to GS_q being large. In Example 5.6, $\tau_{min}^{R2T_{sum},\delta} < \tau_{min}^{LM_{sum},\delta}$ due to the large gap in GS_q and $DS_{q,D}$. In our experiments (Section 7), we empirically show how the errors for SUM queries vary in datasets with larger and smaller GS_q values.

5.3 SVT-based Approach

The *Sparse Vector Technique* (SVT) [14, 26] is a DP mechanism to report whether the output of a query q on D exceeds its specified threshold. It compares noisy versions of $q(D)$ and the threshold, and gives a yes ($\top$ = noisy query answer exceeded noisy threshold) or no ($\bot$ = noisy query answer did not exceed noisy threshold) answer. An advantage of using SVT is that it consumes privacy budget only if output = $\top$. The SVT algorithm is applied on a sequence of queries, each with its own threshold, resulting in an output given by $\{\top,\bot\}^l$, where l is the number of queries answered. The balance on the privacy budget degrades with the number of queries with output = $\top$ until the privacy budget runs out [26] and SVT stops.

We denote the per-query decider for SUM query q that runs SVT on a sequence of sum queries (described below) as SVT_{sum} (Algorithm 2). Given privacy budget ϵ, SVT_{sum} begins by sampling a noise term ρ distributed as $Lap(\frac{1}{\epsilon/2})$ (line 4) and uses it to get the noisy thresholds for all queries. The remaining $\epsilon/2$ privacy budget is used towards the first $\top$ in the output, after which SVT_{sum} stops. We modify the queries from $R2T_{sum}$ (Section 5.2): $q_j(D) = q(D,t_j)/t_j$, for $t_j = 2^j$, $j = 1$ to $\log(GS_q)$. We divide by t_j so that $\forall j, \Delta q_j = 1$ in the worst-case (i.e. the max over the sensitivities of the input queries now does not exceed 1), allowing the noise scales to be proportional to 1 than GS_q. The relevant threshold values are l and r, which must also be divided by t_j. First, SVT is run on queries $q_j(D)$ with threshold $T_j = r/t_j$ (loop in line 5), adding independent Laplace noise distributed as $Lap(\frac{1}{\epsilon/2})$ to each $q_j(D)$. If any noisy $q_j(D)$ value results in $\top$, then $o = 0$ is returned (line 9). If not, only $\epsilon/2$ privacy budget has been consumed by ρ and SVT is run on queries $q_j(D)$ with threshold $T_j = (l+1)/t_j$ (loop in line 10), again adding independent Laplace noise distributed as $Lap(\frac{1}{\epsilon/2})$ to each $q_j(D)$. If any noisy $q_j(D)$ value results in $\top$, then $o = 1$ is returned (line 14). Otherwise, $o = 0$ is returned at the end (line 15).

Definition 5.7. *In the context of SVT [26], the sequence of queries used is called monotonic if, in going from D to a neighboring database D', all query answers that are different change in the same direction, i.e., they all increase or they all decrease.*

Observe that $q_j(D)$s are monotonic because $\forall j, q(D,t_j) \geq q(D', t_j)$ or $\forall j, q(D,t_j) \leq q(D',t_j)$. This helps save a factor of 2 in the noise scale for ν_j (discussed further in the full version [36]). Unlike the previous per-query deciders that post-process a DP estimate, here we need to show that SVT_{sum} preserves DP.

Algorithm 2: Per-query decider SVT_{sum} for SUM

Input : q - SUM query, D - private database, D_s - synthetic database, τ - distance bound, ϵ - privacy budget

Output : $o = 1$ if the desired distance bound from $q(D_s)$ is satisfied for $q(D)$, else $o = 0$.

1 **Function** $SVT_{sum}(q, D, D_s, \tau, \epsilon)$**:**
2 $A_i \leftarrow$ aggregate attribute in q;
3 $l \leftarrow q(D_s) - \tau, r \leftarrow q(D_s) + \tau$;
4 $\rho \leftarrow Lap(\frac{1}{\epsilon/2})$;
5 **for** $j \in \{1, 2, 3, \ldots, \lceil\log(GS_q)\rceil\}$ **do**
6 $t_j \leftarrow 2^j, q(D,t_j) \leftarrow$ SELECT SUM(A_i) FROM D WHERE φ AND $A_i \leq t_j$;
7 $q_j(D) \leftarrow q(D,t_j)/t_j, \nu_j \leftarrow Lap(\frac{1}{\epsilon/2})$;
8 **if** $q_j(D) + \nu_j \geq r/t_j + \rho$ **then**
9 **return** $o = 0$ ("Distance bound unmet");
10 **for** $j \in \{1, 2, 3, \ldots, \lceil\log(GS_q)\rceil\}$ **do**
11 $t_j \leftarrow 2^j, q(D,t_j) \leftarrow$ SELECT SUM(A_i) FROM D WHERE φ AND $A_i \leq t_j$;
12 $q_j(D) \leftarrow q(D,t_j)/t_j, \nu_j \leftarrow Lap(\frac{1}{\epsilon/2})$;
13 **if** $q_j(D) + \nu_j \geq (l+1)/t_j + \rho$ **then**
14 **return** $o = 1$ ("Distance bound satisfied");
15 **return** $o = 0$ ("Distance bound unmet");

Theorem 5.8. *SVT_{sum} is ϵ-DP.*

Proof Sketch. Let $\top_1$ and $\top_2$ denote the 'yes' answers from SVT, i.e., when the noisy query answer exceeds the noisy threshold in lines 9 and 14, respectively. Let $\bot_1$ and $\bot_2$, respectively, denote when these checks fail. Since the algorithm stops as soon as a $\top_1$ or $\top_2$ is returned in lines 9 or 14, the output string a is either of the form $\bot_1,\ldots,\bot_1,\top_1$ or $\bot_1,\ldots,\bot_1,\bot_2,\ldots,\bot_2,\top_2$. For both forms, we show in the full version that $Pr[SVT_{sum}(q,D,D_s,\tau,\epsilon) = a] \leq e^{\epsilon}Pr[SVT_{sum}(q,D',D_s,\tau,\epsilon) = a]$ for $D \approx D'$ holds adapting ideas from [26]. Intuitively, SVT_{sum} is equivalent to running SVT once, with the first half mapped to $o = 0$ and the remaining half mapped to $o = 1$. A complete proof is given in the full version [36]. □

SVT_{sum} incurs high error if $q(D)$ and r (or l) are close because in the later iterations where t_j values are large, the check is easily influenced by noise. Note that $\forall t_k \geq DS_{q,D}, q(D,t_k) = q(D)$, where $DS_{q,D}$ is the downward local sensitivity (Definition 2.4). We present

an optimization in Algorithm 4 in the full version [36] aimed at improving the accuracy of SVT_{sum} by obtaining a private bound for $DS_{q,D}$ to be used as the largest truncation threshold. In the rest of the paper, by SVT_{sum} we will refer to the improved algorithm using the bound from Algorithm 4. We analyze the error of SVT_{sum} in the full version [36].

Comparison of LM_{sum}, $R2T_{sum}$ and SVT_{sum}. As demonstrated by Examples 5.3 and 5.6, when the difference between $DS_{q,D}$ and GS_q is large, the effectiveness threshold for $R2T_{sum}$ is likely to be smaller than that of LM_{sum}. We show that SVT_{sum} can achieve smaller error than $R2T_{sum}$ in the experiments (Section 7).

6 SOLUTIONS FOR MEDIAN QUERY

We present two solutions and analyze their errors for MEDIAN query q on attribute A_i: SELECT MEDIAN(A_i) FROM D WHERE φ. (1) EM_{med} (that instantiates Algorithm 1 with the Exponential Mechanism (EM)) in Section 6.1, and (2) $Hist_{med}$ (that directly solves the problem using a noisy histogram) in Section 6.2. The true output of the median query $q(D)$ is the $\lceil \frac{n'}{2} \rceil$-th element in the sorted list of A_i values among tuples that satisfy the WHERE clause, where n' is the (private) number of tuples in D satisfying φ.

6.1 Exponential Mechanism-based approach

Let $rank_\varphi(D, e)$ be the output of the query: SELECT COUNT(*) FROM D WHERE φ AND $A_i < e$. Our approach EM_{med} uses the algorithm from [6] that computes a noisy estimate for $q(D)$. EM_{med} runs GenericDecider with the EM as DPNoisy, with additional parameters $\mathcal{R} = dom(A_i)$ and score function $u(D, e) = -|rank_\varphi(D, e) - \frac{n'}{2}|$, $\forall e \in \mathcal{R}$. The sensitivity of the score function equals 1 because rank of any e either stays the same or changes in the same direction as n' between databases $D \approx D'$. Since we post-process a DP estimate (Proposition 2.6), the following observation holds:

Observation 6.1. *EM_{med} satisfies ϵ-DP.*

We analyze the error of EM_{med} in the full version [36].

6.2 Histogram-Based Algorithm

We next propose a histogram-based approach called $Hist_{med}$ (Algorithm 3), which uses the intuition that if at least half the values in A_i (from tuples satisfying φ) either are less than or equal to l, or are greater than or equal to r, then $q(D) \notin \mathcal{I}$. These bounds $l = q(D_s) - \tau$ and $r = q(D_s) + \tau$ are compared with a DP estimate, say m, for $\lceil \frac{n'}{2} \rceil$ obtained using the LM (line 3, where n' is the number of tuples in D satisfying φ). If neither count exceeds m, then $o = 1$ (line 13). Otherwise, $o = 0$ (lines 8 and 10).

We next show that $Hist_{med}$ is ϵ-DP.

Proposition 6.2. *$Hist_{med}$ satisfies ϵ-DP.*

Proof Sketch. We spend $\epsilon/2$ to obtain an estimate for n', and the remaining $\epsilon/2$ on $q_1(D)$ and $q_2(D)$ (lines 5-6) that use disjoint sets of tuples ($\tau > 0$). Hence $Hist_{med}$ satisfies ϵ-DP by sequential and parallel composition, and post-processing (Proposition 2.6). □

Comparison of EM_{med} and $Hist_{med}$. Suppose q is a query that computes the median on attribute age, $q(D) = 37, \epsilon = 0.1, \tau = 5$,

Algorithm 3: Per-query decider $Hist_{med}$ for MEDIAN

Input : q - MEDIAN query, D - private database, D_s - synthetic database, τ - distance bound, ϵ - privacy budget for error analysis

Output : Whether the distance bound is met for q

1 **Function** $Hist_{med}(q, D, D_s, \tau, \epsilon)$**:**
2 $n' \leftarrow$ number of tuples in D that satisfy φ in q;
3 $\nu_q \leftarrow Lap(\frac{1}{\epsilon/2}), \tilde{n} \leftarrow n' + \nu_q$;
4 $A_i \leftarrow$ attribute for median used in q;
5 $q_1(D) \leftarrow$ SELECT COUNT(*) FROM D WHERE φ AND $A_i \le q(D_s) - \tau$;
6 $q_2(D) \leftarrow$ SELECT COUNT(*) FROM D WHERE φ AND $A_i \ge q(D_s) + \tau$;
7 $\nu_{q_1}, \nu_{q_2} \leftarrow Lap(\frac{1}{\epsilon/2})$;
8 **if** $q_1(D) + \nu_{q_1} \ge \lceil \tilde{n}/2 \rceil$ **then**
9 **return** $o = 0$ ("Distance bound unmet");
10 **else if** $q_2(D) + \nu_{q_2} \ge \lceil \tilde{n}/2 \rceil$ **then**
11 **return** $o = 0$ ("Distance bound unmet");
12 **else**
13 **return** $o = 1$ ("Distance bound satisfied");

and $\mathcal{I} = (28, 38)$ (as defined in (1)). EM_{med} returns $e = 38$ with probability equal to 0.9995. $38 \notin \mathcal{I}$ but $q(D) \in \mathcal{I}$. $Hist_{med}$ can be the better choice (discussed further in the full version [36]).

Extending the framework to other aggregates. Our solutions can be used to support some other aggregates, e.g., average can be expressed as the output of a SUM query divided by the output of a COUNT query, each consuming some ϵ. The solutions for MEDIAN can be generalized to work for other quantiles. For example, to compute the first quartile, we change the score function in EM_{med} to use $n'/4$ instead of $n'/2$, and change $\lceil \tilde{n}/2 \rceil$ to $\lceil \tilde{n}/4 \rceil$ and $\lceil 3\tilde{n}/4 \rceil$ in lines 8 and 10 (Algorithm 3), respectively. However, supporting more complex aggregates needs careful analysis to establish bounds.

7 EXPERIMENTS

In this section, we analyze the accuracy and efficiency of our proposed per-query deciders for COUNT, SUM and MEDIAN queries with the following questions:

(1) How is the accuracy of each proposed solution affected when τ and ϵ are varied separately?
(2) For each proposed solution, what type of queries benefit most in terms of accuracy?
(3) How does the performance of the specialized solutions compare with that of the solutions bases on Algorithm 1?

We have implemented the per-query deciders in Python 3.8.8 using Pandas [33] and NumPy [22] libraries. All experiments were run on Apple M1 CPU @3.2 GHz with 16 GB of RAM.

7.1 Experimental Setup

We describe the datasets, queries, error measures, and parameters.

Dataset. We consider two datasets as the **private database** D.

(1) The first dataset is derived from the **IPUMS-CPS survey data** [17], an individual-level population database, for the years 2011-2019 with 1,340,703 tuples and 10 attributes: *relate*, *age*, *sex*, *race*, *marst*, *citizen*, *workly*, *classwkr*, *educ* and *inctot*. The only numerical attributes are *age* and *inctot* with domains $\{0, 1, \ldots, 80, 85\}$ and $\{0, 1, \ldots, 999999999\}$, respectively. We only include tuples with *inctot* value less than or equal to 500K. The domain sizes of the categorical attributes vary from 3 to 36. (2) The second dataset is derived from the **NYC Yellow Taxi Trip data** [1] for January 2022 with 2,177,719 tuples and 10 attributes: *vendorID*, *passenger_count*, *trip_distance*, *rateCodeID*, *store_and_fwd_flag*, *payment_type*, *fare_amount*, *tip_amount*, *total_amount* and *congestion_surcharge* (with some pre-processing as discussed in [36]). The domain sizes of the categorical attributes vary from 2 to 6.

We generate a **synthetic database D_s for** D using PrivBayes [43], a Bayesian network based DP-SDG. Nodes and edges in the network represent attributes in D and conditional independence relations between attributes in D. PrivBayes first learns a differentially private Bayesian network $\hat{\mathcal{N}}$ and then uses it to derive a factored form of the joint tuple probabilities based on the noisy conditional probabilities. Note that $\hat{\mathcal{N}}$ can make incorrect conditional independence assumptions between attributes in D.

Queries. We refer to the Summary File 1 (SF-1) [38] released by the U.S. Census Bureau to construct queries for the IPUMS-CPS dataset. We analyze 12 `COUNT`, 9 `SUM` (on *inctot*) and 9 `MEDIAN` (on *age*) queries. For the second dataset, we analyze 10 `COUNT`, 10 `SUM` (on *total_amount*), and 10 `MEDIAN` (on *trip_distance*) queries. Due to space constraints, here we present results on 4 representative queries of each aggregate on the IPUMS-CPS data, and 2 `SUM` queries on the NYC Taxi Trip data with $DS_{q,D}$ much smaller than GS_q (Table 2), while the full list of queries and results for the other queries on both datasets are shown in the full version [36].

Error measure. We measure the error of per-query deciders as follows. We run each of our per-query deciders 100 times to decide whether $q(D)$ lies in $(q(D_s) \cdot (1-\tau), q(D_s) \cdot (1+\tau))$, where τ is a percentage of the query answer on the synthetic data, $q(D_s)$. We measure error as the fraction of times the algorithm makes an error in determining whether $q(D)$ lies in $(q(D_s) \cdot (1-\tau), q(D_s) \cdot (1+\tau))$.

Parameter settings. We set $\beta = 0.05$ in $R2T_{sum}$ (Section 5.2) and $\theta = 0.95$ in Algorithm 4 (Section 5.3). Default $\epsilon = 0.25$ and $\tau = 3.2\%$ of $q(D_s)$. In our experiments, we vary $\tau = 0.2\%, 0.8\%, 3.2\%, 12.8\%, 51.2\%$, and vary $\epsilon = 0.0625, 0.125, 0.25, 0.5, 1$.

7.2 Accuracy and Performance Analysis

7.2.1 Accuracy analysis. We present our analysis of the impact on accuracy as ϵ and τ are varied individually. We also investigate which queries benefit the most for each per-query decider. In the following discussion, we use $\mathcal{I} = (q(D_s) \cdot (1-\tau), q(D_s) \cdot (1+\tau))$, where τ is a percentage of $q(D_s)$.

`COUNT` queries. We present our analysis for 4 queries: q_1, q_3, q_5 and q_{12} (Figures 5-6). Consider the setting where τ varies. $q_1(D)$ equals $q_1(D_s)$ and error from both LM_{count} and EM_{count} decreases when τ increases, as expected. EM_{count} gives smaller error. For $\tau \leq 3.2\%, q_3(D) \notin \mathcal{I}$. At $\tau = 3.2\%$, $q_3(D)$ is closest to one of $\mathcal{I}$'s endpoints, so we see the error from LM_{count} increase (before it goes to 0) because there is a higher probability of $q_3(D)$'s noisy estimate being in $\mathcal{I}$. Similarly for EM_{count} on q_3, where error starts decreasing for $\tau > 3.2\%$. EM_{count}'s error on q_3 for $\tau \leq 3.2\%$ ranges from 0.39 to 0.45. For $\tau \leq 0.8\%, q_5(D) \notin \mathcal{I}$. At $\tau = 3.2\%$, $q_5(D) \in \mathcal{I}$ and is closest to one of $\mathcal{I}$'s endpoints, so LM_{count}'s error has the same trend as that for q_3. In EM_{count} for q_5 and $\tau = 0.2\%$, $c \cdot \exp(\epsilon u'(\cdot)\tau)$ (Definition 2.10) for $o = 0$ and $o = 1$ are closest, so $Pr[o = 1] = 0.27$ (after normalization). EM_{count}'s error increases at $\tau = 3.2\%$ (before it goes to 0) because $q_5(D)$ is close to an endpoint of $\mathcal{I}$. LM_{count} and EM_{count} give 0 error for q_{12} because the resulting τ values are such that $q(D)$ stays far from $\mathcal{I}$'s endpoints.

When ϵ is varied instead, the errors from both LM_{count} and EM_{count} have a decreasing trend except for q_3 and q_{12}. As ϵ increases, the DP computations become less noisy. LM_{count}'s error on q_3 increases initially because probability of $q_3(D)$'s noisy estimate being in $\mathcal{I}$ increases despite the noise scale decreasing (before error goes to 0). Similarly for EM_{count} on q_3. EM_{count}'s error on q_{12} increases for $\epsilon > 0.5$ because $c \cdot \exp(\epsilon u'(\cdot)\tau)$ (Definition 2.10) for $o = 0$ and $o = 1$ is greater than $c \cdot \exp(7527 \cdot u'(\cdot))$, represented as infinity in Python. Either outcome is equally likely to be returned.

Table 2: Queries used in experiments. Blocks 1, 2, 3 are for the IPUMS-CPS data, block 4 is for the NYC Taxi Trip data.

Query	WHERE clause	$q(D), q(D_s)$
1. COUNT	q_1: *sex* LIKE 'Female' AND *race* LIKE 'White-American Indian-Asian' AND *workly* LIKE 'Yes'	34 34
	q_3: *sex* LIKE 'Male' AND *educ* LIKE 'Doctorate degree' AND *marst* LIKE 'Separated'	87 91
	q_5: *sex* LIKE 'Female' AND *educ* LIKE 'Grades 5 or 6' AND *marst* LIKE 'Never married/single'	1560 1606
	q_{12}: *race* LIKE 'White' AND *marst* LIKE 'Married, spouse present' AND *citizen* LIKE 'Born in U.S'	471994 470483
2. SUM on *inctot*	q_{13}: *sex* LIKE 'Female' AND *race* LIKE 'White-Black' AND *workly* LIKE 'No'	6915340 6942866
	q_{16}: *race* LIKE 'White' AND *marst* LIKE 'Divorced' AND *citizen* LIKE 'Not a citizen'	123543040 128497757
	q_{20}: *sex* LIKE 'Female' AND *educ* LIKE 'High school diploma or equivalent' AND *marst* LIKE 'Never married/single'	685635093 690711885
	q_{21}: *race* LIKE 'White' AND *marst* LIKE 'Married, spouse present' AND *citizen* LIKE 'Born in U.S'	23542765109 23434676868
3. MEDIAN on *age*	q_{22}: *workly* LIKE 'Yes' AND *classwkr* LIKE 'Wage/salary, private' AND *educ* LIKE 'Bachelor's degree'	41 41
	q_{23}: *sex* LIKE 'Male' AND *race* LIKE 'White-Black' AND *relate* LIKE 'Spouse'	40 40
	q_{28}: *race* LIKE 'Asian only' AND *marst* LIKE 'Separated' AND *citizen* LIKE 'Born in U.S'	39 39
	q_{29}: *sex* LIKE 'Male' AND *race* LIKE 'White' AND *classwkr* LIKE 'Wage/salary, private'	40 40
4. SUM on *total_amount*	q_{45}: *passenger_count* < 2 AND *trip_distance* $= 8$ AND *tip_amount* ≤ 2	218047 213685
	q_{50}: *trip_distance* ≤ 1 AND *fare_amount* ≤ 6 AND *congestion_surcharge* $= 2$	5680776 5624898

`SUM` queries. We present our analysis for 4 queries: q_{13}, q_{16}, q_{20} and q_{21} (Figures 7-8). $DS_{q,D}$ values are: 127764, 278011, 403353, 500000. Consider the setting where τ varies. $q_{13}(D) \notin \mathcal{I}$ (as defined in (1)) at $\tau = 0.2\%$ and the probability of LM_{sum}'s noisy estimate being outside $\mathcal{I}$ is high, so the error is low. LM_{sum}'s error shoots up at $\tau = 0.8\%$ because $q_{13}(D)$ is now in $\mathcal{I}$, but the probability that $q_{13}(D)$'s noisy estimate is outside $\mathcal{I}$ is high. Error decreases as τ increases further. Similarly for q_{16} and q_{20}. LM_{sum} incurs an error of 0 for q_{21} in all cases because $q_{21}(D)$ is large and for the chosen τ values, it is far from $\mathcal{I}$'s endpoints in comparison to the noise scale. $R2T_{sum}$ gives error close to 1 whenever $q_{13}(D) \in \mathcal{I}$ because the noise in its estimate for $q_{13}(D)$ is large and this estimate falls outside

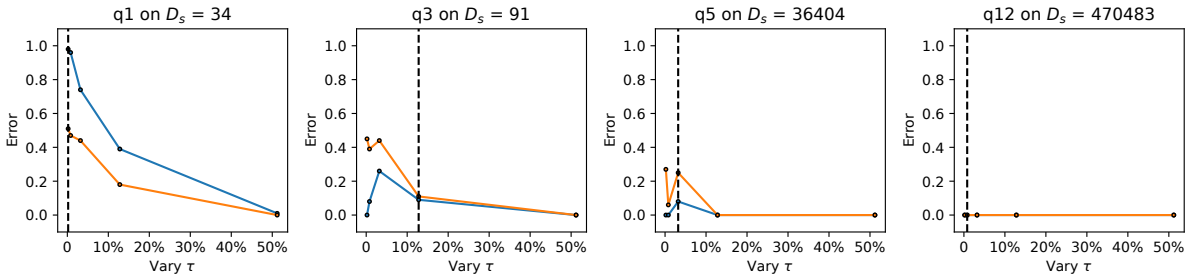

Figure 5: IPUMS-CPS data: Error for `COUNT` queries from LM_{count} (in blue) and EM_{count} (in orange) as τ varies. The dotted line marks the smallest τ value considered such that the query answer on the private data belongs in the interval $\mathcal{I}$.

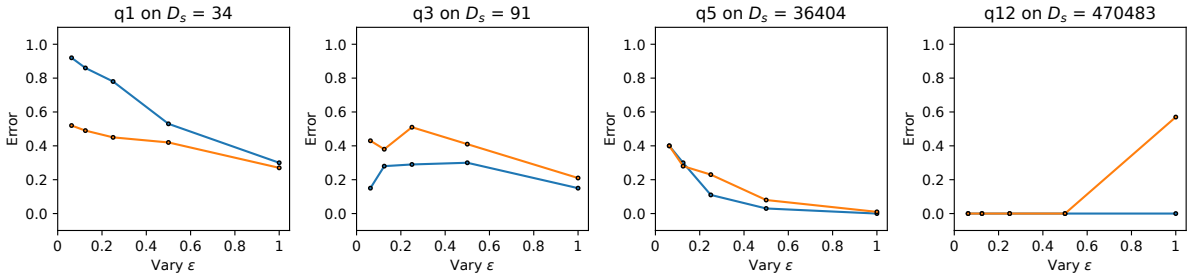

Figure 6: IPUMS-CPS data: Error for `COUNT` queries from LM_{count} (in blue) and EM_{count} (in orange) as ϵ varies.

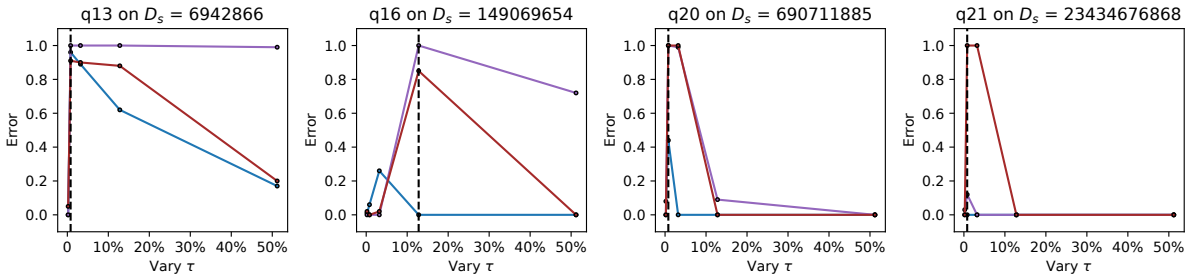

Figure 7: IPUMS-CPS data: Error for `SUM` queries from LM_{sum} (in blue), $R2T_{sum}$ (in purple) and SVT_{sum} (in brown) as τ varies. The dotted line marks the smallest τ value considered such that the query answer on the private data belongs in the interval $\mathcal{I}$.

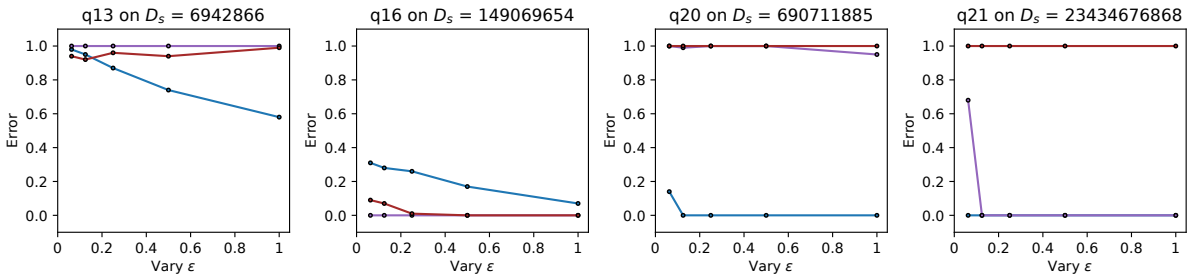

Figure 8: IPUMS-CPS data: Error for `SUM` queries from LM_{sum} (in blue), $R2T_{sum}$ (in purple) and SVT_{sum} (in brown) as ϵ varies.

$\mathcal{I}$. Similarly for q_{16}. $R2T_{sum}$'s error on q_{20} is high when $q_{20}(D) \in \mathcal{I}$ but τ is small compared to the noise in $R2T_{sum}$'s estimate for $q_{20}(D)$. As τ increases, the error decreases. $R2T_{sum}$'s error on q_{21} follows the same trend as that of q_{20}, except the error stays low because $q_{21}(D_s)$ is large and consequently τ values are large. SVT_{sum}'s error on q_{13} is low at $\tau = 0.2\%$ because $q_{13}(D) < l$ and the chance of any noisy truncated sum query exceeding its noisy threshold is low (lines 8 and 13 in Algorithm 2). For larger τ, $q_{13}(D) \in \mathcal{I}$ and

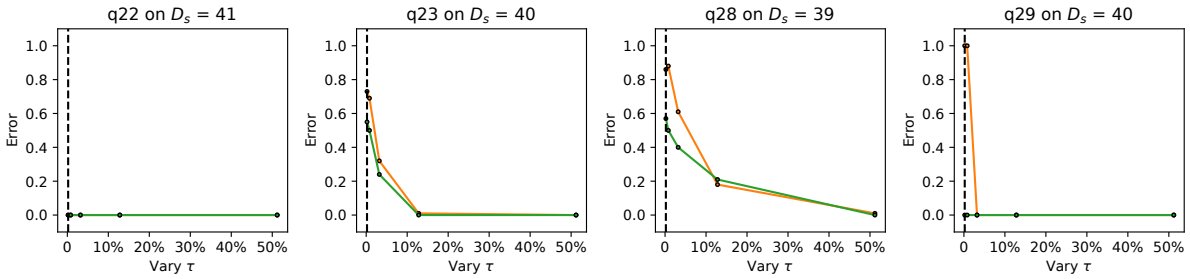

Figure 9: IPUMS-CPS data: Error for `MEDIAN` queries from EM_{med} (in orange) and $Hist_{med}$ (in green) as τ varies. The dotted line marks the smallest τ value considered such that the query answer on the private data belongs in the interval $\mathcal{I}$.

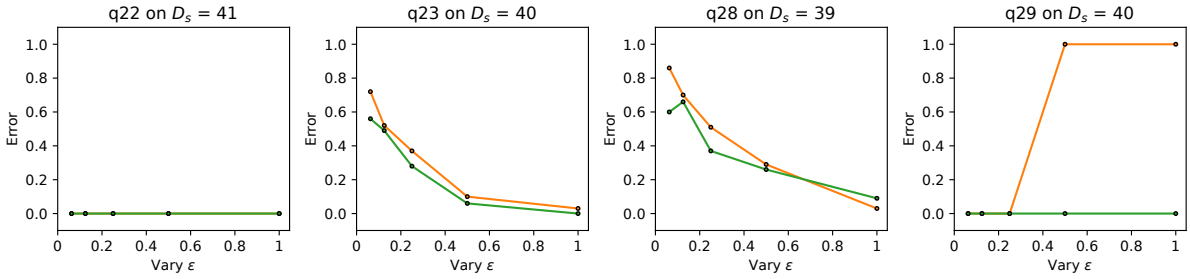

Figure 10: IPUMS-CPS data: Error for `MEDIAN` queries from EM_{med} (in orange) and $Hist_{med}$ (in green) as ϵ varies.

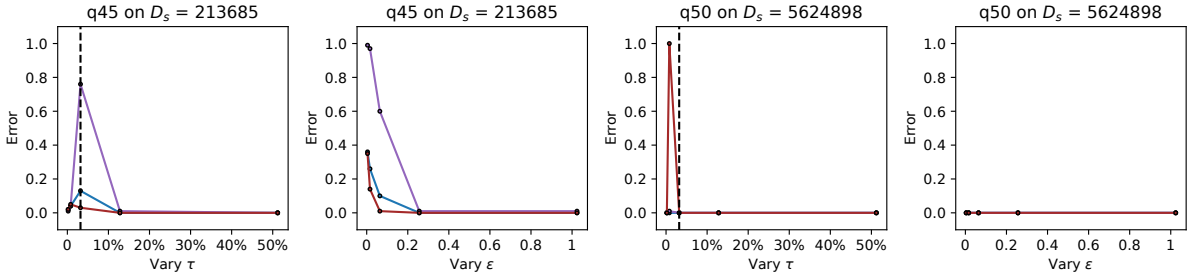

Figure 11: NYC Taxi data: Error for `SUM` queries from LM_{sum} (in blue), $R2T_{sum}$ (in purple) and SVT_{sum} (in brown) as ϵ and τ vary separately. The dotted line marks the smallest τ value considered such that the query answer on the private data belongs in $\mathcal{I}$.

moves away from $\mathcal{I}$'s endpoints, which increases the chance of the threshold in line 8 not being exceeded and in line 13 being exceeded. Similarly for q_{16}, q_{20} and q_{21}, where the last two have large outputs on D_s and so error decreases sooner.

When ϵ is varied, LM_{sum}'s error on the 4 queries shows a decreasing trend because the noise scale decreases as ϵ increases. $q_{13}(D)$, $q_{20}(D)$ and $q_{21}(D)$ are in the associated $\mathcal{I}$ at (default) $\tau = 3.2\%$. For the first two, increasing ϵ does not change $R2T_{sum}$'s error because the noise is $O(\log(GS_q)\log\log(GS_q)\cdot DS_{q,D})$ (from (13)), keeping $\tilde{q}(D)$ outside $\mathcal{I}$ with high probability. Since $q_{21}(D_s)$ is the largest, $R2T_{sum}$'s error decreases because the noise in $\tilde{q}(D)$ decreases as ϵ increases, while $\mathcal{I}$'s width stays the same. For queries with answers on D in $\mathcal{I}$, SVT_{sum}'s error does not change much as ϵ increases. $q_{13}(D)$ and $q_{20}(D)$ are close to the left-endpoints of the respective intervals, and so SVT_{sum}'s error is high (line 13 in Algorithm 2). In contrast, SVT_{sum}'s error on q_{16} decreases as ϵ increases because the noise scale decreases and $q_{16}(D) < l$.

Next, we analyze the impact of larger gaps in $DS_{q,D}$ and GS_q, for small GS_q of 377 (Figure 11). When τ varies, LM_{sum} and $R2T_{sum}$ give highest error for q_{45} at $\tau = 3.2\%$ as $q_{45}(D) \in \mathcal{I}$ (closer to r), but the noise in their estimates for $q_{45}(D)$ is large while interval width is small, causing the estimates to not be in $\mathcal{I}$. At $\tau = 0.8\%$, $q_{50}(D) \notin \mathcal{I}$ (closer to r). LM_{sum} and $R2T_{sum}$ give low error as $\mathcal{I}$'s width is large and their estimates for $q_{50}(D)$ are in $\mathcal{I}$ with high probability. SVT_{sum} gives low error for both queries, except for q_{50} at $\tau = 0.8\%$ as the truncation threshold from Algorithm 4 is often smaller than $DS_{q,D}$, causing the check in line 8 to incorrectly fail.

At default $\tau = 3.2\%$, $q_{45}(D)$ and $q_{50}(D)$ are in their associated intervals (closer to r). For q_{45}, the interval is small, so all 3 solutions give error at least 0.35 when ϵ is small, with $R2T_{sum}$ giving the highest error due to large noise in its estimate. For q_{50}, the interval width is larger, so all 3 solutions give 0 error.

`MEDIAN` queries. We present our analysis for 4 queries: q_{22}, q_{23}, q_{28} and q_{29} (Figures 9-10). Consider the setting where τ varies. Let

us first look at EM_{med}. For q_{22} and q_{29}, the support, i.e., tuples in D that satisfy the predicates in the WHERE clause, is large. As a result, the probability distribution used to sample the noisy estimate for the median is more concentrated around the correct value. The opposite is true for q_{23} and q_{28}. We see error decrease as τ increases. $Hist_{med}$'s error has a decreasing trend in all cases because the noisy bin counts are far from the query's noisy estimate for $\lceil n'/2 \rceil$.

When ϵ increases, the variance of the probability distribution used to sample the noisy estimate in EM_{med} decreases. EM_{med}'s error has a decreasing trend except for q_{29} because the scores are negatives numbers with large magnitudes because the support is large (Definition 2.10). As discussed above for $Hist_{med}$, the noisy bin counts are far from the query's noisy estimate for majority. As ϵ increases, the error decreases because the noise scale decreases.

7.2.2 Performance analysis. We compare average runtimes of per-query deciders for COUNT, SUM and MEDIAN queries in Table 3. The reported average is per query and per 1 run (out of the 100 trials).

7.2.3 Discussion. We now summarize our findings. The experiments suggest the following comparative trends (Table 4). For COUNT, when $q(D) = q(D_s)$, EM_{count}'s error is less than LM_{count}'s error for different ϵ and τ values. When $q(D) \in \mathcal{I}$ and $q(D) \notin \mathcal{I}$, LM_{count} is superior. For SUM, when $q(D) \in \mathcal{I}$ and τ varies, LM_{sum} was the best choice followed by SVT_{sum}. Note that $DS_{q,D}$ plays an important role here. When $q(D) \notin \mathcal{I}$, $R2T_{sum}$ and SVT_{sum} were the better choices. For MEDIAN, $Hist_{med}$ generally gives smaller error than EM_{med} no matter the relationship between $q(D)$ and $\mathcal{I}$.

There is not a clear pattern for when the error is high except for the condition when $q(D)$ is close to one of the endpoints of the interval, or when the downward local sensitivity is not far from the global sensitivity for the given SUM query.

Typically, we expect error of DP algorithms to decrease with higher ϵ. But, we see the reverse in some experiments (see LM_{count} for q_3, EM_{count} for q_{12}, SVT_{sum} and $R2T_{sum}$ for queries with answer on D in the interval, and EM_{med} for q_{29}). This is because the error function for the per-query deciders is not monotonic in ϵ. This may suggest that there exist smarter ways to design algorithms that only use a portion of the overall budget available to get better accuracy.

LM_{count} and EM_{count} were not effective for queries at smaller τ, except when $q(D_s)$ and τ are large, in which case we do not know if either approach is effective. They were effective for the same settings. Similarly for LM_{sum}. $R2T_{sum}$ and SVT_{sum} were not effective for queries with large answer on D at small τ. In general, the accuracy can be improved with a larger privacy budget ϵ, or a larger distance bound τ, which may not always be feasible.

8 RELATED WORK

We have used several existing DP mechanisms for count, sum, and median from the literature [6, 8, 14, 26] for DP-PQD. To the best of our knowledge, most existing works on SDGs do not give per-query error bounds to the user. AIM [30] is a novel differentially private SDG that generates D_s while minimizing average error over all the input marginal queries, and only gives probabilistic upper-bounds on the error for marginal queries in the *downward closure* of the input workload. It follows the *select-measure-generate* paradigm [25, 27, 28]. However, it differs from our model wherein we are given D_s from some black-box SDG and the goal is to decide if the distance between the given (count, sum, or median) query q's output on D and D_s is less than the user-provided threshold τ. We do not make assumptions about the SDG and do not require that the given query should be well approximated by the SDG (or D_s).

We also found that the *parametric bootstrap approach* [16] does not work well because the underlying assumption about the difference between bootstrap estimates and $q(D_s)$ being representative of the difference between $q(D_s)$ and $q(D)$ does not hold when the SDG (e.g. PrivBayes [43]) uses techniques like post-processing.

Table 3: IPUMS-CPS data: Average runtimes.

	LM_{count}	EM_{count}
$Time(s)$	0.312	0.321

	LM_{sum}	$R2T_{sum}$	SVT_{sum}
$Time(s)$	0.450	4.022	9.131

	EM_{med}	$Hist_{med}$
$Time(s)$	0.533	1.043

Table 4: Summary of proposed solutions and our recommendations based on theoretical and empirical results. Those marked by (plug-in) are based on Algorithm 1, whereas the rest solve the problem without plugging-in an estimate for $q(D)$. Some theoretical upper bounds remain open problems.

Query	Solution	$\tau_{min}^{\mathcal{A},\delta}$ (upper bound)	Conclusions
COUNT	LM_{count} (plug-in)	$\frac{1}{\epsilon} \ln \frac{1}{2\delta}$	LM_{count} is the better choice, for $\delta < \frac{1}{2}$ unless $q(D) = q(D_s)$.
	EM_{count} (direct)	$\frac{1}{\epsilon} \ln \frac{1-\delta}{\delta}$	
SUM	LM_{sum} (plug-in)	$\frac{GS_q}{\epsilon} \ln \frac{1}{2\delta}$	If approx. $DS_{q,D}$ value is known to the user, and $DS_{q,D}$ is much smaller than GS_q, then choose SVT_{sum} or $R2T_{sum}$. Otherwise, choose LM_{sum} unless ϵ is small.
	$R2T_{sum}$ (plug-in)	$4\log(GS_q) \cdot \ln\left(\frac{\log(GS_q)}{\delta}\right) \frac{DS_{q,D}}{\epsilon}$	
	SVT_{sum} (direct)	–	
MEDIAN	EM_{med} (plug-in)	–	$Hist_{med}$ empirically gives smaller error than EM_{med}.
	$Hist_{med}$ (direct)	–	

9 CONCLUSIONS AND FUTURE WORK

We have presented the problem of measuring the per-query distance between the output on private data and synthetic data, and detailed our error analysis for COUNT, MEDIAN and SUM queries. Our proposed solutions fall in two classes: (1) use a DP algorithm to answer the query and check if the noisy answer is close to the answer on the synthetic data, and (2) design specialized algorithm. In addition to analyzing the error, we also introduce the notion of effectiveness of a per-query decider and derive upper bounds on the effectiveness thresholds of solutions for COUNT and SUM queries (except SVT_{sum}). Deriving such bounds for MEDIAN query is future work. We find that some mechanisms work better for smaller τ. Extending our work to other useful queries involving other aggregate functions, joins, subqueries, and group-by is future work. Designing baselines and benchmarks for the problem in this work is future work, and may be of interest to the synthetic data and data privacy communities.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments. This work was supported in part by NSF awards IIS-2147061, IIS-2008107, IIS-1703431, and IIS-1552538.

REFERENCES

[1] 2022. New york city taxi and limousine commission (tlc) trip record data. https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

[2] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2867–2867. https://doi.org/10.1145/3219819.3226070

[3] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. 2019. Bounding User Contributions: A Bias-Variance Trade-off in Differential Privacy. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 263–271. https://proceedings.mlr.press/v97/amin19a.html

[4] Kuntai Cai, Xiaoyu Lei, Jianxin Wei, and Xiaokui Xiao. 2021. Data Synthesis via Differentially Private Markov Random Field. *Proc. VLDB Endow.* 14, 11 (2021), 2190–2202. http://www.vldb.org/pvldb/vol14/p2190-cai.pdf

[5] CCPA 2023. California Consumer Privacy Act (CCPA). https://oag.ca.gov/privacy/ccpa.

[6] Graham Cormode, Magda Procopiuc, Entong Shen, Divesh Srivastava, and Ting Yu. 2011. Differentially Private Spatial Decompositions. *CoRR* abs/1103.5170 (2011). arXiv:1103.5170 http://arxiv.org/abs/1103.5170

[7] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3574–3583.

[8] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2022. R2T: Instance-Optimal Truncation for Differentially Private Query Evaluation with Foreign Keys. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 759–772. https://doi.org/10.1145/3514221.3517844

[9] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[10] Cynthia Dwork. 2006. Differential Privacy. In *ICALP*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.), Vol. 4052. 1–12.

[11] Cynthia Dwork. 2019. Differential Privacy and the US Census. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Amsterdam, Netherlands) *(PODS '19)*. Association for Computing Machinery, New York, NY, USA, 1. https://doi.org/10.1145/3294052.3322188

[12] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Advances in Cryptology (EUROCRYPT 2006)* (advances in cryptology (eurocrypt 2006) ed.) *(Lecture Notes in Computer Science)*, Vol. 4004. Springer Verlag, 486–503. https://www.microsoft.com/en-us/research/publication/our-data-ourselves-privacy-via-distributed-noise-generation/

[13] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Third Conference on Theory of Cryptography* (New York, NY) *(TCC'06)*. Springer-Verlag, Berlin, Heidelberg, 265–284. https://doi.org/10.1007/11681878_14

[14] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407. https://doi.org/10.1561/0400000042

[15] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. Association for Computing Machinery, New York, NY, USA, 1054–1067. https://doi.org/10.1145/2660267.2660348

[16] Cecilia Ferrando, Shufan Wang, and Daniel Sheldon. 2020. General-Purpose Differentially-Private Confidence Intervals. *CoRR* abs/2006.07749 (2020). arXiv:2006.07749 https://arxiv.org/abs/2006.07749

[17] Sarah Flood, Miriam King, Renae Rodgers, Steven Ruggles, J. Robert Warren, and Michael Westberry. 2022. Integrated Public Use Microdata Series, Current Population Survey: Version 10.0 [dataset]. Minneapolis, MN: IPUMS, 2022. https://doi.org/10.18128/D030.V10.0

[18] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. 2014. Dual Query: Practical Private Query Release for High Dimensional Data. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014 (JMLR Workshop and Conference Proceedings)*, Vol. 32. JMLR.org, 1170–1178. http://proceedings.mlr.press/v32/gaboardi14.html

[19] 2016-04-27. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *OJ* (2016-04-27).

[20] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F. Ilyas. 2021. Kamino: Constraint-Aware Differentially Private Data Synthesis. *Proc. VLDB Endow.* 14, 10 (2021), 1886–1899. http://www.vldb.org/pvldb/vol14/p1886-ge.pdf

[21] Moritz Hardt, Katrina Ligett, and Frank Mcsherry. 2012. A Simple and Practical Algorithm for Differentially Private Data Release. In *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/208e43f0e45c4c78cafadb83d2888cb6-Paper.pdf

[22] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[23] Ziyue Huang, Yuting Liang, and Ke Yi. 2021. Instance-optimal Mean Estimation Under Differential Privacy. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 25993–26004. https://proceedings.neurips.cc/paper/2021/file/da54dd5a0398011cdfa50d559c2c0ef8-Paper.pdf

[24] Zhiqi Huang, Ryan McKenna, George Bissias, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2019. PSynDB: Accurate and Accessible Private Data Generation. *Proc. VLDB Endow.* 12, 12 (2019), 1918–1921. https://doi.org/10.14778/3352063.3352099

[25] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. 2010. Optimizing Linear Counting Queries under Differential Privacy. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Indianapolis, Indiana, USA) *(PODS '10)*. Association for Computing Machinery, New York, NY, USA, 123–134. https://doi.org/10.1145/1807085.1807104

[26] Min Lyu, Dong Su, and Ninghui Li. 2017. Understanding the Sparse Vector Technique for Differential Privacy. *Proc. VLDB Endow.* 10, 6 (feb 2017), 637–648. https://doi.org/10.14778/3055330.3055331

[27] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing Error of High-Dimensional Statistical Queries under Differential Privacy. *Proc. VLDB Endow.* 11, 10 (jun 2018), 1206–1219. https://doi.org/10.14778/3231751.3231769

[28] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2021. HDMM: Optimizing error of high-dimensional statistical queries under differential privacy. *CoRR* abs/2106.12118 (2021). arXiv:2106.12118 https://arxiv.org/abs/2106.12118

[29] Ryan McKenna, Gerome Miklau, and Daniel Sheldon. 2021. Winning the NIST Contest: A scalable and general approach to differentially private synthetic data. *CoRR* abs/2108.04978 (2021). arXiv:2108.04978 https://arxiv.org/abs/2108.04978

[30] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. 2022. AIM: An Adaptive and Iterative Mechanism for Differentially Private Synthetic Data. *CoRR* abs/2201.12677 (2022). arXiv:2201.12677 https://arxiv.org/abs/2201.12677

[31] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 4435–4444. http://proceedings.mlr.press/v97/mckenna19a.html

[32] Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*. IEEE Computer Society, USA, 94–103. https://doi.org/10.1109/FOCS.2007.41

[33] The pandas development team. 2020. *pandas-dev/pandas: Pandas*. https://doi.org/10.5281/zenodo.3509134

[34] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data Synthesis Based on Generative Adversarial Networks. *Proc. VLDB Endow.* 11, 10 (jun 2018), 1071–1083. https://doi.org/10.14778/3231751.3231757

[35] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. 2016. The Synthetic Data Vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016, Montreal, QC, Canada, October 17-19, 2016*. IEEE, 399–410. https://doi.org/10.1109/DSAA.2016.49

[36] Shweta Patwa, Danyu Sun, Amir Gilad, Ashwin Machanavajjhala, and Sudeepa Roy. 2023. DP-PQD: Privately Detecting Per-Query Gaps In Synthetic Data Generated By Black-Box Mechanisms. arXiv:2309.08574 [cs.DB]

[37] Lucas Rosenblatt, Xiaoyan Liu, Samira Pouyanfar, Eduardo de Leon, Anuj Desai, and Joshua Allen. 2020. Differentially Private Synthetic Data: Applied Evaluations and Enhancements. *CoRR* abs/2011.05537 (2020). arXiv:2011.05537 https://arxiv.org/abs/2011.05537

[38] SF1 2012. https://www2.census.gov/library/publications/cen2010/doc/sf1.pdf.

[39] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and XiaoFeng Wang. 2017. Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12. *CoRR* abs/1709.02753 (2017). arXiv:1709.02753 http://arxiv.org/abs/1709.02753

[40] Om Thakkar, Galen Andrew, and H. Brendan McMahan. 2019. Differentially Private Learning with Adaptive Clipping. *CoRR* abs/1905.03871 (2019). arXiv:1905.03871 http://arxiv.org/abs/1905.03871

[41] Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Steven Wu. 2020. New Oracle-Efficient Algorithms for Private Synthetic Data Release. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 9765–9774. https://proceedings.mlr.press/v119/vietri20b.html

[42] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. *Modeling Tabular Data Using Conditional GAN.* Curran Associates Inc., Red Hook, NY, USA.

[43] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. *ACM Trans. Database Syst.* 42, 4, Article 25 (Oct. 2017), 41 pages. https://doi.org/10.1145/3134428