2021-06-18

Non-parametric differentially private confidence intervals for the median

Non-parametric Differentially Private Confidence Intervals for the Median

Jörg Drechsler[1, 2], Ira Globus-Harris[3], Audra McMillan[*4], Jayshree Sarathy[5], and Adam Smith[†6]

[1]Institute for Employment Research, Germany
[2]The Joint Program in Survey Methodology, University of Maryland, USA
[3]University of Pennsylvania, USA
[4]Apple, USA
[5]Harvard John A. Paulson School of Engineering and Applied Sciences, USA
[6]Department of Computer Science, Boston University, USA

July 6, 2021

Abstract

Differential privacy is a restriction on data processing algorithms that provides strong confidentiality guarantees for individual records in the data. However, research on proper statistical inference, that is, research on properly quantifying the uncertainty of the (noisy) sample estimate regarding the true value in the population, is currently still limited. This paper proposes and evaluates several strategies to compute valid differentially private confidence intervals for the median. Instead of computing a differentially private point estimate and deriving its uncertainty, we directly estimate the interval bounds and discuss why this approach is superior if ensuring privacy is important. We also illustrate that addressing both sources of uncertainty–the error from sampling and the error from protecting the output–simultaneously should be preferred over simpler approaches that incorporate the uncertainty in a sequential fashion. We evaluate the performance of the different algorithms under various parameter settings in extensive simulation studies and demonstrate how the findings could be applied in practical settings using data from the 1940 Decennial Census.

1 Introduction

Statistical agencies constantly need to find the right balance between the two competing goals of disseminating useful information from their collected data and ensuring the confidentiality of the units included in the database. Many methods have been developed in the past decades to address this trade-off. However, with the advent of modern computing and the massive amounts of data collected every day, many of the data protection strategies commonly used at statistical agencies are no longer adequate to sufficiently protect the data [Abowd, 2018, Garfinkel et al., 2019]. The problem's difficulty is amplified by the continual appearance of new data sources that facilitate attacks.

One promising strategy to circumvent this dilemma is to rely on formal privacy guarantees such as those provided by differential privacy (DP) [Dwork et al., 2006b]. These guarantees hold no matter what background knowledge a potential attacker might possess, or how much computational power they have. However, methodology for differential private statistical inference has mostly been studied from a theoretical perspective under asymptotic regimes. Although many algorithms have been proposed to ensure formal privacy guarantees for various estimation tasks, evaluations of their relative performance on real data with limited sample sizes and complex distributional properties are still limited,

[*]Part of this work was completed while the author was at Boston University and Northeastern University.
[†]Authors in alphabetical order.

and only a small fraction of that literature has focused on inference and associated measures of uncertainty. Section 2.3 surveys related work.

In this paper, we address these issues, focusing on one of the key measures of location: the median. We chose the median for two reasons. On one hand, it is a widely used summary statistic for skewed variables such as income (see, for example, the U.S. Census Bureau's tables of median incomes for various subgroups of the population [U.S. Census Bureau, 2020a]). On the other hand, medians provide an interesting technical challenge for differentially private computation. The accuracy of differentially private median computations depends on the exact data distribution; as a result, providing sound and narrow confidence intervals appears to require releasing strictly more information about the data than is required for point estimation.

The discussion of confidence intervals is an important contribution of our paper. None of the previously proposed algorithms for DP median estimation come equipped with a method for additionally releasing DP uncertainty estimates on the point estimator. In fact, the level of uncertainty in the point estimate is typically data dependent, and hence measuring it requires additional privacy budget. Thus, the optimal algorithm for differentially private point estimates can be different from the optimal algorithm for differentially private confidence intervals. Instead of deriving the variance of some differentially private point estimate, we suggest estimating DP confidence intervals directly. We show that our proposed methodology ensures proper confidence interval coverage in a frequentist sense and discuss why this strategy requires less privacy budget than starting from the protected point estimates.

When designing and analysing differentially private algorithms it is tempting to separate the error due to sampling from the error due to privacy and bound the two separately. A main finding in our work is the limitation of this approach. We find that one can obtain considerably tighter confidence intervals by analysing the relationship between the two sources of error. Unlike approaches which treat the analysis of the non-private algorithm as a black-box, this involves looking at the different ways that the sampling error can result in the confidence interval failing to capture the median, and considering how the error due to privacy affects each of these modalities.

We assume simple random sampling throughout the paper. This assumption is often violated in survey practice. However, understanding the implications of complex sampling designs on the privacy guarantees is an open research problem [Drechsler, 2021] and we are not aware of any DP applications that take complex sampling designs into account. We see our contribution as an important first step towards the goal of better serving the needs of statistical agencies, while acknowledging the limitations of the current findings. We will come back to this point in the conclusions.

We evaluate several algorithms for computing valid differentially private confidence intervals. We discuss algorithms that satisfy two versions of differential privacy: the strictest version [Dwork et al., 2006b], now known as *pure differential privacy*, as well as a slight relaxation, *concentrated differential privacy* [Bun and Steinke, 2016, Dwork and Rothblum, 2016]. The focus of our paper is on empirical evaluation, using a mix of simulated and real data. Nevertheless, we found that new methodology and theory was also needed to adapt existing algorithms for confidence interval computation. We include an application using data from the U.S. Census 1940 to illustrate how statistical agencies willing to adopt the methodology could decide which algorithm and parameter settings to pick for their data release.

The algorithms we developed are all *sound* in the nonparametric, frequentist sense: when run with nominal coverage $1-\alpha$ the probability that the true population median is contained in the computed confidence interval is at least $1-\alpha$, where the probability is taken over the entire process of sampling from the population and computing the private confidence intervals based on the drawn sample. Since all algorithms rely on non-parametric strategies for computing the confidence intervals, the intervals are valid for every IID distribution on observations. We summarize our findings briefly:

- In our comparison of several algorithms, the best choice across a range of settings was a variant of the exponential mechanism [McSherry and Talwar, 2007], a generic framework for DP algorithm design that we adapt for confidence interval estimation. This algorithm is tailored to the median, and releases only a single confidence interval.

- A different algorithm, based on a differentially private CDF estimate [Li et al., 2010], consistently produced confidence intervals that were slightly wider than those of the exponential mechanism. However, the algorithm's output can be used to produce a confidence interval for any quantile of the data set or even a confidence band

for the entire CDF. In principle, the approximation to the entire CDF would also allow the incorporation of a parametric model or a Bayesian prior after the fact. Its flexibility makes it a better choice for settings where eventual users will be interested in more than a single median.

- For settings where only a very loose bound on the range of the data range is known a priori, a hybrid algorithm that uses binary search to narrow the range and then switches to the CDF-based estimator produces narrower intervals than other methods.
- The methods we tested exhibited noticeable bias that depends on the underlying distribution and appears to be hard to correct. The bias was low relative to the width of the confidence intervals, and so would not be an issue for one-shot applications. However, it might be a concern when aggregating estimates across many small areas. It is not clear whether bias is necessary for accurate nonparametric DP median approximations. [1]

The remainder of the paper is organized as follows: In Section 2 we review some of the privacy definitions that are relevant for this paper and discuss confidence interval estimation for the median without privacy considerations. We extend these discussions to differentially private confidence intervals in Section 3. Section 4 contains a high-level review of the algorithms we considered (detailed descriptions of the different algorithms can be found in the Appendix). In Section 5 we present the results from extensive simulation studies that evaluate the performance of the algorithms under various parameter settings. Section 6 illustrates how the methodology could be applied in practice by replicating one of the income tables published by the U.S. Census Bureau using publicly available data from the 1940 U.S. Census. The paper concludes with some final remarks.

2 Preliminaries

2.1 Differential Privacy

The algorithms in this paper satisfy a version of differential privacy (DP) called *concentrated differential privacy* (CDP). This notion of privacy lies between the more common notions of *pure differential privacy* and *approximate differential privacy*. Since our algorithms often include hyperparameters, we state a definition of DP for algorithms that take as input not only the dataset, but also the desired privacy parameters and any required hyperparameters. Let $\mathscr{X}$ be a data universe (e.g., $\mathbb{R}$ for medians) and $\mathscr{X}^n$ be the space of datasets of size n. Two datasets $d, d' \in \mathscr{X}^n$ are neighboring, denoted $d \sim d'$, if they differ on a single record. Let $\mathscr{H}$ be the space of hyperparameters and $\mathscr{Y}$ be an output space. In order to build some intuition, let us first define pure and approximate DP.

Definition 2.1 ((ε,δ)-Differential Privacy [Dwork et al., 2006b,a])**.** *Given $\varepsilon \geq 0$ and $\delta \in [0,1]$, a randomized mechanism $M : \mathscr{X}^n \times \mathscr{H} \to \mathscr{Y}$ is* (ε,δ)-differentially private *if for all datasets $d \sim d' \in \mathscr{X}^n$, hyperparams $\in \mathscr{H}$, and events $E \subseteq \mathscr{Y}$,*

$$\Pr[M(d, hyperparams) \in E] \leq e^{\varepsilon} \cdot \Pr[M(d', hyperparams) \in E] + \delta,$$

where the probabilities are taken over the random coins of M.

The key intuition for this definition is that the distribution of outputs on input dataset d is almost indistinguishable from the distribution on outputs on input dataset d'. Therefore, given the output of a differentially private mechanism, it is impossible to confidently determine whether the input dataset was d or d'. If $\delta = 0$, then we refer to this as ε-*pure differential privacy*. If $\delta > 0$, we refer to (ε,δ)-*approximate differential privacy*. For strong privacy guarantees, the privacy-loss parameter is typically taken to be a small constant less than 1 (note that $e^{\varepsilon} \approx 1+\varepsilon$ as $\varepsilon \to 0$). However, in practice, larger values of ε are occasionally used to satisfy utility constraints while providing some level of non-trivial privacy guarantee.

Concentrated differential privacy has the same intuition; it bounds the divergence between the distributions $M(d)$ and $M(d')$.

[1] The one unbiased method that we tested (based on the *smooth sensitivity* framework [Nissim et al., 2007]) produced relatively poor point estimates, and we did not include it in the tests of confidence interval width. See Figure 14 in Appendix F for analysis of performance of CDP point estimators for the median.

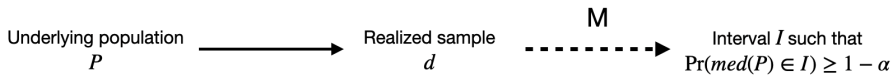


Figure 1: A graphical representation of the process of computing a confidence interval. When privacy is not a concern, no restrictions are placed on the function M. When computing a differentially private confidence interval, we require that M is differentially private. The probability is taken over all the randomness in the system, both the randomness due to sampling, and the randomness in M.

Definition 2.2 (ρ-Concentrated Differential Privacy [Bun and Steinke, 2016])**.** *Given $\rho \geq 0$, a randomized mechanism $M : \mathscr{X}^n \times \mathscr{H} \to \mathscr{Y}$ satisfies* ρ-concentrated differential privacy *if for all datasets $d \sim d' \in \mathscr{X}^n$, hyperparams $\in \mathscr{H}$, and $\alpha \in (1, \infty)$,*

$$D_\alpha(M(d, hyperparams) \| M(d', hyperparams)) \leq \rho$$

where D_α is the α-Rényi divergence and the probabilities are taken over the random coins of M.

In order to give some intuition for concentrated DP, let us elaborate more on its relationship with pure and approximate DP. Given data sets $d \sim d'$, and a randomised mechanism M, we can define a random variable, called the *privacy loss random variable*, denoted $Z = \texttt{Priv}(M(d), M(d'))$, as follows. Let $y \sim M(d)$ (i.e. y is the output of the mechanism M on input d), then $Z = \ln\left(\frac{\Pr(M(d))=y}{\Pr(M(d'))=y}\right)$. Then M is ε-pure differentially private if and only if $\Pr(|Z| > \varepsilon) = 0$, and M being (ε, δ)-approximately differentially private is (almost) captured by the requirement that $\Pr(|Z| > \varepsilon) \leq \delta$. Now, ρ-concentrated differential privacy essentially translates to the requirement that Z is a subgaussian random variable with mean ρ and variance 2ρ. From this perspective, it is clear that concentrated differential privacy lies between pure and approximate DP.

Lemma 2.1. *If M is ρ-CDP, then M is $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-DP for any $\delta > 0$. If M is ε-DP then M is $\frac{1}{2}\varepsilon^2$-CDP.*

We will focus in this paper on algorithms that satisfy concentrated differential privacy. While still satisfying a rigorous notion of privacy, this will allow our algorithms to be significantly more accurate than their corresponding purely differentially private counterparts. For most of our algorithms little accuracy is gained from transitioning to approximate differential privacy. Additionally, CDP has the desirable property of being a one-parameter property, which allows for simpler privacy accounting. The lemma below captures the fact that the class of ρ-CDP algorithms is closed under adaptive composition and post-processing.

Lemma 2.2. *[Bun and Steinke, 2016] Let $M : \mathscr{X}^n \times \mathscr{H} \to \mathscr{Y}$ and $M' : \mathscr{X}^n \times \mathscr{H}' \to \mathscr{Y}'$, where $\mathscr{H}' = \mathscr{Y} \times \mathscr{H}''$. Define $M'' : \mathscr{X}^n \times (\mathscr{H}' \times \mathscr{H}'')$ by*

$$M''(d, hyperparams, hyperparams') = M'(d, (M(d, hyperparams), hyperparams')).$$

If M is ρ-CDP and M' is ρ'-CDP then M'' is $\rho + \rho'$-CDP.

2.2 Confidence Intervals for the Median

In many statistical applications, we assume that data are drawn i.i.d. from an underlying population distribution, and the statistic of interest is a property of the underlying population. However, one typically only has access to a sample from that population, so the statistic computed on the sample is used as an *estimate* of the true population statistic. We will refer to the median of the underlying population as the *population median* and the median of a given sample as the *sample median*. Since there is randomness in the sampling process, there is always uncertainty in how well the sample median matches the true population median. As this uncertainty can be large, sample statistics should be accompanied by a measure of the uncertainty. Providing a measure of uncertainty is even more important for differentially private statistics since randomness in the algorithm provides an additional source of uncertainty.

One method for capturing the uncertainty in an estimate is a confidence interval. We consider the standard set-up for statistical inference. Let $\mathscr{P} \subset \Delta(\mathbb{R})$ be the set of possible population distributions over the data domain $\mathbb{R}$. For any $P \in \mathscr{P}$, a median of P is defined to be any value m such that

$$\int_{-\infty}^{m} P(x)dx \geq 1/2 \text{ and } \int_{m}^{\infty} P(x)dx \geq 1/2.$$

For every distribution the set of medians is a non-empty, compact set. Since defining a convention here will be convenient, we will refer to the midpoint of the set of medians as *the median*, denoted $\texttt{med}(P)$. Let $I_{\mathbb{R}}$ be the set of intervals in $\mathbb{R}$. Given $n \geq 0$, let $M : \mathscr{X}^n \to I_{\mathbb{R}}$ be a randomised mechanism that takes as input a data set of size n and outputs an interval in $\mathbb{R}$. Given a desired confidence level $1-\alpha$, the goal of M is to, with probability $1-\alpha$, output an interval that contains $\texttt{med}(P)$.

Definition 2.3. *For any $\alpha \in [0,1]$ and $n \in \mathbb{N}$, $M : \mathscr{X}^n \to I_{\mathbb{R}}$ is a $(1-\alpha)$-*confidence interval for the median *for $\mathscr{P}$ if for all $P \in \mathscr{P}$,*

$$\Pr(\texttt{med}(P) \in M(d)) \geq 1-\alpha,$$

where the randomness is taken over both the randomness M and the randomness in the sample $d \sim P^n$.

A graphical representation of the framework for computing a confidence interval is given in Figure 1. We will refer to $1-\alpha$ as the *coverage* of the confidence interval.

When one is not concerned with privacy, a non-parametric confidence interval for the median can be computed using the order statistics of the sample. The rank of the median $\texttt{med}(P)$ in a data set $d \in P^n$ is distributed as the binomial $\text{Bin}(n,\beta)$, for some $\beta = \Pr_P(x < \texttt{med}(P))$. We can exploit this to obtain a confidence interval for the median. For a data set $d \in \mathbb{R}^n$, let $d_{(k)}$ denote the k-th smallest value in d, referred to as the k-th order statistic. The median of d is the midpoint of $d_{(\lfloor n/2 \rfloor)}$ and $d_{(\lceil n/2 \rceil)}$.

Lemma 2.3 (Non-private $(1-\alpha)$-confidence interval)**.** *Let C_{Bin} be the CDF of the binomial random variable* $\text{Bin}(n,1/2)$ *and let*

$$M_L^\alpha = \max_{m \in \mathbb{N}} \{m \mid C_{\text{Bin}}(m) \leq \alpha/2\} \text{ and } M_U^\alpha = \min_{m \in \mathbb{N}} \{m \mid C_{\text{Bin}}(m) \geq 1-\alpha/2\}.$$

For any data set $d \in \mathbb{R}^n$, let

$$\texttt{ci}_L^\alpha(d) = d_{(M_L^\alpha)} \text{ and } \texttt{ci}_U^\alpha(d) = d_{(M_U^\alpha)}.$$

Then $M : \chi^n \to I_{\mathbb{R}}$ given by $M(d) = [\texttt{ci}_L^\alpha(d), \texttt{ci}_U^\alpha(d)]$ is a $(1-\alpha)$-confidence interval for the median for $\Delta(\mathbb{R})$.

We will often refer to the interval, $[\texttt{ci}_L^\alpha(d), \texttt{ci}_U^\alpha(d)]$, output by the mechanism in Lemma 2.3 as *the non-private $(1-\alpha)$-confidence interval for the median*, but we note that it is the mechanism M that satisfies Definition 2.3, not the output.

In this vein, the goal of CDP confidence intervals is not to privately estimate the specific interval $[\texttt{ci}_L^\alpha(d), \texttt{ci}_U^\alpha(d)]$, but to output valid confidence intervals. These confidence intervals may, or may not, contain $[\texttt{ci}_L^\alpha(d), \texttt{ci}_U^\alpha(d)]$. Referring to Figure 1, producing a CDP confidence interval involves the same procedure, with the additional requirement that M is CDP. That is, the realised sample d is only accessed through a CDP mechanism.

Definition 2.4. *$M : \mathscr{X}^n \times \mathscr{H} \to I_{\mathbb{R}}$ is a ρ-CDP, $(1-\alpha)$-confidence interval for the median for $\mathscr{P}$ if*

- *M is ρ-CDP*
- *For any hyperparams $\in \mathscr{H}$, $M(\cdot, \text{hyperparams})$ is an $(1-\alpha)$-confidence interval for the median for $\mathscr{P}$.*

The definition of confidence intervals as stated in Definition 2.3 and Definition 2.4 only requires that the confidence interval is valid for distributions $P \in \mathscr{P}$. *Parametric* estimation is when one defines $\mathscr{P}$ to be only distributions of a particular, often quite simple, form. For example, $\mathscr{P}$ might be the set of all log-normal distributions over $\mathbb{R}$. In *non-parametric* estimation, we assume no knowledge of the underlying population and set $\mathscr{P} = \Delta(\mathbb{R})$, the set of all distributions over $\mathbb{R}$. If one has accurate knowledge of the underlying population, then parametric estimation can result in a tighter confidence interval. However, if there is model mismatch (for example if the underlying population is not

exactly log-normal) then parametric estimation can result in invalid confidence intervals. This effect can be amplified by private algorithms which may rely on the modeling assumptions in non-trivial ways. In this paper our goal is to focus on non-parametric confidence intervals, which means our algorithms will always produce valid confidence intervals. As a minor caveat, we restrict ourselves to the set of distributions with continuous probability density functions on $\mathbb{R}$.[2] Denote the set of all continuous distributions on $\mathbb{R}$ by $\Delta_{\mathscr{C}}(\mathbb{R})$.

Note that a confidence interval does not directly output a point estimate for the median itself. In the absence of privacy constraints, one can simply additionally release the sample median $\texttt{med}(d)$. However, under privacy constraints, it is typically desirable to compute as few statistics as possible, in order to allocate the maximum amount of privacy budget to each statistic. As such, rather than allocating some of the privacy budget to providing a point estimate of the median, it is often preferable to allocate the entire budget to estimating the confidence interval, then use the midpoint of that interval as a point estimate of the median.

2.3 Related Work

Computing confidence intervals for the median is one of the most fundamental statistical tasks. However, finding a differentially private estimator for this task that is accurate across a range of datasets and parameter regimes is surprisingly nuanced. There has been a significant amount of prior work on differentially private point estimators for the median [Nissim et al., 2007, Bun and Steinke, 2019, Asi and Duchi, 2020, Alabi et al., 2020, Tzamos et al., 2020] and other quantiles [Gillenwater et al., 2021]. To the best of our knowledge, none of these works addressed DP confidence intervals for the median. However, there has been significant work on DP confidence intervals for other estimation tasks like (Gaussian or sub-Gaussian) mean estimation [Karwa and Vadhan, 2018, Gaboardi et al., 2019, Du et al., 2020, Biswas et al., 2020], and linear regression [Barrientos et al., 2017, Evans and King, 2021]. There are also several works on designing more general DP confidence intervals using bootstrapping, or a technique called subsample-and-aggregate [Nissim et al., 2007], to account for the combined uncertainty from sampling and noise due to privacy [Barrientos et al., 2017, Ferrando et al., 2020, Brawner and Honaker, 2018, D'Orazio et al., 2015, Evans et al., 2021]. These algorithms typically require a parametric model on the data or a normality assumption on the quantity being estimated; neither hold in our setting.

The areas of differentially private bayesian inference [Dimitrakakis et al., 2014, Wang et al., 2015a, Foulds et al., 2016, Heikkilä et al., 2017, Bernstein and Sheldon, 2018, 2019, Gong, 2019] and hypothesis testing [Vu and Slavkovic, 2009, Couch et al., 2019, Degue and Ny, 2018, Gaboardi et al., 2016, Wang et al., 2015b] study related problems of quantifying uncertainty, but specific goals differ. Wang [2018], Du et al. [2020], and Biswas et al. [2020] perform experimental evaluations of DP confidence intervals, however they focus on different estimators (linear regression and mean estimation) and focus on large datasets of at least 1,000, and generally many more, data points.

To the best of our knowledge, our work is unique in focusing on valid non-parametric differentially private confidence intervals for the median. This approach allows us to define algorithms that provide accurate and private confidence intervals without requiring distributional assumptions on the underlying population.

3 Designing DP Confidence Intervals

3.1 Roadblocks and first attempts

There are several roadblocks in designing CDP confidence intervals for the median. Firstly, CDP algorithms that estimate the median using data independent output perturbation methods (methods that involve simply adding noise to the non-private estimate) necessarily perform poorly since the median is very sensitive for worst-case data sets. Thus, in order to design algorithms that perform well on "typical" data sets, the noise addition must be data dependent. This creates difficulties when releasing information regarding the uncertainty in the private estimate since the uncertainty itself might reveal sensitive information. In order to explore these roadblocks in more detail, let us first consider the simpler task of designing a CDP point estimator for the median. A first attempt may be to consider the global sensitivity [Dwork et al., 2006b]:

[2]Note this caveat is minor since continuous distributions are dense in $\Delta(\mathbb{R})$. That is every distribution on $\mathbb{R}$ is within negligible distance of a continuous distribution. We discuss a practical method for handling non-continuous distributions in Appendix A

Definition 3.1 (Global Sensitivity)**.** *For a query $f: \mathscr{X}^n \to \mathbb{R}$, the **global sensitivity** is*

$$GS_f = \max_{d \sim d'} |f(d) - f(d')|.$$

For any function f, one can create a differentially private mechanism by adding noise proportional to $\mathrm{GS}_f/\sqrt{\rho}$. If one has no bound on the data, then $\mathrm{GS}_{\mathtt{med}}$ is infinite. Even if one knows that all the data lie in a bounded range $[a,b]$, $\mathrm{GS}_{\mathtt{med}} = |b-a|$, and adding noise proportional to $|b-a|/\sqrt{\rho}$ essentially removes the signal for any reasonable value of ρ.

However, for the type of datasets that we typically see in practice, changing one data point, or even a few data points, does not result in a major change in the median. For such data sets, one might consider the local sensitivity [Nissim et al., 2007], which can be substantially smaller than the global sensitivity.

Definition 3.2 (Local Sensitivity [Nissim et al., 2007])**.** *The **local sensitivity** of a query $f: \mathscr{X}^n \to \mathbb{R}$ with respect to a dataset $d \in \mathscr{X}^n$ is*

$$LS_f(d) = \max_{d \sim d'} |f(d) - f(d')|.$$

Unfortunately, since the local sensitivity itself is data dependent, adding noise proportional to the local sensitivity is not differentially private. Several approaches have been explored in the DP literature for adding noise that is *close* to the local sensitivity, or at least significantly less than the global sensitivity for typical data sets. In [Nissim et al., 2007], Nissim et al. define the *smooth sensitivity*, SS_f, a smooth upper bound on the local sensitivity such that adding noise proportional to $SS_f/\sqrt{\rho}$ is differentially private. They showed that for many statistics, including the median, the smooth sensitivity can be much smaller than the global sensitivity on typical data sets. Several other DP mechanisms for median estimation have been proposed that avoid the large $\mathrm{GS}_{\mathtt{med}}$ by calibrating the noise introduced to the specific data set [Nissim et al., 2007, Dwork and Lei, 2009]. While these estimators can perform well as point estimators for the median, our goal is not just to provide an estimate of the median, but also to quantify the uncertainty in our estimate. In algorithms like the smooth sensitivity mechanism that tailor the noise to the specific data set, the uncertainty itself is data dependent and thus sensitive. The task of differentially privately releasing an estimate of this uncertainty is nontrivial. Even if one could release a DP estimate of the amount of noise added to the non-private median, the uncertainty in the non-private median is still unaccounted for. For this reason, we focus on DP algorithms that attempt to directly estimate the confidence interval.

3.2 Accounting for all sources of randomness

Accurate and tight coverage analysis is a crucial component of designing good algorithms since overly conservative coverage estimates can result in confidence intervals that are wider than necessary. Valid differentially private confidence intervals need to account for two sources of error; sampling error and error due to privacy. Sampling error, also present in the non-private context, captures how well the realised sample d represents the underlying population P. The error due to privacy takes into account the additional randomness in M as a result of the privacy guarantee. Our experimental results highlight that it is important to carefully exploit the dependence between the two sources of randomness.

As a primer, let us first consider the coverage analysis of the non-private algorithm described in Lemma 2.3. This coverage analysis relies on the fact that if P is continuous then for all $m \in n$,

$$\Pr(\mathtt{rank}_d(\mathtt{med}(P)) = m) = \Pr(\mathrm{Bin}(n, 1/2) = m).$$

There are two ways that the interval $[\mathtt{ci}_L^\alpha(d), \mathtt{ci}_U^\alpha(d)]$ can fail to capture $\mathtt{med}(P)$; $\mathtt{med}(P) < \mathtt{ci}_L^\alpha(d)$ or $\mathtt{med}(P) > \mathtt{ci}_U^\alpha(d)$. Let us focus on the probability of the first type of failure, $\mathtt{med}(P) < \mathtt{ci}_L^\alpha(d)$. For every $P \in \Delta_{\mathscr{C}}(\mathbb{R})$,

$$\Pr(\mathtt{med}(P) < \mathtt{ci}_L^\alpha(d)) = \Pr(\mathtt{med}(P) < d_{(\mathtt{N}_L^\alpha)}) = C_{\mathrm{Bin}}(\mathtt{N}_L^\alpha - 1) \leq \alpha/2,$$

where C_{Bin} is the CDF of the binomial random variable $\mathrm{Bin}(n, 1/2)$. The probability of failure at the upper end of the confidence interval is analogous.

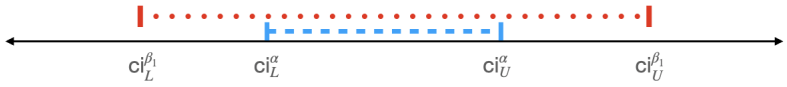


Figure 2: Graphical representation of naive coverage analysis

Now, let us turn to the coverage analysis of a ρ-CDP algorithm $M : \mathscr{X}^n \times \mathscr{H} \to I_{\mathbb{R}}$. Let $M(d) = [M(d)_L, M(d)_U]$. A naive way to analyse the coverage error of M is to attempt to find β_1 and β_2 such that assuming β_1 denotes the failure probability for the non-private confidence interval, the $M(d)$ contains the non-private interval $[\mathtt{ci}_L^{\beta_1}, \mathtt{ci}_U^{\beta_1}]$ with probability $1-\beta_2$. Then M has coverage at least $1-(\beta_1+\beta_2)$. Even if β_1 and β_2 are chosen carefully, this analysis can be overly conservative. In particular, it assumes that the only way that $M(d)$ can succeed in containing $\mathtt{med}(P)$ is if both $\mathtt{med}(P) \in [\mathtt{ci}_L^{\beta_1}, \mathtt{ci}_U^{\beta_1}]$ and $[\mathtt{ci}_L^{\beta_1}, \mathtt{ci}_U^{\beta_1}] \subset M(d)$. In Figure 2, this corresponds to ensuring that $M(d)$ contains the red dotted interval with high probability. It's clear from this figure that neither of these events are necessary.

A more careful analysis of the relationship between the sampling error and the error due to privacy results in a tighter coverage analysis. As in the non-private setting, there are two ways that $M(d)$ can fail to contain $\mathtt{med}(P)$, and we will focus on analysing the probability that $\mathtt{med}(P) < M(d)_L$

$$\begin{aligned}\Pr(\mathtt{med}(P) < M(d)_L) &= \sum_{m=0}^{n} \Pr(\mathtt{rank}_d(\mathtt{med}(P)) = m) \cdot \Pr(\mathtt{med}(P) < M(d)_L \mid \mathtt{rank}_d(\mathtt{med}(P)) = m) \\ &= \sum_{m=0}^{n} \Pr(\mathrm{Bin}(n, 1/2) = m) \cdot \Pr(\mathtt{med}(P) < M(d)_L \mid \mathtt{rank}_d(\mathtt{med}(P)) = m) \qquad (1)\end{aligned}$$

Now, we have reduced the problem to analysing the failure probability conditioned on the empirical rank of $\mathtt{med}(P)$ in the data set d. This is a helpful reduction since, as we will see in the following section, most of our algorithms will come with accuracy guarantees on the rank of $M(d)_L$. Accuracy guarantees of this form can then be exploited, via Equation (1), to obtain a coverage analysis of M.

Our experiments show a stark difference between the performance of algorithms designed using the naive analysis, and those using the tighter, more careful analysis. In Figure 7 we directly compare the confidence intervals that arise from the different analyses. This highlights the importance of understanding the relationship between the two sources of error.

4 Algorithms

In this section we will introduce the four algorithms for releasing CDP confidence intervals for the median that will be the focus of this paper; `ExpMech`, `CDFPostProcess`, `NoisyBinSearch`, and `BinSearch + CDF`. The first algorithm, which we call `ExpMech`, is based on the exponential mechanism. This mechanism is efficient, satisfies the stronger privacy guarantee of pure differential privacy and outputs the tightest, or close to the tightest confidence intervals in a majority of parameter regimes we studied. The remaining three algorithms partially address a common frustration with differentially private data analysis; that exploratory data analysis to visualise the data set and verify findings typically requires additional privacy budget. For many tasks, this means allocating privacy budget away from the primary task resulting in a noisier algorithm. A key feature of the three algorithms `CDFPostProcess`, `NoisyBinSearch` and `BinSearch + CDF` is that they release additional information about the data set without consuming additional budget. In particular, `CDFPostProcess` releases a full CDP estimate to the empirical CDF. It is then notable, and perhaps surprising, that in many settings these algorithms perform almost as well as `ExpMech`, which releases no side information.

In this section we will give a high level description of each algorithm. Further information for all algorithms, including pseudo-code and proofs of the privacy and validity guarantees, can be found in the online supplement accompanying this paper. Real code is available in our GitHub repository.[3] We note that we also experimented with

[3] `https://github.com/anonymous-conf-medians/dp-medians`

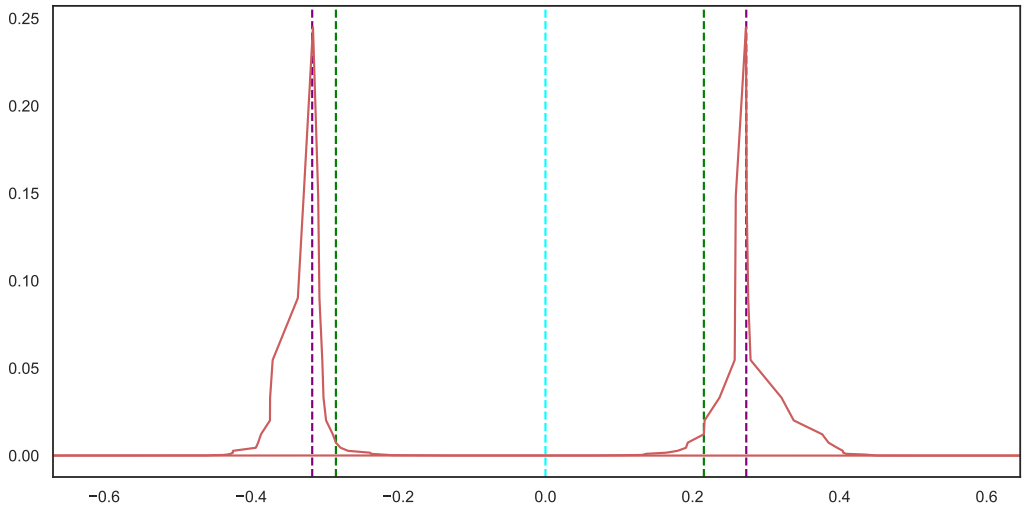


Figure 3: Graphical representation of the distribution of a 1.0-CDP `ExpMech` confidence interval on a single dataset d, whose 500 datapoints are sampled i.i.d. from $\mathcal{N}(0,4)$. The range $\mathscr{R} = [-5,5]$, granularity $\theta = 0.05$, and $\alpha = 0.05$. The cyan line indicates the population median, the green interval represents $\texttt{ci}_L^\alpha$ and $\texttt{ci}_U^\alpha$, and the purple interval represents $d_{(k_L)}$ and $d_{(k_U)}$, where k_L, k_U are chosen according to Equation (2). The red curves illustrate the theoretical distributions of the outputs of $\texttt{ExpMech}(d)$.

several other algorithms that are not discussed in this section. Brief descriptions of these additional algorithms can be found in the online supplement F, but we do not focus on them here since they are outperformed by other algorithms in every parameter regime we studied.

4.1 Confidence intervals based on exponential mechanism, `ExpMech`

Our first private mechanism is an instantiation of the exponential mechanism [McSherry and Talwar, 2007], a differentially private algorithm designed for general optimization problems. The exponential mechanism has been used in prior work to give DP point estimates for the median [Dwork and Lei, 2009, Thakurta and Smith, 2013, Johnson and Shmatikov, 2013, Alabi et al., 2020, Asi and Duchi, 2020]. Our extension to providing confidence intervals for the median, while using similar ideas to prior work, requires a careful coverage analysis that is new to this work.

The exponential mechanism is defined with respect to a utility function u, which maps (data set, output) pairs to real values. For a data set d, the mechanism aims to output a value r that maximizes $u(d,r)$ by sampling a value r with probability proportional to $e^{\varepsilon u(d,r)/\Delta u}$ where $\Delta_u = \max_r \max_{d,d' \text{neighbours}} |u(d,r) - u(d',r)|$. Recall that the rank of a value $r \in \mathbb{R}$ in a data set d, denoted $\texttt{rank}_d(r)$, is the number of data points in d that are less than or equal to r. For any $k \in [n]$, one way to instantiate the exponential mechanism to compute the k-th order statistic is by using the following utility function. Let

$$u_k(d,r) = -|\texttt{rank}_d(r) - k|.$$

In practice we will use a slight variant of the exponential mechanism described in online supplement B, which uses a granularity parameter θ. This version of the exponential mechanism has the guarantee that with high probability it will output a value within θ of some r, such that $\texttt{rank}_d(r)$ is close to k. Let us denote the exponential mechanism with privacy parameter ε for the k-th order statistic by A_k^ε.

While it is tempting to use the exponential mechanism to release DP estimates of $\texttt{ci}_L^\alpha(d) = d_{(\mathtt{N}_L^\alpha)}$ and $\texttt{ci}_U^\alpha(d) = d_{(\mathtt{N}_U^\alpha)}$, this will not create a DP α-confidence interval. This is because the private algorithm may under or over estimate

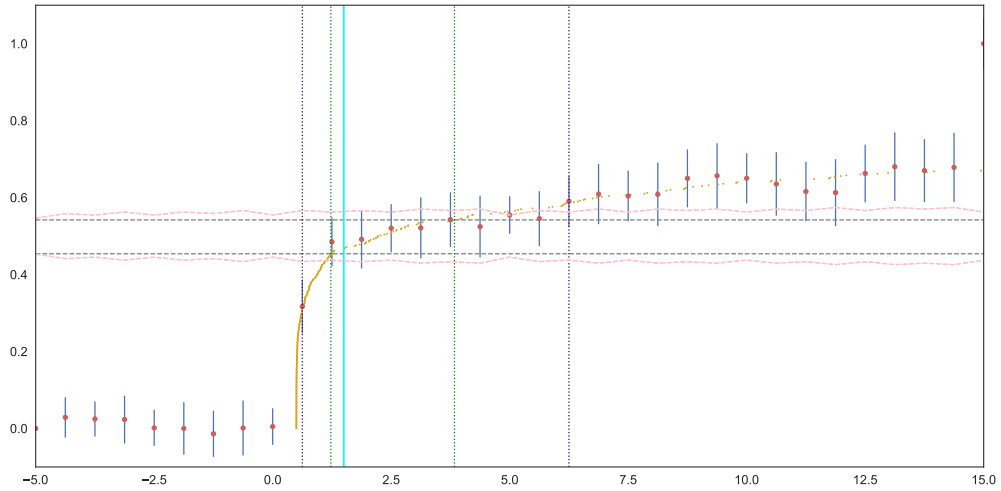


Figure 4: Graphical representation of a single run of 0.1-CDP `CDFPostProcess` (with hyperparameters $\mathscr{R} = [-5, 15], \theta = 0.5$) on a single dataset. The 500 datapoints are drawn i.i.d. from Lognormal$(\ln(1.5), 5.0)$. The cyan line is the population median, the gray dashed horizontal lines represent $\mathtt{N}_L^{0.05}/n, \mathtt{N}_U^{0.05}/n$, and the green dashed vertical lines represent $\mathtt{ci}_L^{0.05}(d)$ and $\mathtt{ci}_U^{0.05}$. For each $x \in \mathscr{R}_{\text{discrete}}$, the red dot is at $\tilde{C}(x)$, the yellow dot is at $\hat{C}(x)$, and the pink dot is at the upper and lower thresholds a_x^L and a_x^U. For each x the blue vertical line represents a point wise 99% confidence interval on $\hat{C}(x)$ based on the measurement $\tilde{C}(x)$. The purple dashed vertical lines represent the DP interval $\mathtt{CDFPostProcess}(d)$.

these quantities resulting in an invalid confidence interval. Instead we need to choose k_L and k_U carefully so that $M_{k_L,k_U}(d) = [A_{k_L}^{\varepsilon/2}(d) - \theta, A_{k_U}^{\varepsilon/2}(d) + \theta]$ is a valid confidence interval. In order to analyse the coverage, we return to Equation (1),

$$\begin{aligned}\Pr(\mathtt{med}(P) < A_{k_L}^{\varepsilon/2}(d) - \theta) &= \sum_{m=0}^{n} \Pr(\mathrm{Bin}(n,1/2) = m) \cdot \Pr(\mathtt{med}(P) < A_{k_L}^{\varepsilon/2}(d) - \theta \mid \mathtt{rank}_d(\mathtt{med}(P)) = m) \\ &\leq C_{\mathrm{Bin}}(k_L - 1) + \sum_{m=k_L}^{n} \Pr(\mathrm{Bin}(n,1/2) = m) \cdot \Pr(|\mathtt{rank}_d(A_{k_L}^{\varepsilon/2}(d) - \theta) - k_L| \geq m - k_L)\end{aligned} \tag{2}$$

In this computation, we use the inequality $\Pr(\mathtt{med}(P) \leq A_{k_L}^{\varepsilon/2}(d) - \theta \mid \mathtt{rank}_d(\mathtt{med}(P)) = m) \leq 1$ for all $m < k_L$. When $m < k_L$, $\mathtt{med}(P) < d_{(k_L)}$ so $\mathtt{med}(P) < A_{k_L}^{\varepsilon/2}(d) - \theta$ occurs with high probability if $A_{k_L}^{\varepsilon/2}$ outputs something close to $d_{(k_L)}$ (which is the goal of this algorithm) with high probability. Some additional performance could be obtained by tighter analysis of these terms, but we expect the improvement to be small. Now the first term is only due to sampling error and easily bounded. The second term depends on both the sampling error and the error rate of the private algorithm $A_{k_L}^{\varepsilon/2}$. In Appendix B, we show how to obtain a distribution independent upper bound on the error rate $\Pr(|\mathtt{rank}_d(A_{k_L}^{\varepsilon/2}(d) - \theta) - k_L| \geq m)$. This allows us to give an upper bound for $\Pr(\mathtt{med}(P) \leq A_{k_L}^{\varepsilon/2}(d) - \theta)$ which holds for any distribution $P \in \Delta_{\mathscr{C}}(\mathbb{R})$. Given this coverage analysis, we can search for k_L and k_U that maximise $\Pr(\mathtt{med}(P) \notin M_{k_L,k_U}(d))$ subject to the constraint that $\Pr(\mathtt{med}(P) \notin M_{k_L,k_U}(d)) \leq \alpha$.

We will refer to this algorithm as `ExpMech`. Pseudo-code and details of the analysis can be found in online supplement B. A graph representing the distribution of `ExpMech` on a single data set can be found in Figure 3.

4.2 Confidence intervals based on CDF estimator, `CDFPostProcess`

Our second CDP confidence interval `CDFPostProcess` is obtained from post-processing the output of a CDP cumulative distribution function (CDF) estimator. Unlike `ExpMech`, which only released the confidence interval, `CDFPostProcess` can additionally release a CDP estimate of the CDF without consuming additional privacy budget. There has been considerable work in the DP literature on DP cumulative distribution function (CDF) estimators, both for the parametric and non-parametric models [Diakonikolas et al., 2015, Brunel and Avella-Medina, 2020]. We will focus on a particular CDF estimator based on the tree-based mechanism introduced in Li et al. [2010], Dwork et al. [2010], and Chan et al. [2011]. This mechanism was further refined in Honaker [2015], whose algorithm we base our mechanism on.

Given a range for the data $[r_\ell, r_u]$, and a discretization of this range $\mathscr{R}_{\text{discrete}} = [r_\ell, r_\ell + \theta, r_\ell + 2\theta, \cdots, r_u]$, the algorithm in Honaker [2015] outputs a DP estimate, $\tilde{C}$, of the empirical CDF, $\hat{C}$, of a data set restricted to $\mathscr{R}_{\text{discrete}}$. The relevant feature of this estimator for our analysis is that we understand the marginal distribution of $\hat{C}(x)$ for each $x \in \mathscr{R}_{\text{discrete}}$. Specifically, for all $x \in \mathscr{R}_{\text{discrete}}$, there exists $\sigma_x > 0$ such that the value we release is equal to the empirical CDF with normally distributed noise, that is: $\tilde{C}(x) = \hat{C}(x) + \mathscr{N}(0, \sigma_x^2)$. The noise values are not independent across different values of x, nor are they perfectly correlated. The estimates $\tilde{C}(x)$ don't even increase monotonically with x (see Figure 4). Thus, we seek a procedure that uses only our knowledge of the marginal distribution. The main observation is that we can rewrite the distribution of $\tilde{C}(x)$ in terms of the true distribution CDF value, $C(x)$:

$$\tilde{C}(x) = \tfrac{1}{n} \cdot \text{Bin}(n, C(x)) + \mathscr{N}(0, \sigma_x^2).$$

We will use this, for each value x in $\mathscr{R}_{\text{discrete}}$, to test the hypothesis that x is less than the median. Observe that $x < \texttt{med}(P)$ if and only if $C(x) < \frac{1}{2}$. Thus, if $x < \texttt{med}(P)$, then $\tilde{C}(x)$ is stochastically dominated by the distribution $\frac{1}{n} \cdot \text{Bin}\big(\big(n, \frac{1}{2}\big) + \mathscr{N}(0, \sigma_x^2)$. We have

$$Pr(\tilde{C}(x) > a) \leq \sum_{m=0}^{n} \Pr\big(\text{Bin}(n, 1/2) = m\big) \Pr\big(\tfrac{m}{n} + \mathscr{N}(0, \sigma_x^2) > a\big),$$

which we can easily numerically evaluate. Thus, for each x, there exists a value a_x (that just depends on σ_x) such that if $x < \texttt{med}(P)$ then $\Pr(\tilde{C}(x) > a_x) \leq \alpha/2$, where $1 - \alpha$ is our desired coverage. Equivalently, if $\tilde{C}(x) > a_x$ then we can be confident that $x \geq \texttt{med}(P)$. The upper end of our confidence interval is then defined as the grid point just to the right of the largest value x for which the test accepts, that is,

$$\begin{aligned}
\widetilde{\texttt{ci}}_U^{\alpha}(d) &= \theta + \max\{x \in \mathscr{R}_{\text{discrete}} \ : \ \text{test for } x < \texttt{med}(P) \text{ accepts}\} \\
&= \theta + \max\big\{x \in \mathscr{R}_{\text{discrete}} \ : \ \tilde{C}(x) \leq a_x\big\} \\
&= \min\big\{x \in \mathscr{R}_{\text{discrete}} \ : \ (\forall x' \geq x)\ \tilde{C}(x') > a_{x'}\big\}.
\end{aligned}$$

To see why this leads to a valid confidence interval, let x^* be the largest value in $\mathscr{R}_{\text{discrete}}$ that is less than $\texttt{med}(P)$. With probability at least $1 - \frac{\alpha}{2}$, the test at x^* accepts. In that case, $\widetilde{\texttt{ci}}_U^{\alpha}(d)$ will be a grid point greater than x^*, and thus we will have $\widetilde{\texttt{ci}}_U^{\alpha}(d) \geq \texttt{med}(P)$.

The process and reasoning for the left end of the interval are symmetric. Pseudo-code and a proof that this process gives valid confidence intervals can be found in Appendix D. We will refer to this algorithm as `CDFPostProcess`. A graph representing a single run of this algorithm can be found in Figure 4. In particular, note that the horizontal pink dotted lines represent the values of a_x, which are closer to $\texttt{N}_L^{\alpha}$ and $\texttt{N}_U^{\alpha}$ when σ_x is small. [4]

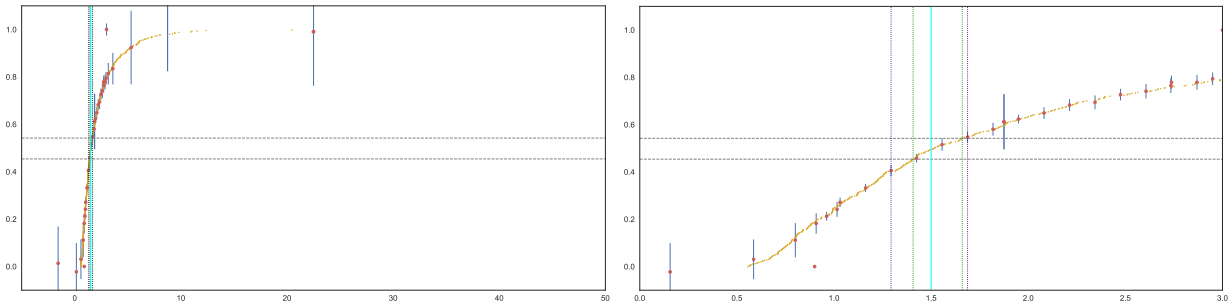
4.3 Confidence intervals based on noisy binary search, `NoisyBinSearch`

One draw-back of the CDF-based estimator `CDFPostProcess` is that it spends its privacy budget roughly evenly across the entire range $\mathscr{R}$. This can result in substantially reduced performance if the data is concentrated in a small subset of the range. We can see this effect in Figure 4 where the values of the $\tilde{C}$ below $x = 0$ are very unlikely to impact

[4] Of potential independent interest is our efficient implementation for computing the pointwise error rates σ_x. While Honaker discusses computing this error for a single x, we developed and implemented an efficient algorithm for computing this error for all x. This has potential for impact beyond confidence intervals for the median.

the confidence interval. `NoisyBinSearch` attempts to locate the data within the region $\mathscr{R}$ using a small amount of the privacy budget then delves more deeply into the region actually containing the data. This defines a CDP confidence interval in its own right but we will see the real power of it in the design of the next algorithm.

`NoisyBinSearch` uses noisy queries $\hat{C}(x)+\mathscr{N}(0,\sigma^2)$ to the empirical CDF to search for the relevant quantiles using binary search. Unlike `CDFPostProcess` that obtains a CDP estimate to $\hat{C}$ over the entire $\mathscr{R}_{\text{discrete}}$, `NoisyBinSearch` minimises the number of values we need to evaluate $\hat{C}$ on. Full details and pseudo-code are given in online supplement C. As in the design of `ExpMech`, this algorithm starts with two target quantiles k_L and k_U which are defined to ensure that the resulting confidence interval has the desired coverage. We will discuss this choice in online supplement C. Let $\hat{C}$ be the empirical CDF. In the non-private setting, we can find an approximation to $d_{(k_L)}$ by iteratively asking queries of the form "is $\hat{C}(x) \leq k_L$?" and adjusting our search accordingly. This search method is called binary search and minimises the number of values we need to evaluate $\hat{C}$ on. We can perform a CDP version of binary search using noisy queries of the form "is $\hat{C}(x)+\mathscr{N}(0,\sigma^2) \leq k_L$?" In setting the variance σ we need to balance the desire for the noisy query to answer correctly (so the binary search moves in the right direction) and adding enough noise to guarantee privacy. Notice that if $|\hat{C}(x)-k_L|$ is large, for example if x is inside $\mathscr{R}$ but far from the concentration of the data, then σ can be chosen to be quite large while still ensuring the response is correct with high probability. A single query of this form is $\frac{1}{n\sigma}$-CDP [Bun and Steinke, 2016] so if $|\hat{C}(x)-k_L|$ is large, then we only need to consume a small amount of privacy budget in order to ensure the binary search moves in the right direction. The privacy guarantee for the entire algorithm is sum of the privacy guarantees for each iterate. Since a priori we don't know $|\hat{C}(x)-k_L|$, at each iteration we'll start with a quite noisy query, then keep decreasing the noise until we are confident which direction to move in. We repeat until the privacy budget is consumed. We search for both the upper and lower end points of the confidence interval using noisy binary search then a final post-processing step on the noisy measurements ensures we release a valid confidence interval. Full details and pseudo-code are given in online supplement C. At the points x_i at which `NoisyBinSearch` obtains noisy measurements, the noisy measurement $\hat{C}(x_i)+\mathscr{N}(0,\sigma^2)$ can be released in addition to the confidence interval without consuming additional privacy budget. While this is not as informative as the full CDP CDF released using `CDFPostProcess`, it does provide useful additional information about the distribution. We note that unlike `ExpMech` and `CDFPostProcess`, our analysis of `NoisyBinSearch` separates the two sources of randomness coming from sampling and noise added for privacy. An interesting open problem is whether this analysis can be improved by considering the relationship between the two sources of randomness.



(a) (b)

Figure 5: Graphical representation of a single run of 0.8-CDP 95%-confidence interval using `BinSearch`$+$`CDF` (with hyperparameters $\mathscr{R}=[-5,50]$, $\theta=0.1$) on a single dataset. The 500 datapoints are sampled from lognormal$(\ln(1.5),1.0)$. The yellow dots represent the empirical CDF, the gray horizontal dashed lines correspond to $\mathtt{N}_L^{0.05}/n, \mathtt{N}_U^{0.05}/n$, and the green vertical dashed lines correspond to $\mathtt{ci}_L^{0.05}, \mathtt{ci}_U^{0.05}(d)$. The cyan line is the population median. The red dots correspond to DP estimates, with 99% error bars in blue. Figure (a) shows how `NoisyBinSearch` is used to narrow down the range from [-5, 50] to roughly [0.8,3], and Figure (b) zooms in to display the estimates made by `CDFPostProcess` within this smaller range. The vertical purple dashed lines represent the DP interval that `BinSearch`$+$`CDF` outputs.

4.4 Range-robust estimator based on CDF estimator, BinSearch + CDF

The mechanism NoisyBinSearch was designed to improve the performance of CDFPostProcess when the data is concentrated in a small region within $\mathscr{R}$. That is, when NoisyBinSearch can greatly reduce the search range for the median using a small amount of privacy budget. However, when the data is well spread out within the range, we do not expect NoisyBinSearch to perform as well as CDFPostProcess which is highly optimised to provide an accurate approximation to the CDF, C. Our final algorithm, BinSearch + CDF first uses NoisyBinSearch with a portion of its privacy budget to narrow down the search space $\mathscr{R}$. It then clips the data to this range, and runs CDFPostProcess with the remaining privacy budget within this smaller range to obtain a confidence interval. Full details and validity proofs are given in online supplement E.

The privacy budget and coverage α both need to be partitioned between the two stages of the algorithm. We expect the optimal split to be distribution dependent. In particular, it likely depends on how large a region the data occupies within the range $\mathscr{R}$. We found experimentally, for the parameter regimes we studied, using $\rho/4$ for the first step, and $3\rho/4$ for the second step seemed to be a good choice. Similarly, we ensure that the region found in the first setp contains the median with probability $1-\alpha/4$, and the second step finds a $1-3\alpha/4$-confidence interval within that region.A representation of a single run of BinSearch + CDF is shown in Figure 5. Notice that relative to CDFPostProcess in Figure 4, it uses only a few measurements to narrow into the range of interest, about $[0.8, 3]$ then takes more measurements within this range.

4.5 A note on hyperparameters

All our algorithms require some hyperparameter tuning and domain knowledge. Throughout this manuscript we will attempt to give guidance on how one may set these parameters and how sensitive the performance is to these choices.

All our algorithms require as input a range space for the median. That is, an interval $\mathscr{R} \subset \mathbb{R}$ that is promised to contain $\texttt{med}(P)$. This should be chosen as small as is reasonable, but one can typically be quite conservative when defining $\mathscr{R}$. We expect the dependence of all algorithms to be approximately $\log|\mathscr{R}|$, we see this explicitly for ExpMech in Lemma B.2 in the online supplement. We will empirically explore the dependence on $|\mathscr{R}|$ in Figure 6d below. Note that requiring a range on the median is different to requiring a range on the data points themselves. It is a preferable condition since the data points may occupy a considerably larger range than $\mathscr{R}$, and, in practice, guaranteeing a bound on outliers in the data may be difficult. The fact that a bound on the median suffices comes from the fact that if $\mathscr{R}$ contains the median then projecting the data into $\mathscr{R}$ leaves the median unchanged.

All the algorithms we consider require an additional hyperparameter we refer to as the "granularity" parameter, θ. In order to gain some intuition, one can imagine each algorithm as discretizing the range $\mathscr{R}$ so that any two potential output points are θ apart. This intuition is not exact, please refer to the online supplement for more details. This granularity parameter affects each algorithm differently. ExpMech is not very sensitive to this parameter in general, and it can typically be taken to be very small. In fact, setting this parameter to 0 is a reasonable choice in a wide variety of parameter settings. CDFPostProcess and BinSearch + CDF can be sensitive to this parameter. In all algorithms, the width of the confidence interval will be at least θ.

5 Simulation Studies

In this section we present extensive simulation studies to evaluate the different algorithms under various parameter settings.[5] We focus on log-normal data, as this is the natural use-case for computing a non-private median and corresponding confidence interval. We expect many of our findings to extend to other types of skewed data. We evaluate the performance of the four algorithms described in Section 4, as well as the non-private confidence interval described in Lemma 2.3, in terms of width of confidence interval, coverage, and bias.

[5] Code for producing these simulations can be found at https://github.com/anonymous-conf-medians/dp-medians.

5.1 Data description

In order to visualize the distribution of the noisy confidence intervals, we run each private algorithm 5 times on 100 independently drawn datasets. Let $\mathbf{x}_1, \ldots, \mathbf{x}_{100}$, each contain $n = 1000$ i.i.d. draws from the underlying log-normal distribution. The underlying normal random variable has mean $\mu = \ln(1.5)$ and standard deviation of either $\sigma = 1$ or $\sigma = 5$. We run each DP algorithm on each $\mathbf{x}_i$ for 5 trials. In our experiments, we show the relative performance of the algorithms as we vary n, ρ, σ, and $|\mathscr{R}|$.

5.2 Utility measures

We consider two main utility measures in our experiments. The first measure is the relative width of the CDP confidence interval $M(d)$ compared to the width of the non-private confidence interval, $[\texttt{ci}_L^\alpha, \texttt{ci}_U^\alpha]$. For a data set $d \in \mathscr{D}^n$, $\alpha \in [0,1]$, and interval $I = [I_L, I_U]$, the relative width is defined as

$$\text{rel-width}^\alpha(d, I) = \frac{I_U - I_L}{\texttt{ci}_U^\alpha(d) - \texttt{ci}_L^\alpha(d)}$$

We are concerned with the distribution of rel-width$^\alpha(d, I)$ when $I = M(d)$. We expect the private confidence intervals to be wider than the non-private confidence intervals, so if $I = M(d)$ then we expect rel-width$^\alpha(d) \geq 1$ with high probability. If rel-width$^\alpha(d) \leq 2$, then, intuitively, the additional uncertainty due to privacy is less than the uncertainty due to sampling. We are interested in the distribution of the relative width over multiple trials, so for each algorithm we show box plots of this metric over 500 trials (100 datasets times 5 trials of the DP mechanism on each).

The second utility measure is *empirical* coverage of the DP confidence interval. For intervals $I_1, \cdots, I_T$ and distribution P, let

$$\text{cov}_T(P, I_1, \cdots, I_T) = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}_{\texttt{med}(P) \in I_t}.$$

Given $n \in \mathbb{N}$ and a $1-\alpha$-confidence interval M, for all $t \in [T]$, let $d_t \sim P^n$ and $I_t = M(d_t)$ then

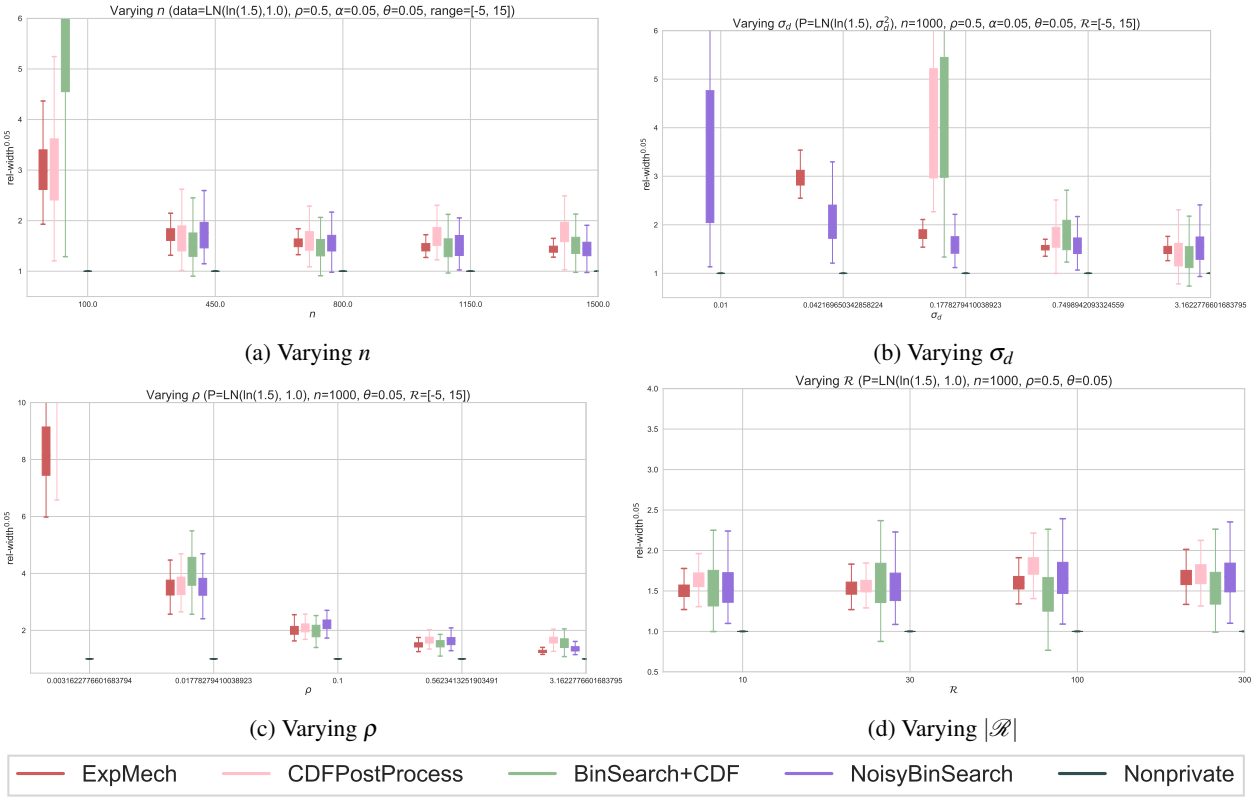
$$\text{cov}_{T,n,P}(M) = \text{cov}_T(P, I_1, \cdots, I_T)$$

gives an estimate of the empirical coverage of M on the distribution P. We estimate the coverage over 5,000 trials (1,000 drawn samples of size $n = 1{,}000$ times 5 trials of the DP mechanisms on each). A key component of the confidence intervals presented in the previous section is that the coverage should be at least $1-\alpha$. However, the empirical coverage may, and in many settings will, exceed $1-\alpha$.

5.3 Results and Discussion

Comparison Among Algorithms Figure 6 demonstrates the performance of our four CDP confidence interval algorithms across a range of parameter regimes on log-normal data, in terms of the relative width metric. Notice that in a variety of regimes, including large n, large ρ and large σ_d all of the CDP algorithms provide confidence intervals that are at most twice the width of the non-private confidence interval with high probability. Our results indicate that `ExpMech` provides the tightest, or close to the tightest, confidence intervals in most parameter regimes we studied. This algorithm is the most targeted of the CDP algorithms we discuss and is carefully calibrated to not waste privacy budget on estimating additional information about the underlying distribution. It is a good general choice when one is solely interested in confidence intervals for the median. There are a few regimes in which the other algorithms outperform `ExpMech` which we will discuss in this section.

The `CDFPostProcess` algorithm is appealing in practice since it allows a CDP estimate of the CDF to be released without consuming additional privacy budget. This can be used not only to produce confidence intervals for the median, but to produce a wealth of other insights about the distribution P. Surprisingly, in a variety of parameter regimes, `CDFPostProcess` provides confidence intervals that are almost as tight as those obtained by `ExpMech`. In fact, when σ_d is large, `CDFPostProcess` can result in tighter confidence intervals than `ExpMech` (Figure 6b). We explore this further in Appendix G. Conversely, when σ_d is small, or $|\mathscr{R}|$ is large, `CDFPostProcess` is not a good



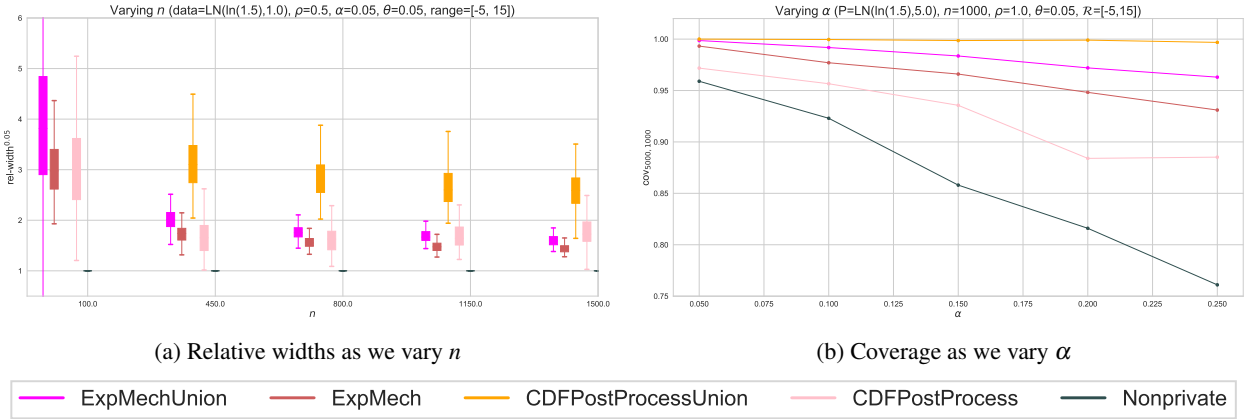
(a) Varying n

(b) Varying σ_d

(c) Varying ρ

(d) Varying $|\mathscr{R}|$

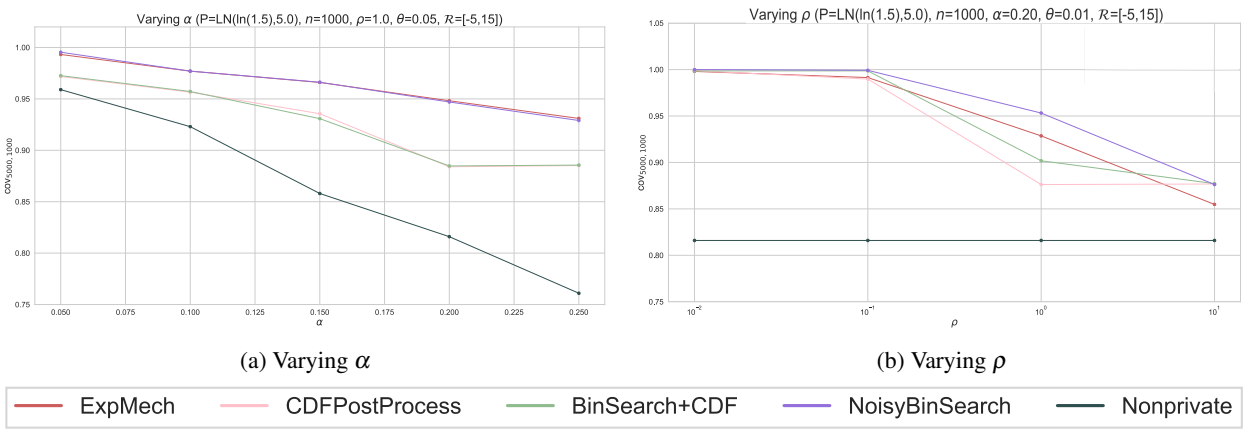
Figure 6: Relative widths of DP confidence intervals as we vary (a) dataset size n, (b) dataset standard deviation σ_d, (c) privacy parameter ρ, and (d) size of range $|\mathscr{R}|$ on log-normal data. By definition, rel-width$^{\alpha}(d, I) = 1$ when $I = [\texttt{ci}_L^{\alpha}, \texttt{ci}_U^{\alpha}]$.



(a) Relative widths as we vary n

(b) Coverage as we vary α

Figure 7: Performance of naive vs. more careful DP confidence intervals



(a) Varying α (b) Varying ρ

Figure 8: Coverage of DP confidence intervals as we vary α and ρ

choice. As discussed earlier, these are regimes where `CDFPostProcess` spends a lot of its privacy budget estimating the CDF in regions that are far from the median.

The remaining CDP confidence intervals `BinSearch` + `CDF` and `NoisyBinSearch` were designed to outperform `CDFPostProcess` when the data is concentrated in a small subset of $\mathscr{R}$. This setting is shown in Figure 6b, when the data is very concentrated and Figure 6d when $|\mathscr{R}|$ is large. In both settings we see either `BinSearch` + `CDF` or `NoisyBinSearch` outperforming the other CDP algorithms. `BinSearch` + `CDF` involves a hyperparameter that decides how much of the privacy and coverage budget is spend on each step of the algorithm. In our experiments, we used one quarter of the privacy budget to perform the range-finding step (using `NoisyBinSearch`), and three quarters to find the confidence interval within that region (using `CDFPostProcess`). This choice was made experimentally by testing multiple choices for this partition. Since this parameter interpolates between `CDFPostProcess` and `NoisyBinSearch`, the optimal choice appears to be data dependent. In particular, we conjecture that the smaller the fraction of the range that is occupied by the data, the higher the fraction of the budget that should be allocated to the `NoisyBinSearch` step. Of course, one typically does not know a priori how concentrated the data is, so the (1/4, 3/4) seems to be a reasonably safe split that performs well in a variety of contexts.

Empirical coverage analysis. A key component of the algorithmic design of each of the CDP confidence intervals was the coverage analysis. We discussed in Section 3.2 how a careful coverage analysis that leverages the relationship between the two sources of the randomness in the CDP confidence intervals potentially results in a much tighter coverage analysis than the naive analysis that separates the sources of randomness. Our experimental results presented in Figure 7 highlight two key findings regarding the coverage; that the careful analysis does result in substantially tighter intervals, and that the empirical coverage of the CDP confidence intervals is still notably above the target coverage.

Figure 7a compares the relative width of the different confidence intervals. `ExpMechUnion` and `CDFPostProcessUnion` refer to the versions of `ExpMech` and `CDFPostProcess` resulting from the naive coverage analysis. While the relative width is only slightly reduced for `ExpMech`, the relative width of `CDFPostProcess` is almost halved if the improved approach is used to produce the confidence intervals. These findings are also reflected in Figure 7b, which compares the empirical coverage of the naive coverage analyses of `ExpMechUnion` and `CDFPostProcessUnion` and the more careful analyses described in Section 4. While theoretically we can show that the careful analysis will result in empirical coverage that is much closer to the target coverage, Figure 7b shows that this improvement is practically relevant. While the improved analysis only leads to modest reduction in the overcoverage for `ExpMech`, the changes for `CDFPostProcess` are more substantial. The naive approach results in coverage rates that are close to 1 irrespective of the selected α. The improved approach leads to coverage rates that are much closer to the nominal coverage rates. This highlights the importance of designing algorithms that consider the relationship between the two sources of randomness.

Despite the substantial improvement, Figure 8a shows that all the CDP algorithms exhibit empirical coverage higher than the target coverage for moderate values for ρ. As expected, Figure 8b reveals that the coverage improves with ρ as each algorithm trends towards outputting the non-private confidence interval $[\texttt{ci}_L^\alpha, \texttt{ci}_U^\alpha]$ when $\rho = \infty$. Over-coverage does not necessary correspond to substantially larger confidence intervals. We see in Figure 6 that in a wide range of parameter regimes our CDP algorithms still result in confidence intervals that are at most twice as wide as their non-private counterparts. However, it does suggest an opportunity for improvement. An important question for future work is to what degree this over-coverage is necessary? In particular, is there an inherent tension between privacy guarantee, and learning enough about the data set to accurately quantify the uncertainty?

In many estimation tasks defining a non-parametric confidence interval that gives close to nominal coverage rates is difficult. Without information regarding the underlying distribution, the confidence intervals need to be wide enough to ensure valid coverage rates for any possible distribution. This can result in the non-parametric confidence intervals having higher than expected empirical coverage when the data is drawn from a nice distribution, e.g a log-normal distribution. This effect is one possible explanation for the fact that the CDP confidence intervals have empirical coverage higher than $1-\alpha$ in Figure 8. In fact, we see evidence of this in the analysis of `ExpMech`. The error of the exponential mechanism is data dependent (and hence distribution dependent), but in our coverage analysis we are forced to use the worst case error of the exponential mechanism over all datasets.

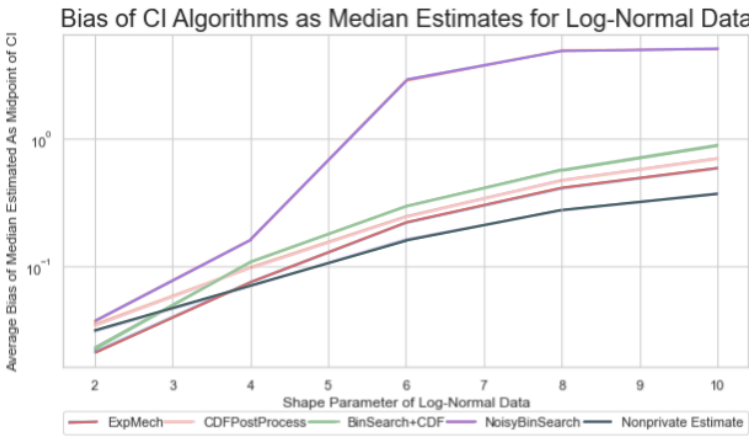


Figure 9: Bias of the algorithms (average of the difference between the value obtained and the true median) for log-normal data centered at 1 as the shape parameter of the distribution (the variance of the normal distribution that is exponentiated) increases from 2 to 10.

Bias. The goal of this work was to design algorithms that output valid confidence intervals for the median, not to estimate the median itself. An ad-hoc estimate of the median can be obtained from a confidence interval by taking the estimate to be the mid-point of the interval. This approach is preferable in the differential privacy context since it allocates its entire budget to the object of interest (the confidence interval) and we discussed in Section 3.1 some of the reasons why direct estimators of the median are difficult to generalise to CDP confidence interval algorithms. For all of our CDP algorithms, as well as the non-private confidence interval, this results in a biased estimator for the median, if the underlying distribution is skewed. In Figure 9, we explore the bias of the inherited median estimators. As expected, the bias increases with the skew of the data. The bias of most of the DP algorithms (except for `NoisyBinSearch`) is not substantially different from the bias for the non-private estimate. This implies that most of the bias can be attributed to the ad-hoc strategy of using the mid-point of the confidence interval as the point estimate for the median. As mentioned earlier, one benefit of `CDFPostProcess`, `NoisyBinSearch` and `BinSearch` + `CDF` is that they come with additional information about the distribution which could potentially be used to release a less biased estimate of the median. We leave this for future work.

6 Real data application

Characteristic of household(er)	Number in 1% sample	CDP median income	DP 90% CI	Non-private median income	Non-private 90 % CI	Population median income
Type of Household						
Family households	9,142	489.00	(469.99, 508.01)	499.97	(480.01, 500.93)	499.95
Nonfamily households	1,479	65.50	(0.0*, 136.01)	20.12	(0.17, 99.89)	0.20
Metropolitan status						
Not in metropolitan area	8243	324.99	(290.03, 359.95)	329.85	(300.06, 359.85)	360.07
In metropolitan area	2380	708.12	(640.00, 776.23)	699.99	(659.94, 749.85)	699.91
Age						
Age < 65	9,259	564.89	(529.94, 599.84)	560.05	(540.03, 597.95)	540.10
Age ≥ 65	1,366	0.00	(0.0*, 5.0)	0.03	(0.02, 0.03)	0.03

Table 1: Income summary measures by selected characteristics based on 1% simple random sample of mountain division householder records (i.e., in Arizona, Colorado, Idaho, Montana, Nevada, New Mexico, Utah, Wyoming) from the 1940 decennial census. Income is shown in 1940s dollars and is top-coded at \$5001. Differentially private estimates are obtained using `ExpMech` on a single sample with total privacy budget $\rho = 0.5$, range $\mathscr{R} = [0, 5001]$ and granularity $\theta = 5$. Note that the lower DP confidence interval values with a * are truncated to 0 based on the assumption that incomes are nonnegative. The DP point estimates are computed before the truncation to avoid introducing bias. All values are rounded to the nearest cent.

In this section, we illustrate how the findings from the previous sections could inform the implementation of a differentially private median release strategy in practice. We also demonstrate what level of accuracy one could reasonably expect for realistic applications. Our motivating example is the median income tables published by the U.S. Census Bureau for various subgroups of the population. Specifically, we aim to replicate a subset of statistics from Table *A1.Income Summary Measures by Selected Characteristics: 2018 and 2019* [Semega et al., 2020]. This table reports median household income broken down by the following characteristics: Type of household, Race and Hispanic Origin of Householder, Age of Householder, Nativity of Householder, Region, and Residence. For each of the 32 subgroups specified, the table provides the estimated median income and estimated margin of error (based on $\alpha = 0.1$) for 2018 and 2019. The estimates are computed using the Current Population Survey, 2019 and 2020 Annual Social and Economic Supplements (CPS ASEC).

Since we want to assess the accuracy of the CDP estimates, we use income data from the 1940 Decennial Census [Ruggles et al., 2021], which enables us to compare the noisy estimates to the true values in the population. We restrict the population data to heads of households in the mountain division region (Arizona, Colorado, Idaho, Montana, Nevada, New Mexico, Utah, and Wyoming) and focus on the variables type of household (two categories), metropolitan area (two categories) and age (two categories). To mimic the illustrative application described above, we repeatedly sample from this population and treat the resulting data as the survey from which the (noisy) estimated medians will be computed. For simplicity, we draw 1% simple random samples without replacement. We acknowledge that the sampling design for the CPS ASEC is far more complex. However, as indicated in the introduction understanding the subtle effects of complex sampling designs on the privacy guarantees is currently an area of active research and is beyond the scope of this paper.

The variable we use for our evaluations is 'INCWAGE,' which "reports each respondent's total pre-tax wage and salary income for the previous year." The amounts are displayed in "contemporary dollars," which means they are not adjusted for inflation. In the 1940's dataset, the variable is topcoded at $5,001$ dollars. We remove all N/A and missing values from the dataset, and only consider records corresponding to the head of each household. We note that we do not propose simply dropping all cases with missing values in practice as this will likely introduce bias. However, properly integrating any non-response adjustments into the DP algorithms is beyond the scope of the paper. Thus, we treat the fully observed data on household heads as our population of interest. Finally, for the purpose of error analysis

we treat the empirical distribution of the entire population dataset (from which we sample 1%) as the true underlying distribution P.[6]

6.1 Selecting the algorithm and (hyper)parameters

To generate the privatized confidence intervals, we use the algorithm identified as the winner in a wide range of regimes in the simulation studies: `ExpMech`. The hyperparameters are set to $\mathscr{R} = [0, 5001]$, and $\theta = 5$. The lower and upper bounds are chosen based on the assumption that the threshold used for top coding is public knowledge and the reasonable assumption that the median income will not be less than zero. The granularity parameter θ is chosen based on Census Bureau data visualizations that report median incomes from 1967 to present, which are rounded to the nearest \$100 [U.S. Census Bureau, 2020b] indicating that a granularity of \$5 for 1940 median incomes is likely sufficient for data users. We split the overall privacy budget of $\rho = 0.5$ equally across three characteristics: type of household, metropolitan status, and age.

6.2 Results

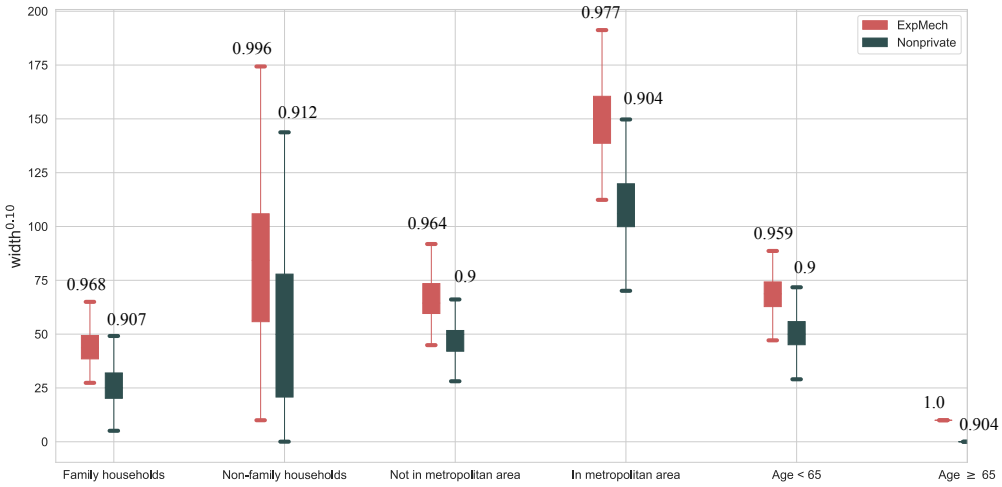


Figure 10: Comparing widths of 90% `ExpMech` and non-private confidence intervals. Algorithms are run on 1,000 samples of income data by selected characteristics from the 1940 Decennial Census (the DP algorithm is run 20 times for each sample). Empirical coverage rates are displayed for each algorithm.

Results based on the first simulation run are included in Table 1. Note that the CDP confidence intervals and median estimate are the result of a single run of `ExpMech` so this table is indicative of what we would expect in practice. The CDP point estimates for the median incomes are chosen as the midpoint of the corresponding CDP confidence intervals. Note that we could also leverage a prior assumption of the right-skewness of income data by choosing the CDP point estimator from the left half of the CDP confidence interval, rather than from its center, but we leave this type of parametric estimation to future work. We leverage the assumption that the incomes are non-negative, so we set the lower endpoint of the CDP confidence intervals at the maximum of the output of the algorithm and 0. However, we compute the point estimates before the truncation step to avoid introducing bias. The table also provides non-private and private 90% confidence intervals (the margin of error reported in the Census tables could be computed as the half-width of these intervals).

[6]Note that many respondents report incomes rounded to the nearest \$5, \$10 or \$50 dollars, which results in substantial overcoverage even for the non-private confidence intervals due to the spikes in the data. Therefore, we add a negligible amount of noise ($\mathcal{N}(0, 0.01)$) to each population income data point, so that the distribution being sampled from is continuous. See Appendix A for further discussion.

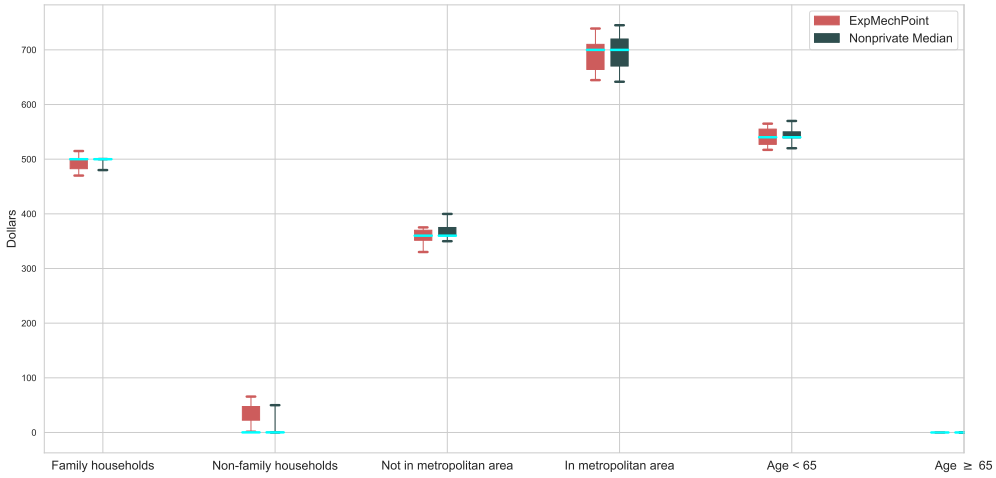


Figure 11: Comparing point estimates of `ExpMechPoint` and non-private medians. Algorithms are run on 1,000 samples of income data by selected characteristics from the 1940 Decennial Census (the DP algorithm is run 20 times for each sample). Population medians by characteristic are denoted by the cyan lines. The whiskers of the boxplots denote the 5th and 95th quantiles of the estimates.

The non-private median estimates are closer to the true values than the DP estimates for all sub-populations. However, for many statistics the difference between the point estimates is small relative to the width of the confidence interval indicating that the bias introduced by the ad-hoc approach of using the center of the confidence interval as the point estimator for the median is only minor. Except for the householders aged 65 and above, the relative increase in uncertainty also seems to be acceptable. The relative increase in the length of the confidence intervals ranges between 20.7% and 81.7%, that is, the uncertainty from data protection is always less than the uncertainty from sampling. The large relative increase of the confidence intervals for householders aged 65 and above can be explained if we note that the width of the CDP intervals is lower bounded by the granularity hyperparameter (which we chose to be $\theta = 5$), which leads to a large relative uncertainty if the non-private interval has width close to 0. However, the absolute increase in uncertainty is still acceptable.

In Figure 10 and Figure 11 we explore the performance of `ExpMech` and the non-private algorithm over 1,000 randomly sampled datasets. Figure 10, which contains boxplots showing the width of the private and non-private confidence intervals, confirms the findings based on one simulation run. While the private confidence intervals are typically wider than the non-private intervals, the increase in width is less than a multiplicative factor of two for all sub-populations except for head of households older than 65, and less than $100 in all sub-populations. The figure also reports the coverage rates for the non-private and private confidence intervals. The coverage rates are computed over 1,000 simulation runs and 20 trails of the CDP algorithm within each simulation run. While the non-private coverage rates are close to the nominal 90% coverage, the CDP confidence intervals overcover substantially. The coverage rates vary between 95.9% and 100%. The results are in line with the simulation studies.

Figure 11 contains boxplots showing the variability of the private and non-private point estimates (where `ExpMechPoint` is the exponential mechanism point estimator). The whiskers of the boxplots indicate the 5th and 95th quantile of the empirical estimates to ensure consistency with the 90% confidence intervals reported in Figure 10. The true medians from the population are indicated by cyan lines for each of the sub-populations.

We find that from an inferential perspective the difference between the private and non-private estimates is small. For most of the estimates, the range of the boxplots overlap to a large extent and similar to Table 1 the bias is small relative to the variability in the estimates. The only estimate for which we find noticeable bias is the private estimate for non-family households. The bias arises because of the large fraction of zeros among this sub-population in the original data. Since 51% of the records in the original data report an income that is essentially zero (except for the small amount of noise that we introduce to make our data approximately continuous), the sample median will also be close to zero in many simulation runs. However, the CDP point estimate is based on the midpoint of the CDP

confidence interval and the upper limit of this confidence interval will almost always be larger than the 51st quantile in the population, that is, it will almost always be larger than zero. Thus, the estimated CDP median will be biased. This highlights that the midpoint strategy can run into problems if the data is highly concentrated in certain areas of the distribution. However, such a scenario does not necessarily introduce substantial bias. This can be seen for the head of households that are 65 years or older. For this sub-population, the percentage of householders with zero reported income is about 81%. As a result, the upper limit of the CDP confidence interval is still close to zero in most simulation runs and the point estimate remains almost unbiased.

7 Conclusion

Measuring the uncertainty in differentially private estimates is a challenging task, especially if the input data is a sample from a larger population. In this case, both sources of randomness—the sampling error as well as the error from the CDP algorithm—need to be taken into account. If the mechanism is data dependent, the two sources are no longer independent making it difficult to quantify the uncertainty in the final output.

In this paper we addressed this challenge for the median, evaluating several strategies to obtain differentially private confidence intervals for this commonly used statistic. All the algorithms proposed produced valid non-parametric confidence intervals. We also demonstrated that directly accounting for both sources of uncertainty simultaneously allowed us to give tighter confidence bounds than relying on naive approaches that account for the two components sequentially. Our simulation results showed that an algorithm we called `ExpMech` produced reliable and consistent confidence intervals which were less than twice the width of the non-private confidence intervals in a wide variety of parameter regimes. A pair of algorithms called `CDFPostProcess` and `BinSearch` + `CDF` provide confidence intervals that are almost as tight, or slightly tighter, than `ExpMech` in a variety of regimes. These algorithms are practically appealing since they release a wealth of additional information about the distribution P without consuming additional privacy budget.

The private confidence intervals in the application based on the 1940 Decennial Census were not substantially wider than the confidence intervals released by the non-private algorithm, illustrating that the extra uncertainty due to data protection can be small in practice. It should be noted that the total privacy budget always needs to be divided among all the characteristics of interest (type of household, metropolitan status and age in our application) so the accuracy will necessarily decrease if more statistics are to be released under the same privacy budget.

We also found that the bias introduced by the ad-hoc strategy of using the midpoint of the confidence interval as an estimate for the median was limited for most estimates in our real data application. One strategy to further reduce this bias would be to use available information regarding the skewness of the data to come up with a better point estimate for the median. The fact that most of the algorithms provide additional information regarding the CDF of the data could be helpful for this endeavour as the information could be exploited in a post-processing step to model the distribution of the data. We leave this for future research.

We saw in our experiments on both simulated and real data that the empirical coverage rate of our private confidence intervals was often (sometimes substantially) higher than the nominal coverage rate. An interesting open question is whether this is inherent for non-parametric CDP confidence intervals for the median. Further, if this is unavoidable, then what distributional assumptions are required to narrow the gap between the empirical and nominal coverage rates?

Finally, perhaps the strongest limitation of our paper is the reliance on the assumption that the sample is drawn using simple random sampling with replacement. Such a sampling design will never be used in the survey context in practice. Thus, the important next step will be to extend the methodology to allow for more complex designs. However, this raises many challenging problems. First, all algorithms assume that the sample is iid, which is typically not true in practice and it is not obvious which adjustments would be necessary to account for this. Second, many sampling designs are informative, meaning that the sampling design is data dependent, which has consequences on the privacy guarantees that are difficult to quantify. Third, the sampling weights that would be used to compute the median (or any other quantile) would influence the sensitivity of the statistic. Fourth, further adjustments such as calibration or dealing with non-response have additional impacts on the privacy guarantees. Fifth, computing confidence intervals for the median is challenging for many sampling designs even for the non-private case. Addressing all these aspects is well beyond the scope of this paper. However, each of these aspects would be an interesting and important area of

future research with impact well beyond the median application considered in this paper.

Acknowledgments

The work of Drechsler, Globus-Harris, Sarathy and Smith on this project was funded in part by US Census Bureau cooperative agreements CB16ADR0160001 and CB20ADR0160001. The work of McMillan (while at BU) and Smith was also supported in part by NSF award CCF-1763786 as well as a Sloan Foundation research award. Part of this work was done while McMillan was supported by a Fellowship from the Cybersecurity & Privacy Institute at Northeastern University and NSF grant CCF-1750640. Globus-Harris' work at BU was supported by funding from the Hariri Institute for Computing, and was supported by the CIS PhD Graduate Fellowship at U Penn. The opinions, findings, conclusions and recommendations expressed herein are those of the authors and do not necessarily reflect the views of the US Census Bureau or other funding sources.

Our work was prompted in part by discussions with sociologists John Logan and Brian Stults, in the context of their work on integrating data across time-varying tract boundaries [Logan et al., 2021]. We are also grateful for helpful conversations with and comments from (in no particular order) Rolando Rodriguez, Ryan Cummings, Thomas Steinke, Shurong Lin, Eric Kolaczyk and Salil Vadhan.

References

Jacob Abernethy, Chansoo Lee, and Ambuj Tewari. Perturbation techniques in online learning and optimization. *Perturbations, Optimization, and Statistics*, page 233, 2016.

John M Abowd. Staring-down the database reconstruction theorem. In *Joint Statistical Meetings, Vancouver, Canada*, 2018.

Daniel Alabi, Audra McMillan, Jayshree Sarathy, Adam Smith, and Salil Vadhan. Differentially private simple linear regression, 2020.

Hilal Asi and John C Duchi. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14106–14117. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/a267f936e54d7c10a2bb70dbe6ad7a89-Paper.pdf`.

Andr'es F. Barrientos, J. Reiter, Ashwin Machanavajjhala, and Yan Chen. Differentially private significance tests for regression coefficients. *Journal of Computational and Graphical Statistics*, 28:440 – 453, 2017.

R. Bassily, A. D. Smith, and Abhradeep Thakurta. Private empirical risk minimization, revisited. *ArXiv*, abs/1405.7085, 2014.

Garrett Bernstein and Daniel R Sheldon. Differentially private bayesian inference for exponential families. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/08040837089cdf46631a10aca5258e16-Paper.pdf`.

Garrett Bernstein and Daniel R Sheldon. Differentially private bayesian linear regression. In *Advances in Neural Information Processing Systems 32*, pages 523–533. 2019.

Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan Ullman. Coinpress: Practical private mean and covariance estimation. *arXiv preprint arXiv:2006.06618*, 2020.

Thomas W. Brawner and J. Honaker. Bootstrap inference and differential privacy: Standard errors for free. 2018.

Victor-Emmanuel Brunel and Marco Avella-Medina. Propose, test, release: Differentially private estimation with high probability. *ArXiv*, abs/2002.08774, 2020.

Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

Mark Bun and Thomas Steinke. Average-case averages: Private algorithms for smooth sensitivity and mean estimation. In *Advances in Neural Information Processing Systems 32*, pages 181–191, 2019.

T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3), November 2011. ISSN 1094-9224.

Graham Cormode. Building blocks of privacy: Differentially private mechanisms. pages 18–19.

Simon Couch, Zeki Kazan, Kaiyan Shi, Andrew Bray, and Adam Groce. Differentially private nonparametric hypothesis testing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 737–751, 2019.

K. H. Degue and J. L. Ny. On differentially private gaussian hypothesis testing. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 842–847, 2018. doi: 10.1109/ALLERTON.2018.8635911.

Ilias Diakonikolas, Moritz Hardt, and Ludwig Schmidt. Differentially private learning of structured discrete distributions. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper/2015/file/2b3bf3eee2475e03885a110e9acaab61-Paper.pdf`.

Christos Dimitrakakis, Blaine Nelson, Aikaterini Mitrokotsa, and Benjamin I. P. Rubinstein. Robust and private bayesian inference. In Peter Auer, Alexander Clark, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory*, pages 291–305, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11662-4.

Vito D'Orazio, J. Honaker, and G. King. Differential privacy for social science inference. *Alfred P. Sloan Foundation Economic Research Paper Series*, 2015.

Joerg Drechsler. Differential privacy for government agencies–are we there yet? *arXiv preprint arXiv:2102.08847*, 2021.

Wenxin Du, Canyon Foot, Monica Moniot, Andrew Bray, and Adam Groce. Differentially private confidence intervals. *arXiv*, arXiv:2001.02285, 2020.

C. Dwork and G. N. Rothblum. Concentrated differential privacy. *ArXiv*, abs/1603.01887, 2016.

Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *STOC*, volume 9, pages 371–380, 2009.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 486–503, Berlin, Heidelberg, 2006a. Springer Berlin Heidelberg. ISBN 978-3-540-34547-3.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 265–284, 2006b.

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 715–724, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300506.

Georgina Evans and Gary King. Statistically valid inferences from differentially private data releases, with application to the facebook urls dataset. *Political Analysis*, 2021 2021.

Georgina Evans, Gary King, Margaret Schwenzfeier, and Abhradeep Thakurta. Statistically valid inferences from privacy protected data, 2021.

C. Ferrando, Shu-Fan Wang, and D. Sheldon. General-purpose differentially-private confidence intervals. *ArXiv*, abs/2006.07749, 2020.

James Foulds, Joseph Geumlek, Max Welling, and Kamalika Chaudhuri. On the theory and practice of privacy-preserving bayesian data analysis. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, page 192–201, Arlington, Virginia, USA, 2016. AUAI Press. ISBN 9780996643115.

Marco Gaboardi, Hyun Lim, Ryan Rogers, and Salil Vadhan. Differentially private chi-squared hypothesis testing: Goodness of fit and independence testing. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2111–2120, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/rogers16.html.

Marco Gaboardi, Ryan Rogers, and Or Sheffet. Locally private mean estimation: z-test and tight confidence intervals. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2545–2554. PMLR, 16–18 Apr 2019.

Simson L. Garfinkel, John M. Abowd, and Christian Martindale. Understanding database reconstruction attacks on public data. *Commun. ACM*, 62(3):46–53, 2019. doi: 10.1145/3287287. URL https://doi.org/10.1145/3287287.

Jennifer Gillenwater, Matthew Joseph, and Alex Kulesza. Differentially private quantiles, 2021.

Ruobin Gong. Exact inference with approximate computation for differentially private data via perturbations, 2019.

Mikko Heikkilä, Eemil Lagerspetz, Samuel Kaski, Kana Shimizu, Sasu Tarkoma, and Antti Honkela. Differentially private bayesian learning on distributed data. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/dfce06801e1a85d6d06f1fdd4475dacd-Paper.pdf.

James Honaker. Efficient use of differentially private binary trees, 2015.

Aaron Johnson and Vitaly Shmatikov. Privacy-preserving data exploration in genome-wide association studies. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 1079–1087, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321747. doi: 10.1145/2487575.2487687. URL https://doi.org/10.1145/2487575.2487687.

V. Karwa and S. Vadhan. Finite sample differentially private confidence intervals. In *ITCS*, 2018.

Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 123–134, 2010.

John R. Logan, Charles Zhang, Brian Stults, and Todd Gardner. Improving estimates of neighborhood change with constant tract boundaries. *Applied Geography*, 132, 2021. doi: 10.1016/j.apgeog.2021.102476.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 94–103, 2007.

Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 75–84, 2007.

Steven Ruggles, Sarah Flood, Sophia Foster, Ronald Goeken, Jose Pacas, Megan Schouweiler, and Matthew Sobek. Ipums usa: Version 11.0 1940 decennial census. https://doi.org/10.18128/D010.V11.0, 2021.

Jessica Semega, M. Kollar, E.A. Shrider, and John F. Creamer. Current population reports, p60-270, income and poverty in the united states: 2019. Technical report, U.S. Census Bureau, U.S. Government Publishing Office, Washington, DC, 2020, 2020. Table also available at https://www.census.gov/data/tables/2020/demo/income-poverty/p60-270.html.

Adam Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 813–822, 2011.

Abhradeep Guha Thakurta and Adam Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 819–850, Princeton, NJ, USA, 12–14 Jun 2013. PMLR. URL http://proceedings.mlr.press/v30/Guha13.html.

Christos Tzamos, Emmanouil-Vasileios Vlatakis-Gkaragkounis, and Ilias Zadik. Optimal private median estimation under minimal distributional assumptions. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3301–3311. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/21d144c75af2c3a1cb90441bbb7d8b40-Paper.pdf.

U.S. Census Bureau. Income data tables. https://www.census.gov/topics/income-poverty/income/data/tables.html, 2020a. Accessed: 2020-03-14.

U.S. Census Bureau. Income, poverty, and health insurance: 2019. Live Press Conference, September 15, 2020, 2020b. Slides available at https://www.census.gov/content/dam/Census/newsroom/press-kits/2020/iphi/20200915-iphi-slides-plot-points.pdf. Accessed: 2020-03-16.

D. Vu and A. Slavkovic. Differential privacy for clinical trial data: Preliminary evaluations. In *2009 IEEE International Conference on Data Mining Workshops*, pages 138–143, 2009. doi: 10.1109/ICDMW.2009.52.

Yu-Xiang Wang. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 93–103, 2018.

Yu-Xiang Wang, Stephen E. Fienberg, and Alexander J. Smola. Privacy for free: Posterior sampling and stochastic gradient monte carlo. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2493–2502. JMLR.org, 2015a.

Yue Wang, Jaewoo Lee, and D. Kifer. Differentially private hypothesis testing, revisited. *ArXiv*, abs/1511.03376, 2015b.

Appendix

This appendix contains further details and pseudocode for the four mechanisms evaluated in the main paper: `ExpMech`, `CDFPostProcess`, `NoisyBinSearch`, and `BinSearch` + `CDF` (Sections B to E). Note that code for these algorithms can be found at `https://github.com/anonymous-conf-medians/dp-medians`. This appendix also provides simulation results for some additional algorithms (Section F) that we did not explore further as they were strictly dominated by other algorithms in all the parameter settings considered in the main paper.

A From Continuous Distributions to All Distributions

The algorithms and proofs in this paper focus on confidence intervals for the class of continuous distributions $\Delta_{\mathscr{C}}(\mathbb{R})$. The relevant property of this class of distributions is that the distribution of the rank of the median is exactly given by

$$\Pr(\texttt{rank}_d(\texttt{med}(P)) = m) = \Pr(\text{Bin}(n, 1/2) = m).$$

This property can fail for distributions where the median itself has non-zero mass. However, we can use a simple transformation to extend our confidence intervals for continuous distributions to a function that is arbitrarily close to a confidence interval for the set of all distributions on $\mathbb{R}$, $\Delta(\mathbb{R})$. The transformation involves adding a small amount of Gaussian noise to the samples from P, in order to produce samples from a continuous distribution that are close to samples from P. A confidence interval algorithm for $\Delta_{\mathscr{C}}(\mathbb{R})$ is then run on the resulting samples.

We'll say a function $M : \mathscr{X}^n \to I_{\mathbb{R}}$ is a $(\beta, 1-\alpha)$-*good confidence interval* for $Q \in \Delta_{\mathscr{C}}(\mathbb{R})$ if with probability at least $1-\alpha$,

$$\exists m \in M(X) \text{ s.t. } \Pr_{x\sim Q}(x \le m) \in [1/2-\beta, 1/2+\beta]$$

where the probability is taken over the randomness of M and $X \sim Q^n$. Note that a $(0, 1-\alpha)$-good confidence interval is simply a $1-\alpha$-confidence interval.

Let Φ_σ be the cumulative distribution function (CDF) of the Gaussian $\mathscr{N}(0, \sigma^2)$. Algorithm 1 describes the transformation of M. Note that M' expands the confidence interval by $\Phi_\sigma^{-1}(1-\beta)$. We can make β and $\Phi_\sigma^{-1}(1-\beta)$ both arbitrarily small by setting σ^2 to be arbitrarily small.

Algorithm 1: M', a $(\beta, 1-\alpha)$-good confidence interval for P

Data: $X \sim P^n$, where $P \in \Delta(\mathbb{R})$
Hyperparams: $(0, 1-\alpha)$-confidence interval algorithm M for distributions in $\Delta_{\mathscr{C}}(\mathbb{R})$, $\sigma^2 > 0$, $\beta \in [0, 1/2]$
$X' = X + \mathscr{N}(0, \sigma^2 I_n)$
Let $a = \Phi_\sigma^{-1}(1-\beta)$, where Φ_σ is the CDF of $\mathscr{N}(0, \sigma^2)$
return [M(X')-a, M(X')+a]

Lemma A.1. *For all M, σ^2 and $\beta \in [0,1]$, if M is an α-confidence interval for $\Delta_{\mathscr{C}}(\mathbb{R})$ then M' (as defined in Algorithm 1) is a $(\beta, 1-\alpha)$-good confidence interval for $\Delta(\mathbb{R})$.*

Proof. Let $a = \Phi_\sigma^{-1}(1-\beta)$, and let $P \in \Delta(\mathbb{R})$ and Q be the distribution of the sum $z = x + y$ where $x \sim P$ and $y \sim \mathbb{N}(0, \sigma^2)$. Note first that $Q \in \Delta_{\mathscr{C}}(\mathbb{R})$. Now, let $M(Z) = [L, U]$ be the output of the confidence interval on Q^n. Notice that $\texttt{med}(Q) \in M(Z)$ if and only if $\Pr_{z\sim Q}(z \le L) \le \frac{1}{2}$ and $\Pr_{z\sim Q}(z \ge U) \le \frac{1}{2}$. Suppose that $\texttt{med}(Q) \in M(Z)$ (which happens with probability $\ge 1-\alpha$), then

$$\Pr_{x\sim P}(x \le L - a) = \Pr(z - y \le L - a) \le \Pr(z \le L \text{ or } y \ge a) \le \frac{1}{2} + (1 - \Phi_\sigma(a)) = \frac{1}{2} + \beta$$

and similarly

$$\Pr(x \ge U + a) = \Pr_{x\sim P}(z - y \ge U + a) \le \Pr(z \ge U \text{ or } y \le -a) \le \frac{1}{2} + \Phi_\sigma(-a) = \frac{1}{2} + \beta.$$

Therefore, there exists a pair m and γ such that $\Pr_{x\sim P}(x \le m) = \gamma$, $m \in [L-a, U+a]$ and $\gamma \in [\frac{1}{2} - \beta, \frac{1}{2} + \beta]$. □

B Details: Confidence intervals based on exponential mechanism, `ExpMech`

Theorem B.1 (Exponential Mechanism McSherry and Talwar [2007])**.** *Given a space of datasets $\mathscr{X}^n$ and an arbitrary range, $\mathscr{R}$, let $u : \mathscr{X}^n \times \mathscr{R} \to \mathbb{R}$ be a utility function that maps dataset/output pairs to utility scores. For a fixed dataset $d \in \mathscr{X}^n$ and privacy parameter $\varepsilon \in \mathbb{R}_+$, the exponential mechanism outputs $x \in [r_\ell, r_u]$ with probability proportional to* $\exp\left(\frac{\varepsilon u(d,x)}{2\Delta u}\right)$*, where*

$$\Delta(u) = \max_{x \in \mathscr{R}} \max_{d,d' neighbours} |u(d,x) - u(d',x)|.$$

The exponential mechanism is ε-DP.

Let the range of possible outputs be $\mathscr{R} = [r_\ell, r_u]$. The standard exponential mechanism to estimate the value $d_{(k)}, k \in [1,2,\ldots,n]$ (described in [Smith, 2011] for $k = n/2$) uses the following utility function.

$$u(d,x) = -\left|\texttt{rank}_d(x) - k\right|$$

This utility function captures how far x is *in rank* from $d_{(k)}$. However, this standard mechanism estimates $d_{(k)}$ poorly if the datapoints in d are highly concentrated around this value.[7] Below is a variant on the standard exponential mechanism designed to perform well even in this situation.

Definition B.1 (θ-Widened Exponential Mechanism [Alabi et al., 2020])**.** *For a widening parameter $\theta > 0$ and target rank $k \in [1,2,\ldots,n]$, the θ-widened exponential mechanism uses the following utility function.*

$$u(d,x) = -\min\left\{\left|\texttt{rank}_d(a) - k\right| \;:\; |a - x| \leq \theta\right\}$$

The θ-widened utility function can be implemented in different ways; Algorithm 2 offers one method of doing so. To sample efficiently from the distribution defined by the utility function, we implement a two-step strategy as shown in prior work [Alabi et al., 2020, Cormode]: First, we sample an interval, using the fact that sampling from the exponential mechanism is equivalent to choosing the value with maximum utility score after i.i.d. Gumbel-distributed noise has been added to the utility scores [Abernethy et al., 2016]. Second, we sample an output uniformly at random from that interval.

Lemma B.1. *`ExpMechPoint` (Algorithm 2) is ε-DP.*

Proof. Follows directly from Theorem B.1. □

Definition B.2 ((t,θ,β)-good)**.** *Let A be a randomized mechanism that outputs a real-valued variable m. For a fixed dataset $d \in \mathscr{X}$, $\theta \in \mathbb{R}_+$, and $\beta \in (0,1)$, m is (t,θ,β)-good with respect to target rank $k \in [1,2,\ldots,n]$ if there exists a datapoint $a \in d$ such that with probability at least $1-\beta$,*

$$|m - a| \leq \theta \text{ and } \left|\texttt{rank}_d(a) - k\right| \leq t,$$

where the probability is over the randomness of A.

Lemma B.2. *Let $d \in \mathscr{X}^n$ be a dataset and $k \in [1,2,\ldots,n]$ be a target rank. Let $A_k^\varepsilon(d)$ be the θ-widened exponential mechanism with privacy parameter $\varepsilon \in \mathbb{R}_+$, widening parameter $\theta \in \mathbb{R}$, and range parameter $\mathscr{R} \subset \mathbb{R}$, and let us assume that $d_{(k)} \in \mathscr{R}$. For $\beta \in (0,1)$, let $t = \ln\left((|\mathscr{R}| - 2\theta)/(2\theta\beta)\right)/\varepsilon$. Then, the output of $A_k^\varepsilon(d)$ is (t,θ,β)-good.*

Proof. We will upper bound the probability density of outputs that are *not* (t,θ)-good with respect to target rank k, ie. outputs m for which there does not exist a datapoint $a \in d$ such that $|m-a| \leq \theta$ and $\left|\texttt{rank}_d(a) - k\right| \leq t$.

To do so, recall that the θ-widened exponential mechanism assigns utility scores to dataset/output pairs according to Definition B.1. For a given t, let us define *good outputs* as those having a utility score $\geq -t$, which are assigned unnormalized probability density of at least 1, and *bad outputs* as those having a utility score $< -t$, which are assigned unnormalized probability density of at most $\exp(-t\varepsilon)$. By definition of the θ-widened utility function, the good

[7] See [Alabi et al., 2020] for an explanation of this case.

Algorithm 2: `ExpMechPoint`: θ-Widened Exponential Mechanism for Quantile Estimation

Data: $d = (d_1, \ldots, d_n) \in \mathbb{R}^n$
Privacy params: $\varepsilon \in \mathbb{R}_+$
Hyperparams: $k \in [n], \mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}, \theta \in \mathbb{R}_+$
Clip d to the range $[r_\ell, r_u]$, setting values less than r_ℓ or greater than r_u to r_ℓ and r_u respectively.
$n = |d|$
Sort d in increasing order
for $i \in [1,k]$ **do**
  $d_i = \max(r_\ell, d_i - \theta)$
for $i \in [k+1, n]$ **do**
  $d_i = \min(r_u, d_i + \theta)$
Insert r_ℓ and r_u into d and set $n = n+2$
Set maxNoisyScore $= -\infty$
Set argMaxNoisyScore $= -1$
for $i \in [2, n)$ **do**
  score $= \log(d_i - d_{i-1}) - \frac{\varepsilon}{2} \cdot |i-k|$
  $N \sim \text{Gumbel}(0,1)$
  noisyScore $=$ score $+ N$
  if *noisyScore* $>$ *maxNoisyScore* **then**
    maxNoisyScore $=$ noisyScore
    argMaxNoisyScore $= i$
left $= d_{\text{argMaxNoisyScore}-1}$
right $= d_{\text{argMaxNoisyScore}}$
Sample $\tilde{m} \sim \text{Unif}[\text{left}, \text{right}]$
return $\tilde{m}$

outputs must span an interval of size at least 2θ and the bad outputs span an interval of size at most $|\mathscr{R}| - 2\theta$. Therefore, we have that

$$\begin{aligned}
&\Pr_A(\nexists\, a \in d : |A_k^\varepsilon(d) - a| \le \theta \text{ and } |\texttt{rank}_d(a) - k| \le t) \\
&\quad \le \Pr_A(\nexists\, x \in \mathscr{R} : |A_k^\varepsilon(d) - x| \le \theta \text{ and } |\texttt{rank}_d(x) - k| \le t) \\
&\quad = \Pr_A(\forall\, x \in \mathscr{R}, |A_k^\varepsilon(d) - x| > \theta \text{ or } |\texttt{rank}_d(x) - k| > t) \\
&\quad = \Pr_A(\ \forall x \in [A_k^\varepsilon(d) - \theta, A_k^\varepsilon(d) + \theta] \text{ we have } |\texttt{rank}_d(x) - k| > t) \\
&\quad = \Pr_A([A_k^\varepsilon(d) - \theta, A_k^\varepsilon(d) + \theta] \subseteq \text{bad outputs}) \\
&\quad \le \frac{\Pr_A\left([A_k^\varepsilon(d) - \theta, A_k^\varepsilon(d) + \theta] \subseteq \text{bad outputs}\right)}{P_A\left([A_k^\varepsilon(d) - \theta, A_k^\varepsilon(d) + \theta] \subseteq \text{good outputs}\right)} \\
&\quad \le \frac{(|\mathscr{R}| - 2\theta)\exp(-t\varepsilon)}{2\theta}
\end{aligned}$$

Setting this probability to be within β, we can solve for t as

$$t \ge \frac{1}{\varepsilon} \ln\left(\frac{|\mathscr{R}| - 2\theta}{2\theta\beta}\right)$$

The resulting bound is tight by virtue of the worst-case example. □

Next, we will consider two ways in which we can use `ExpMechPoint` to create a confidence interval for the median. The first (`ExpMechUnion`) consists of taking a union bound over the probability that the non-private interval

fails to capture the true median, and the probability that the private interval fails to capture the non-private interval. The second (ExpMech) is a more nuanced approach that accounts for the noise due to sampling and noise due to privacy together.

B.1 Union bound confidence interval

The following pseudocode describes ExpMechUnion or ExpMech (depending on the boolean hyperparameter Union). The sub-algorithm ComputeExpMechTargets will be described in the next subsection, as it is only called when $\texttt{Union} = 0$.

Algorithm 3: ExpMech(Union): ε-DP Algorithm

Data: $d = (d_1, \dots, d_n) \in \mathbb{R}^n$
Privacy params: $\varepsilon \in \mathbb{R}_+$
Hyperparams: $\alpha \in (0,1), \texttt{Union} \in \{0,1\}, \mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}, \theta \in \mathbb{R}_+, \beta_2 \in (0, \alpha)$
$t = \frac{1}{\varepsilon} \cdot \ln\left(\frac{|\mathscr{R}|-2\theta}{\theta \cdot \beta_2}\right)$
if Union **then**
  $\beta_1 = \frac{\alpha - \beta_2}{1-\beta_2/2}$
  $\mathtt{N}^{\alpha}_{\varepsilon,L} = \lfloor \mathtt{N}^{\beta_1}_L - t \rfloor$ // $[d_{(\mathtt{N}^{\beta_1}_L)}, d_{(\mathtt{N}^{\beta_1}_U)}]$ is the nonprivate $(1-\beta_1)$-confidence interval for the median (see Lemma 2.3).
  $\mathtt{N}^{\alpha}_{\varepsilon,U} = \lceil \mathtt{N}^{\beta_1}_U + t \rceil$
else
  $\mathtt{N}^{\alpha}_{\varepsilon,L}, \mathtt{N}^{\alpha}_{\varepsilon,U} = \texttt{ComputeExpMechTargets}(n, \varepsilon, \alpha, \mathscr{R}, \theta)$
$\widetilde{\mathtt{ci}^{\alpha}_L}(d) = \texttt{ExpMechPoint}(d, \varepsilon/2, (\mathtt{N}^{\alpha}_{\varepsilon,L}, \mathscr{R}, \theta)) - \theta$
$\widetilde{\mathtt{ci}^{\alpha}_U}(d) = \texttt{ExpMechPoint}(d, \varepsilon/2 (\mathtt{N}^{\alpha}_{\varepsilon,U}, \mathscr{R}, \theta)) + \theta$
return $[\widetilde{\mathtt{ci}^{\alpha}_L}(d), \widetilde{\mathtt{ci}^{\alpha}_U}(d)]$

First, we show that both ExpMechUnion and ExpMech are ε-DP.

Lemma B.3. ExpMech(Union) *(Algorithm 3) is ε-DP.*

Proof. The computations of $t, \beta_1, \mathtt{N}^{\alpha}_{\varepsilon,L}$, and $\mathtt{N}^{\alpha}_{\varepsilon,U}$ do not depend on the dataset d. Therefore, when analyzing the privacy loss, we simply need to consider the two calls the algorithm makes to ExpMechPoint, each with privacy parameter $\varepsilon/2$. By Lemma B.1, each of these algorithms is $\varepsilon/2$-DP, so by composition, ExpMech is ε-DP. □

Then, we show that ExpMechUnion produces a valid confidence interval.

Lemma B.4. *Let dataset d be drawn i.i.d. from a distribution $P \in \Delta_{\mathscr{C}}(\mathbb{R})$ with population median $\texttt{med}(P)$. Let $\varepsilon > 0, \mathscr{R} \in \mathbb{R}, \theta > 0$, and let us assume that $\texttt{med}(P) \in \mathscr{R}$. For $\alpha \in (0,1)$ and $\beta_2 \in (0, \alpha)$, let $[\widetilde{\texttt{ci}^{\alpha}_L}, \widetilde{\texttt{ci}^{\alpha}_U}]$ be the output of $\texttt{ExpMechUnion}(d, \varepsilon, (\alpha, \texttt{Union} = 1, \mathscr{R}, \theta, \beta_2))$. Then, with probability at least $1-\alpha$,*

$$\texttt{med}(P) \in [\widetilde{\texttt{ci}^{\alpha}_L}, \widetilde{\texttt{ci}^{\alpha}_U}],$$

where the probability is over the randomness in both the dataset d and the mechanism $\texttt{ExpMechUnion}$.

Proof. First, letting $\texttt{ci}^{\beta_1}_L = d_{(\mathtt{N}^{\beta_1}_L)}$ and $\texttt{ci}^{\beta_1}_U = d_{(\mathtt{N}^{\beta_1}_U)}$, for any $\beta_1 \in (0,1)$ we have by Lemma 2.3 that

$$\Pr_d\left(\texttt{med}(P) < \texttt{ci}^{\beta_1}_L\right) = \Pr_d\left(\texttt{rank}_d(\texttt{med}(P)) < \mathtt{N}^{\beta_1}_L\right) \le \beta_1/2. \tag{3}$$

Then, for $\mathtt{N}^{\alpha}_{\varepsilon,L} = \mathtt{N}^{\beta_1}_{L} - t$, and $\mathtt{N}^{\alpha}_{\varepsilon,U} = \mathtt{N}^{\beta_1}_{U} + t$, `ExpMechUnion` (Algorithm 3) outputs the interval $[\widetilde{\mathtt{ci}^{\alpha}_{L}}, \widetilde{\mathtt{ci}^{\alpha}_{U}}]$, where $\widetilde{\mathtt{ci}^{\alpha}_{L}} = A^{\varepsilon/2}_{\mathtt{N}^{\alpha}_{\varepsilon,L}}(d) - \theta$ and $\widetilde{\mathtt{ci}^{\alpha}_{U}} = A^{\varepsilon/2}_{\mathtt{N}^{\alpha}_{\varepsilon,U}}(d) + \theta$. By Lemma B.2, the output of $A^{\varepsilon/2}_{\mathtt{N}^{\alpha}_{\varepsilon,L}}(d)$ is $(t, \theta, \beta_2/2)$-good with respect to rank $\mathtt{N}^{\alpha}_{\varepsilon,L}$. By Definition B.2, this means that

$$\Pr_{A}\left(\widetilde{\mathtt{ci}^{\alpha}_{L}} > \mathtt{ci}^{\beta_1}_{L}\right) = \Pr_{A}\left(\mathtt{rank}_d(A^{\varepsilon/2}_{\mathtt{N}^{\alpha}_{\varepsilon,L}}(d) - \theta) > \mathtt{N}^{\beta_1}_{L}\right) \leq \beta_2/2 \tag{4}$$

Putting these together, we consider the lower endpoint of the interval $[\widetilde{\mathtt{ci}^{\alpha}_{L}}, \widetilde{\mathtt{ci}^{\alpha}_{U}}]$. We can upper bound the failure probability as follows.

$$\begin{aligned}
\Pr_{A,d}\left(\mathtt{med}(P) < \widetilde{\mathtt{ci}^{\alpha}_{L}}\right) &= \Pr_{A}\left(\mathtt{med}(P) < \widetilde{\mathtt{ci}^{\alpha}_{L}} \mid \mathtt{med}(P) < \mathtt{ci}^{\beta_1}_{L}\right) \cdot \Pr_{d}\left(\mathtt{med}(P) < \mathtt{ci}^{\beta_1}_{L}\right) \\
&\qquad + \Pr_{A}\left(\mathtt{med}(P) < \widetilde{\mathtt{ci}^{\alpha}_{L}} \mid \mathtt{med}(P) \geq \mathtt{ci}^{\beta_1}_{L}\right) \cdot \Pr_{d}\left(\mathtt{med}(P) \geq \mathtt{ci}^{\beta_1}_{L}\right) \\
&\leq 1 \cdot \Pr_{d}\left(\mathtt{med}(P) < \mathtt{ci}^{\beta_1}_{L}\right) + \Pr_{A}\left(\widetilde{\mathtt{ci}^{\alpha}_{L}} \geq \mathtt{ci}^{\beta_1}_{L}\right) \cdot \Pr_{d}\left(\mathtt{med}(P) \geq \mathtt{ci}^{\beta_1}_{L}\right) \\
&\leq \beta_1/2 + (\beta_2/2) \cdot (1 - \beta_1/2) \\
&= \alpha/2
\end{aligned}$$

where the second inequality follows from (3) and (4), and the final equality follows from the definition of β_1 in Algorithm 3. A similar inequality holds for the upper endpoint of the interval, so a union bound gives the desired result. □

B.2 Tighter Confidence Interval

Next, we consider the more nuanced approach. Let $P \in \Delta_{\mathscr{C}}(\mathbb{R})$ be a population distribution function, where $\mathtt{med} = \mathtt{med}(P)$ is the population median. For a dataset $d = (d_1, \ldots, d_n)$ where d_i is sampled i.i.d. from distribution P, let $\mathtt{rank}_d(a)$ denote the rank of real value a within dataset d. Let $A^{\varepsilon}_{k}(d)$ be the output of the θ-widened exponential mechanism on dataset d that estimates the value at rank k. For a given k_L, k_U, we would like to control the probability that the interval $[A(d,k_L) - \theta, A(d,k_U) + \theta]$ fails to contain the true median $\mathtt{med}$. In particular, for $\alpha \in (0,1)$, we would like to find the target ranks k_L and k_U closest to $n/2$ such that

$$\begin{aligned}
\Pr_{A,d}\left(A^{\varepsilon/2}_{k_L}(d) - \theta > \mathtt{med}\right) &\leq \alpha/2 \\
\Pr_{A,d}\left(A^{\varepsilon/2}_{k_U}(d) + \theta < \mathtt{med}\right) &\leq \alpha/2
\end{aligned}$$

Algorithm 4: `ComputeExpMechTargets`

Input: $n \in \mathbb{N}, \varepsilon \in \mathbb{R}_+, \alpha \in (0,1), \mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}, \theta \in \mathbb{R}_+$
for $k_L \in \mathbb{N}, 1 \leq k_L \leq n/2$ **do**
  $p_{k_L} = C_{\mathrm{Bin}}(k_L - 1) + \sum_{m=k_L}^{n} C'_{\mathrm{Bin}}(m) \cdot \frac{(|\mathscr{R}| - 2\theta)\exp(-(m-k_L)\cdot\varepsilon/2)}{2\theta}$
$\mathtt{N}^{\alpha}_{\varepsilon,L} = \max_{k_L \in \mathbb{N}, 1 \leq k_L < \lceil n/2 \rceil} \{k_L : p_{k_L} \leq \alpha/2\}$
for $k_U \in \mathbb{N}, n/2 \leq k_U < n$ **do**
  $p_{k_U} = \sum_{m=1}^{k_U} C'_{\mathrm{Bin}}(m) \cdot \frac{(|\mathscr{R}| - 2\theta)\exp(-(k_U - m)\cdot\varepsilon/2)}{2\theta} + (1 - C_{\mathrm{Bin}}(k_U + 1))$
$\mathtt{N}^{\alpha}_{\varepsilon,U} = \min_{k_U \in \mathbb{N}, \lceil n/2 \rceil \leq k_U < n} \{j : p_{k_U} \leq \alpha/2\}$
return $\mathtt{N}^{\alpha}_{\varepsilon,L}, \mathtt{N}^{\alpha}_{\varepsilon,U}$

In Algorithm 4 (`ComputeExpMechTargets`), we find these target ranks by first computing the probabilities above

for all possible k_L and k_U's, and then by numerically searching for the target ranks closest to $n/2$ such that the probabilities above are both within $\alpha/2$.[8] The following lemma characterizes these probabilities.

Lemma B.5. *Let $P \in \Delta_{\mathscr{C}}(\mathbb{R})$ be a population distribution function, where $\texttt{med} = \texttt{med}(P)$ is the population median. For a dataset $d = (d_1, \ldots, d_n)$ where d_i is sampled iid. from distribution P, let $\texttt{rank}_d(a)$ denote the rank of real value a within dataset d. Let $k_L, k_U \in [1, 2, \ldots, n]$ be target ranks, and let $A^{\varepsilon}_{k_L}(d)$ and $A^{\varepsilon}_{k_U}(d)$ be θ-widened exponential mechanisms on dataset d that estimate the value at rank k_L and k_U, respectively. Let C_{Bin} and C'_{Bin} be the CDF and PDF of the binomial random variable* Bin*$(n, 1/2)$. Then,*

$$\Pr_{A,d}\left(A^{\varepsilon/2}_{k_L}(d) - \theta > \texttt{med}\right) \leq C_{\text{Bin}}(k_L) + \sum_{m=k_L+1}^{m=n} C'_{\text{Bin}}(m) \cdot \frac{(|\mathscr{R}| - 2\theta)\exp\left(-(m-k_L)\varepsilon/2\right)}{2\theta}$$

$$\Pr_{A,d}\left(A^{\varepsilon/2}_{k_U}(d) + \theta < \texttt{med}\right) \leq (1 - C_{\text{Bin}}(k_U + 1)) + \sum_{m=1}^{m=k_U} C'_{\text{Bin}}(m) \cdot \frac{(|\mathscr{R}| - 2\theta)\exp\left(-(k_U-m)\varepsilon/2\right)}{2\theta}$$

Proof. For simplicity, we consider just the first statement pertaining to the lower endpoint of the interval. We can split up the probability into two cases: first, when $\texttt{rank}_d(\texttt{med}) < k_L$, and second, when $\texttt{rank}_d(\texttt{med}) \geq k_L$.

$$\begin{aligned}\Pr_{A,d}\left(A^{\varepsilon/2}_{k_L}(d) - \theta > \texttt{med}\right) = &\sum_{m=1}^{m=k_L-1} \Pr_A\left(A^{\varepsilon/2}_{k_L}(d) - \theta > \texttt{med} \mid \texttt{rank}_d(\texttt{med}) = m\right) \cdot \Pr_d(\texttt{rank}_d(\texttt{med}) = m) \\ &+ \sum_{m=k_L}^{m=n} \Pr_A\left(A^{\varepsilon/2}_{k_L}(d) - \theta > \texttt{med} \mid \texttt{rank}_d(\texttt{med}) = m\right) \cdot \Pr_d(\texttt{rank}_d(\texttt{med}) = m)\end{aligned}$$

For the first case, where $\texttt{rank}_d(\texttt{med}) < k_L$, we simply upper bound the first probability in the summation by 1 and note that the random variable $\mathbf{1}_{\texttt{rank}_d(\texttt{med})=m}$ follows a binomial distribution.

$$\sum_{m=1}^{m=k_L-1} \Pr_A\left(A^{\varepsilon/2}_{k_L}(d) - \theta > \texttt{med} \mid \texttt{rank}_d(\texttt{med}) = m\right) \cdot \Pr_d(\texttt{rank}_d(\texttt{med}) = m) \leq C_{\text{Bin}}(k_L - 1)$$

In the second case, we first observe that the probability of any real value a being greater than med is monotonically increasing in a, which gives

$$\begin{aligned}\Pr_A\left(A^{\varepsilon/2}_{k_L}(d) - \theta > \texttt{med} \mid \texttt{rank}_d(\texttt{med}) = m\right) &\leq \Pr_A\left(A^{\varepsilon/2}_{k_L}(d) > \texttt{med} \mid \texttt{rank}_d(\texttt{med}) = m\right) \\ &= \Pr_A\left(\texttt{rank}_d(A^{\varepsilon/2}_{k_L}(d)) > m\right) \\ &\leq \Pr_A\left(|\texttt{rank}_d(A^{\varepsilon/2}_{k_L}(d)) - k_L| > m - k_L\right) \\ &\leq \frac{(|\mathscr{R}| - 2\theta)\exp\left(-(m-k_L)\varepsilon/2\right)}{2\theta},\end{aligned}$$

where the last inequality follows from Lemma B.2. Therefore, we have that

$$\Pr_{A,d}\left(A^{\varepsilon/2}_{k_L}(d) - \theta > \texttt{med}\right) \leq C_{\text{Bin}}(k_L - 1) + \sum_{m=k_L}^{m=n} C'_{\text{Bin}}(m) \cdot \frac{(|\mathscr{R}| - 2\theta)\exp\left(-(m-k_L)\varepsilon/2\right)}{2\theta}$$

A similar result holds for $\Pr_{A,d}\left(A^{\varepsilon/2}_{k_U}(d) + \theta < \texttt{med}\right)$. □

Validity of the ExpMech confidence interval then follows directly from the selection of the target ranks.

[8] This search can be implemented more efficiently by noting that k_L is greater than or equal to $\lfloor \mathtt{N}_L^{\beta_1} - t \rfloor$ as defined in Algorithm 3, and similarly k_U is less than or equal to $\lceil \mathtt{N}_U^{\beta_1} + t \rceil$.

Lemma B.6. *Let dataset d be drawn i.i.d. from a distribution $P \in \Delta_{\mathscr{C}}(\mathbb{R})$ with population median $\texttt{med}(P)$. For (hyper)parameters $\varepsilon > 0, \mathscr{R} \in \mathbb{R}, \theta > 0$, and $\alpha \in (0,1)$, let $[\widetilde{\texttt{ci}_L^{\alpha}}, \widetilde{\texttt{ci}_U^{\alpha}}]$ be the output of* $\texttt{ExpMech}(d, \varepsilon, (\alpha, \texttt{Union} = 0, \mathscr{R}, \theta, \cdot))$. *If $\texttt{med}(P) \in \mathscr{R}$, then with probability at least $1-\alpha$,*

$$\texttt{med}(P) \in [\widetilde{\texttt{ci}_L^{\alpha}}, \widetilde{\texttt{ci}_U^{\alpha}}],$$

where the probability is over the randomness in both the dataset d and the mechanism ExpMech.

Proof. ComputeExpMechTargets (Algorithm 4, relying on Lemma B.5) returns target ranks $\texttt{N}_{\varepsilon,L}^{\alpha}$ and $\texttt{N}_{\varepsilon,U}^{\alpha}$ such that $Pr_{A,d}\left(A_{\texttt{N}_{\varepsilon,L}^{\alpha}}^{\varepsilon/2}(d) - \theta > \texttt{med}\right) \leq \alpha/2$ and $\Pr_{A,d}\left(A_{\texttt{N}_{\varepsilon,U}^{\alpha}}^{\varepsilon/2}(d) + \theta < \texttt{med}\right) \leq \alpha/2$. ExpMech (Algorithm 3) then sets $\widetilde{\texttt{ci}_L^{\alpha}}(d) = A_{\texttt{N}_{\rho,L}^{\alpha}}^{\varepsilon/2}(d) - \theta$ and $\widetilde{\texttt{ci}_U^{\alpha}}(d) = A_{\texttt{N}_{\rho,U}^{\alpha}}^{\varepsilon/2}(d) + \theta$. The result follows from a union bound. □

C Details: Confidence intervals based on noisy binary search, NoisyBinSearch

Algorithm 5: NoisyBinSearch: ρ-CDP Algorithm

Data: $d = (d_1, \ldots, d_n) \in \mathbb{R}^n$
Privacy params: $\rho \in \mathbb{R}_+$
Hyperparams: $\alpha \in (0,1)$, $\mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}$, $\theta \in \mathbb{R}_+$, γ, $\texttt{LB}, \texttt{UB} \in (0,1)$
$\beta_1 = \gamma\alpha$
$\beta_2 = \frac{\alpha - \beta_1}{1 - \beta_1/2}$
$n = |d|$
$m = \log((r_u - r_\ell)/\theta)$ // number of steps required to get to desired granularity
$\rho_{\text{step}} = \rho/(2m)$
$\beta_{\text{step}} = \beta_2/(2m)$
$t_{\beta_{\text{step}}}^{\rho_{\text{step}}} = \sqrt{\frac{\log(1/\beta_{\text{step}})}{\rho_{\text{step}} n}}$
$q_L = \min\{\texttt{LB}, \texttt{N}_L^{\beta_1}/n - t_{\beta_{\text{step}}}^{\rho_{\text{step}}}\}$
$q_U = \max\{\texttt{UB}, \texttt{N}_U^{\beta_1}/n + t_{\beta_{\text{step}}}^{\rho_{\text{step}}}\}$
noisy-counts-lower = $\texttt{GetNoisyCounts}(d, \rho/2, (n, \beta_2/2, q_L, q_L, q_U, \emptyset, \mathscr{R}, \rho_{\text{step}}, \beta_{\text{step}}))$
noisy-counts-upper = $\texttt{GetNoisyCounts}(d, \rho/2, (n, \beta_2/2, q_U, q_L, q_U, \texttt{noisy-counts-lower}, \mathscr{R}, \rho_{\text{step}}, \beta_{\text{step}}))$
noisy-counts = noisy-counts-lower $\cup$ noisy-counts-upper
return $\texttt{PostProcessUnion}(\texttt{noisy-counts}, n, \texttt{N}_L^{\beta_1}, \texttt{N}_U^{\beta_1}, \beta_2)$

In this section we provide the algorithmic details and validity and privacy proofs for NoisyBinSearch.

Given a dataset $d \in \mathscr{X}^n$, and target quantile $q_{\text{target}} \in (0,1)$, an initial range $\mathscr{R}$ and granularity θ, NoisyBinSearch (outlined in Algorithm 5) consists of two steps. In the first step, the mechanism GetNoisyCounts uses noisy measurements of the empirical CDF to search for $d_{n(q_{\text{target}})}$ using binary search. This noisy binary search step is designed so that with high probability it moves in the right direction at each step, however there is some probability of making a wrong move, hence we need to perform a post-processing step that takes the noisy measurements as input and returns a valid confidence interval.

Pseudo-code for the first step, which we will call GetNoisyCounts is given in Algorithm 6. Let us focus on finding the lower limit of the confidence interval. Given a target quantile q_{target}, this algorithm iterates reduces the search domain by querying the rank of the mid-point x_t of the range. If it is confident that the mid-point is to left of the target quantile then it cuts the domain in half and only keep the right half (similarly if it is confident that the mid-point is to the right, it keep the left half of the range). It continues this process until the entire privacy budget is consumed.

At each iteration we use a portion of the privacy budget ρ to release the noisy rank of the query point x_t. The total privacy budget consumed by the algorithm is the sum of the privacy budget consumed by each step (Lemma 2.2). One

Algorithm 6: `GetNoisyCounts`: ρ-CDP Algorithm

Data: $d = (d_1, \ldots, d_n) \in \mathbb{R}^n$
Privacy params: $\rho \in \mathbb{R}_+$
Hyperparams: $n \in \mathbb{N}, \beta_2 \in (0,1), q_{\text{target}}, q_L, q_U \in (0,1)$, `prev-queries`, $\mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}, \rho_{\text{step}}, \beta_{\text{step}}$
`// prev-queries =` $\{(x, r_x, \sigma_x)\}$ `is a collection of noisy measurements where` $x \in [r_l, r_u]$ `and` $r_x = \texttt{rank}_d(x) + \mathscr{N}(0, \sigma_x^2)$
`lower =` r_ℓ, `upper =` r_u `// The initial search space is the entire range.`
$\rho_{\text{init}} = \rho_{\text{step}}/10$, $\beta_{\text{init}} = \beta_{\text{step}}/10$ `// Budget for initial measurement at every query point; can be arbitrarily small.`
$t = 0$ `// Counter for number of query points.`
$\rho_{\text{used}} = 0$ `// Counter for used privacy budget.`
while $\rho_{used} + \rho_{init} \leq \rho$ **do**
  $x_t = (\texttt{lower} + \texttt{upper})/2$ `// Query point`
  `est-good-enough = False`
  if there exists r_{x_t} and σ_{x_t} such that $(x_t, r_{x_t}, \sigma_{x_t}) \in$`prev-queries` **then**
    $\texttt{avg-noisy-count}_t = r_{x_t}$, $\texttt{avg-var}_t = \sigma_{x_t}$
    `est-good-enough = True`
  `numMeasurements = 0`, $\rho_{\text{t}} = 0$, $\beta_{\text{t}} = 0$
  while `est-good-enough = False` and $\rho_{\text{t}} + \rho_{\text{init}} \leq \rho_{\text{step}}$ and $\rho_{\text{used}} + \rho_{\text{t}} + \rho_{\text{init}} \leq \rho$ **do**
    `numMeasurements = numMeasurements+1`, $\rho_{\text{t}} = \rho_{\text{t}} + \rho_{\text{init}}$, $\beta_{\text{t}} = \beta_{\text{t}} + \beta_{\text{init}}$
    $\texttt{noisy-count}_{\texttt{numMeasurements}} \sim \mathscr{N}(\texttt{rank}_d(x_t), 1/2\rho_{\text{init}})$
    $\texttt{var}_{\texttt{numMeasurements}} = 1/2\rho_{\text{init}}$
    $\texttt{avg-noisy-count}_t = \sum_{k=1}^{\texttt{numMeasurements}} \texttt{noisy-count}_k / \texttt{numMeasurements}$
    $\texttt{avg-var}_t = (\sum_{k=1}^{\texttt{numMeasurements}} \texttt{var}_k) / \texttt{numMeasurements}$
    $K = \sqrt{\texttt{avg-var}} \cdot \Phi^{-1}(1 - \beta_{\text{t}})$ `//` Φ `is the standard normal distribution function`
    if ($\texttt{avg-noisy-count}_t - K > q_U \cdot n$ or $\texttt{avg-noisy-count}_t + K < q_U \cdot n$) and ($\texttt{avg-noisy-count}_t - K > q_L \cdot n$ or $\texttt{avg-noisy-count}_t + K < q_L \cdot n$) **then**
      `est-good-enough = True`
  if $\texttt{avg-noisy-count}_t < n \cdot q_{\text{target}}$ **then**
    `lower=` x_t
  else
    `upper` $= x_t$
  $\rho_{\text{used}} = \rho_{\text{used}} + \rho_{\text{t}}$
  $t = t + 1$
return $(x_1, \texttt{avg-noisy-count}_1, \texttt{avg-var}_1), (x_2, \texttt{avg-noisy-count}_2, \texttt{avg-var}_2), \cdots$

Algorithm 7: `PostProcessUnion`

Input: $(x_1, \texttt{ns}_1, \texttt{var}_1), (x_2, \texttt{ns}_2, \texttt{var}_2), \cdots, (x_T, \texttt{ns}_T, \texttt{var}_T), n \in \mathbb{N}, q_L, q_U, \beta_2 \in (0,1)$
for $t \in [T]$ **do**
  $R_t = \sqrt{\texttt{var}_t}\Phi^{-1}(1 - 2T/\beta_2)$ `//` Φ `is the standard normal distribution function`
  $L_t = \texttt{ns}_t + R_t$
  $U_t = \texttt{ns}_t - R_t$
$l = \max\{x_t \mid \forall t' < t, L_t < q_L\}$
$u = \min\{x_t \mid \forall t' > t, U_t > q_U\}$
return $[l, u]$

option for allocating the privacy budget is to decide in advance the number of iterations and divide the privacy budget by the number of iterations to obtain a *per step* privacy budget. However, we can actually improve on this approach by noticing that if $|\texttt{rank}_d(x_t) - q_{\text{target}} \cdot n|$ is large then we can tolerate a lot of noise in our estimate of $\texttt{rank}_d(x_t)$ and

still determine with high confidence whether $\texttt{rank}_d(x_t) > q_{\text{target}} \cdot n$ or $\texttt{rank}_d(x_t) < q_{\text{target}} \cdot n$. Thus, we may be able to only allocate a very small amount of privacy budget to some steps. For a given query point x_t we do not know a priori how large $|\texttt{rank}_d(x_t) - q_{\text{target}} \cdot n|$ is, and hence how much noise the query can handle. Thus, we start by adding a large amount of noise (using only a small amount of the privacy budget ρ_{init}) to $\texttt{rank}_d(x_t)$. If this noisy estimate is far enough from q_{target} that we can confidently determine which direction to continue with the binary search, then we move and this step has only consumed ρ_{init} privacy budget. Otherwise, we take another noisy measurement of $\texttt{rank}_d(x_t)$ and average the two together. This produces a less noisy estimate, and consumes $2\rho_{\text{init}}$. We continue in this way until either the variance of the estimate is low enough that we can confidently move, or this step has consumed the maximum amount of privacy budget per step ρ_{step}, and we move in the more likely direction. While we search for the left and right hand limit of the confidence interval separately, in many settings the early query points of the binary search will be same for both. Thus, we can improve accuracy by not repeating these noisy queries. This is why we pass `prev-queries` into `GetNoisyCounts`.

The next step is processing the noisy counts to obtain a valid confidence interval. Pseudo-code is given in Algorithm 7 and a validity proof is given in Lemma C.2. This step does not consume additional privacy budget since it is simply post-processing on top of the ρ-CDP output of `GetNoisyCounts`.

Lemma C.1. *Mechanism* `NoisyBinSearch` *(Algorithm 5) is ρ-CDP.*

Proof. The lemma follows immediately from Lemma 2.2 in the main text. The privacy budget ρ is divided between the two calls to `GetNoisyCounts`, which each use privacy budget $\rho/2$. By [Bun and Steinke, 2016, Proposition 1.6], each new noisy measurement $\texttt{noisy-count}_{\texttt{numMeasurements}}$ is ρ_{init}-CDP, so ρ_t and thus ρ_{used} accurately capture the privacy budget consumed per step, and in total at any point during the algorithms run. □

Lemma C.2. *Let dataset d be drawn i.i.d. from a distribution $P \in \Delta_{\mathscr{C}}(\mathbb{R})$ with population median $\texttt{med}(P)$. Given (hyper)parameters $\mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}, \theta \in \mathbb{R}_+, \gamma \in (0,1)$, failure rate $\alpha \in (0,1)$ and privacy parameter $\rho \in \mathbb{R}_+$, let $[\widetilde{ci_L^\alpha}, \widetilde{ci_U^\alpha}] = \texttt{NoisyBinSearch}(d, \rho, (\alpha, \mathscr{R}, \theta, \gamma, 0.5, 0.5))$. If $\texttt{med}(P) \in \mathscr{R}$, then with probability at least $1-\alpha$,*

$$\texttt{med}(P) \in [\widetilde{ci_L^\alpha}, \widetilde{ci_U^\alpha}],$$

where the probability is over the randomness in both the dataset d and the mechanism `NoisyBinSearch`*.*

Proof. While `GetNoisyCounts` is designed to ensure the final output is close to the right quantile, the validity of the confidence interval really comes from the post-processing function `PostProcessUnion`. We have that

$$\begin{aligned}
\Pr_{A,d}(\texttt{med}(P) < \widetilde{\texttt{ci}_L^\alpha}) &\le \Pr_A(\texttt{med}(P) < \widetilde{\texttt{ci}_L^\alpha} \mid \texttt{rank}_d(\texttt{med}(P)) < \texttt{N}_L^{\beta_1}) \cdot \Pr_d(\texttt{rank}_d(\texttt{med}(P)) < \texttt{N}_L^{\beta_1}) \\
&\quad + \Pr_A(\texttt{med}(P) < \widetilde{\texttt{ci}_L^\alpha} \mid \texttt{rank}_d(\texttt{med}(P)) \ge \texttt{N}_L^{\beta_1}) \cdot \Pr_d(\texttt{rank}_d(\texttt{med}(P)) \ge \texttt{N}_L^{\beta_1}) \\
&\le \Pr_d(\texttt{rank}_d(\texttt{med}(P)) < \texttt{N}_L^{\beta_1}) + \Pr_A(\texttt{med}(P) < \widetilde{\texttt{ci}_L^\alpha} \mid \texttt{rank}_d(\texttt{med}(P)) \ge \texttt{N}_L^{\beta_1}) \cdot \Pr_d(\texttt{rank}_d(\texttt{med}(P)) \ge \texttt{N}_L^{\beta_1}) \\
&\le \beta_1/2 + \Pr_A(\texttt{N}_L^{\beta_1} < \texttt{rank}_d(\widetilde{\texttt{ci}_L^\alpha})) \cdot (1 - \beta_1/2),
\end{aligned}$$

where the subscript A denotes that the probability is over the randomness of the mechanism `NoisyBinSearch`. Now, let $\texttt{noisy-counts} = (x_1, \texttt{ns}_1, \texttt{var}_1), (x_2, \texttt{ns}_2, \texttt{var}_2), \cdots, (x_T, \texttt{ns}_T, \texttt{var}_T)$ be the concatenated outputs of the two runs of `GetNoisyCounts` in `NoisyBinSearch`; these are inputs to `PostProcessUnion`. We have the guarantee that for all $t \in T$, $\texttt{ns}_t = \texttt{rank}_d(x_t) + \mathscr{N}(0, \texttt{var}_t)$, and therefore with probability $1 - \beta_2/2$, for all $x \in \{x_1, \cdots, x_T\}$, $|\texttt{ns}_t - \texttt{rank}_d(x_t)| \le R_t$. Now, if $\texttt{N}_L^{\beta_1} < \texttt{rank}_d(\widetilde{\texttt{ci}_L^\alpha})$ implies that there exists x_t such that $x_t < \texttt{med}(P)$ but $\texttt{ns}_t \ge \texttt{N}_L^{\beta_1} + R_t$. But this would imply that $|\texttt{ns}_t - \texttt{rank}_d(x_t)| \ge R_t$. Therefore, $\Pr(\texttt{med}(P) < \widetilde{\texttt{ci}_L^\alpha}) \le \beta_1/2 + \beta_2/2(1-\beta_1/2) = \alpha/2$. Similarly we can argue that $\Pr(\texttt{med}(P) > \widetilde{\texttt{ci}_U^\alpha}) \le \beta_1/2 + \beta_2/2(1-\beta_1/2) = \alpha/2$, so we are done. □

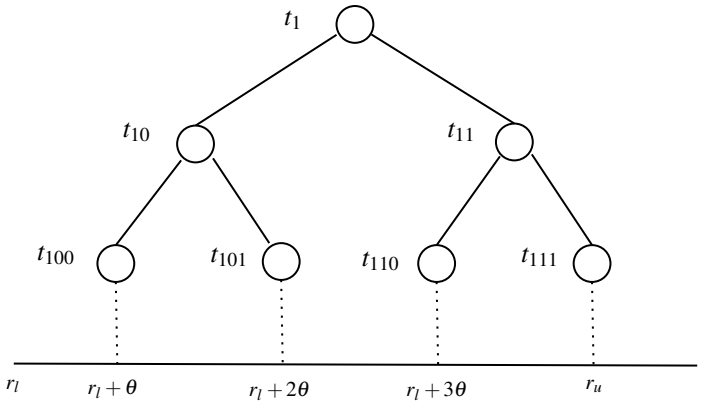


Figure 12: Tree representation of the CDF algorithm, where the counts $t_{100},\ldots,t_{111}$ at the leaves of the tree represent the counts for histogram bins with width θ, their parent nodes represent a histogram with bin width 2θ, and so on.

D Details: Confidence intervals based on CDF estimator `CDFPostProcess`

Instead of generating a series of count queries that vary asymmetrically across the data set as in the binary search process, we could generate the entire empirical CDF of the data and use a similar approach to the binary search algorithm to generate confidence intervals. In our setting, where the number of queries are limited by the privacy budget, if the empirical CDF is of separate interest to a researcher, this is a particularly compelling method. There are many methods for generating a DP CDF [Diakonikolas et al., 2015, Brunel and Avella-Medina, 2020]. We focus on a tree-based mechanism introduced in Li et al. [2010], Dwork et al. [2010], and Chan et al. [2011] and refined in Honaker [2015]. We rely on Honaker [2015]'s algorithm.

D.1 A Tree-Based Approach to Differentially Private CDFs

Note that a simple way to estimate the CDF would be to create a differentially private histogram with a set bin size and sum the bins to the left of a point of interest to generate an estimate of the CDF at that point. However, this means summing multiple noisy counts together, so the accuracy will diminish the more bins that you have to sum together. To avoid summing too many points together, one might instead use a tree-based approach, which uses a tree of multiple histograms that have multiple levels of granularity: we denote this as $T;\text{BinTree}(\mathbb{N},m)$, a binary tree of counts in $\mathbb{N}$ with m levels, $L_1,\ldots,L_m$, as depicted in Figure 12 and described in detail in Algorithm 8. Note that to get a CDF estimate at point $r_\ell+2\theta$ one need only look at t_{10}, and to get the estimate at $r_\ell+3\theta$, one need only sum t_{10} and t_{110}.

Algorithm 8: `DPTree`: ρ-CDP Histogram Tree Algorithm

Data: $d=(d_1,\ldots,d_n)\in\mathbb{R}^n$
Privacy params: $\rho\in\mathbb{R}_+$
Hyperparams: $\mathscr{R}=[r_\ell,r_u]\subset\mathbb{R}$, $m\in\mathbb{N}$
$n=|d|$
Let $T\in\text{BinTree}(\mathbb{N},m)$ be a binary tree with m levels, $L_1,\ldots,L_m$
for $j\in[m]$ **do**
  Let $\text{bin}_1,\ldots,\text{bin}_{2^j}$ be 2^j equally-sized partitions of the range $\mathscr{R}$.
  Generate histogram $\texttt{hist}=\{\#i:d_i\in\text{bin}_b,\ \ 1\le b\le 2^j\}\in\mathbb{N}^{2^j}$
  Add noise sampled from $\mathscr{N}(0,2m/\rho)$ to each element of `hist`.
  Set $L_j=\texttt{hist}$
return T

This concept may be further improved through post-processing by noting that the different noisy counts at different

levels of granularity ought to sum to the same values. Honaker [2015] proposes an optimal method to leverage this information, which we use here. Following Honaker's notation, label each node in the tree in binary, so the root node is 1, the left child of the root is node 10 and its right child is node 11, and so on as in Figure 12. Let the count at node i be t_i, and let the leaf nodes of the tree in Figure 12 represent a histogram with granularity θ, so that t_{100} is the number of data-points that lie between the left of the histogram's range, r_ℓ, to $r_\ell + \theta$, t_{101} is the number of data-points between $r_\ell + \theta$ and $r_\ell + 2\theta$, and so on. Let the histogram at any level have granularity twice that of its children. Note that if the counts were perfectly accurate, it then follows that the parent node's value should be equal to the sum of its children nodes' value, e.g. $t_{10} = t_{100} + t_{101}$ in Figure 12. Similarly, a child nodes' value in a perfectly accurate tree should be equal to its parent node, minus the adjacent child node's count, e.g. $t_{100} = t_{10} - t_{101}$. Let $i\Lambda 1$ be the neighboring child of node i (e.g. $11\Phi 1 = t_{10}$), and let $i\Phi 1$ be the parent of node i.

Honaker leverages these relationships between the counts at each of the nodes to generate an optimal tree in a recursive process. First, the counts at each node using the children node are incorporated into a weighted estimate of each of the counts, where the weight of the child counts at node i is denoted w_i^- and the optimized count "from below" is denoted t_i^-. These are then recursively combined with the counts "from above" (i.e. the count using the parent and the adjacent child) to generate t_i^+, which weights the count from above with w_i^+. Finally, the two are combined to get a fully efficient estimate of each of the nodes, with weight w and efficient count t_i^*.[9] In our setting, we only consider trees with an equal amount of noise on each of the nodes in the tree. This results in several simplifications of the equations in Honaker 2015.

Lemma D.1. *If each noisy count in the tree has noise with variance s added to it, then the weight vectors w^- and w^+ only need to be calculated once per level of the tree, the weight at node i for the summation from below is*

$$w_i^- = \frac{2w_{2i}^-}{2w_{2i}^- + 1},$$

the weight at node i for the summation from above is

$$w_i^+ = \frac{1}{1 + (w_{i\Phi 1}^+ + w_{i\Lambda 1}^-)^{-1}},$$

and the optimal weight w is equal to w^+.

Note that if each node has noise with variance s added to it, then $\forall i$, the variance at node $i, \sigma_i = s$, so $\sigma^-(t_{2i}^-) = \sigma^-(t_{2i+1}^-)$. Then, the weight at node i, w_i may be recursively defined as

$$\begin{aligned}
w_i^- &= \frac{s^{-2}}{s^{-2} + (1/2)(\sigma_{2i}^-)^{-2}} \\
&= \frac{s^{-2}}{s^{-2} + (1/2)\left(s\sqrt{w_{2i}^-}\right)^{-2}} \\
&= \frac{s^{-2}}{s^{-2}(1 + (1/2)(w_{2i}^-)^{-1})} \\
&= \frac{2w_{2i}^-}{2w_{2i}^- + 1},
\end{aligned}$$

where the first line comes from Honaker's definition in his Equation 10. Similarly, from Honaker's Equation 11 it follows that when the noise at each node has the same variance,

$$\begin{aligned}
w_i^+ &= \frac{(\sigma_i)^{-2}}{(\sigma_i)^{-2} + \left[(\sigma_{i\Phi 1}^+)^2 + (\sigma_{i\Lambda 1}^-)^2\right]^{-1}} \\
&= \frac{1}{1 + (w_{i\Phi 1}^+ + w_{i\Lambda 1}^-)^{-1}},
\end{aligned}$$

[9] See Honaker equations 10, 11, and 13 for the full statement of the values of these weights and counts.

which is equivalent to then the expression for the optimal weights in this setting.

Once there is a tree of fully optimized counts, one can read off the CDF at an arbitrary point by traversing the tree in a root-to-leaf path, summing as few of the values together as possible to get the desired estimate, as shown in Algorithm 9.

Algorithm 9: `TreeToCDF`

Input: $n \in \mathbb{N}$, $T \in \text{BinTree}(\mathbb{N}, m)$, $\mathscr{R}_{\text{discrete}} \in \mathbb{R}^{2^m}$, $\theta \in \mathbb{R}_+$, $m \in \mathbb{N}$
noisy-cdf = []
for $x \in \mathscr{R}_{\text{discrete}}$ **do**
  min $\leftarrow r_\ell$, max $\leftarrow r_u$
  count $\leftarrow 0, i \leftarrow 0$
  for $0 \leq j < m$ **do**
    mid $\leftarrow$ (min + max)/2
    if $x =$ max *or* $j = m$ **then**
      $k \leftarrow 2^j + i$
      count $\leftarrow$ count + t_k
      break
    else if $x \leq mid$ **then**
      max $\leftarrow$ mid
      $i \leftarrow 2i$
    else
      min $\leftarrow$ mid
      $i \leftarrow 2i + 1$
      $k \leftarrow 2^{j+1} + i - 1$
      count $\leftarrow T_k$
  Add $(x, \text{count}/n)$ to noisy-cdf
return noisy-cdf

D.2 Confidence Intervals for Quantiles Estimated from Tree-Based CDF

In order to generate a confidence interval for the desired quantiles, we would need to understand what the uncertainty of each of the counts in our estimated CDF was. Since each of these counts is a combination of all of the different counts in the tree, weighted in a way that is recursively defined, this is not trivial to do in a closed-form manner. We can compute the effect of each node on any other node by generating a tree with every node valued at 0 except the node we are interested in, then running the recursive weighting algorithm on that tree, as shown in Alg. 10.

Algorithm 10: `ComputeNodeEffect`

Input: $\sigma^2 \in \mathbb{R}_{\geq 0}$, $i^* \in \mathbb{N}$, $m \in \mathbb{N}$
Construct a binary tree $T \in \text{BinTree}(\mathbb{N}, m)$, where for $0 \leq i < 2^m$,

$$\begin{cases} T_i = 1 & \text{if} \quad i = i^* \\ T_i = 0 & \text{o.w.} \end{cases}$$

Let $T' \in \text{BinTree}(\mathbb{N}, m)$ be the output of the CDF post-processing algorithm from Honaker [2015] on differentially private tree T, where each noisy count in T has variance σ^2.
return T'

We now have a method to understand how much each node effects any other node. If we run this on every single node of the tree, we can then combine them to generate a tree for every node on the tree that describes how much its optimized count is affected by any other node.[10] When summing the counts to generate the CDF, we can then keep track of the total weight of each node in the final count, and from here generate the variance of the count.

[10] Since we add identically distributed noise added to each node's count, there are symmetries in the node effects that can be leveraged to make this process substantially more efficient in practice.

Algorithm 11: `GetVariances`

Input: $T \in \text{BinTree}(\mathbb{N}, m), m \in \mathbb{N}, \mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}, \rho \in \mathbb{R}_+$
Let $\sigma^2 \leftarrow 2m/\rho$
Create binary tree $E \in \text{BinTree}(\mathbb{N}, m)$ with all nodes are set to 0.
$\mathbf{T}' \leftarrow \{\texttt{ComputeNodeEffect}(\sigma^2, i, m)\}_{0 \le i < 2^m}$
$\mathbf{v} \leftarrow \emptyset$
for $0 \le i < 2^m$ **do**
  $\min \leftarrow r_\ell, \max \leftarrow r_u$
  for $0 \le j < 2^m$ **do**
    $\text{mid} \leftarrow (\min + \max)/2$
    if *i is a leftmost node of the tree* **then**
      break
    if *T_j corresponds to a bin with upper endpoint* max *or T_j is a leaf node* **then**
      for $0 \le k < 2^m$ **do**
        $E_k \leftarrow E_k + \mathbf{T}'_{j,k}$
    else if *$T_j < mid$* **then**
      $\max = \text{mid}$
      $j \leftarrow 2j$
    else
      $\min \leftarrow \text{mid}$
      $j \leftarrow 2j + 1$
      for $0 \le k < 2^m$ **do**
        $E_k \leftarrow E_k + \mathbf{T}'_{2j-1,k}$
  $v \leftarrow 0$
  for $0 \le j < 2^m$ **do**
    $v \leftarrow v + E_i^2 \cdot \sigma^2$
  $\mathbf{v}_i \leftarrow v$
return $\mathbf{v}$

Now that we have a way to estimate the variance of the count at each of the nodes, we need to generate the actual confidence interval. One way to do this is with the same `PostProcessUnion` algorithm used in the binary search approach (Alg. 7); the validity of this interval follows the proof of the algorithm's validity for binary search. However, we can do slightly better here, since the choice of query points is just based on the granularity of the tree's histograms rather than dependent on previous queries. This improved method is described in Alg. 12 and the entire confidence

interval generation process is summarized in Alg. 13.

Algorithm 12: `PostProcess`

Input: $n \in \mathbb{N}$, $\mathscr{R}_{\text{discrete}} \in \mathbb{R}^{2^m}$, noisy CDF counts $\{x, \tilde{C}(x), \sigma_x\}_{x \in \mathscr{R}_{\text{discrete}}}$
for $x \in \mathscr{R}_{\text{discrete}}$ **do**
  $a_x^u = \min\{a \mid \int_q C'_{\text{Bin}}(qn) \cdot \Pr(q + \mathscr{N}(0, \sigma_x^2) > a) \leq \alpha/2\}$ `// can approximate using binary search`
  $a_x^l \leftarrow 1 - a_x^u$
$\ell = \max\{x \in \mathscr{R}_{\text{discrete}} \mid \forall x' \leq x \in \mathscr{R}_{\text{discrete}}, \tilde{C}(x') < a_{x'}^l\}$
$u = \min\{i \in [N] \mid \forall x' \geq x \in \mathscr{R}_{\text{discrete}}, \tilde{C}(x') > a_{x'}^u\}$
return $[\ell, u]$

Algorithm 13: `CDFPostProcess(Union)`: ρ-CDP algorithm

Data: $d = (d_1, \ldots, d_n) \in \mathbb{R}^n$
Privacy params: $\rho \in \mathbb{R}_+$
Hyperparams: $\alpha \in (0,1)$, `Union` $\in \{0,1\}$, $\mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}$, $\theta \in \mathbb{R}_+$, $\gamma \in (0,1)$
$n = |d|$
$m = \lceil \log((r_u - r_\ell)/\theta) \rceil$
$T =$ `DPTree`$(d, \rho, (\mathscr{R}, m))$
$T^* =$ `OptimizedTree`(T, ρ) `// Optimized post-processing algorithm from Honaker [2015] with upper and lower weights as in Lemma D.1`
$\mathscr{R}_{\text{discrete}} = \{r_\ell, r_\ell + \theta, r_\ell + 2\theta, \ldots, r_\ell + 2^m\theta\}$
$(x_i, \tilde{C}(x_i))_{x_i \in \mathscr{R}_{\text{discrete}}} =$ `TreeToCDF`$(n, T^*, \mathscr{R}_{\text{discrete}}, \theta, m)$
$\{\texttt{var}_i\}_{x_i \in \mathscr{R}_{\text{discrete}}} =$ `GetVariances`(T^*, ρ)
if `Union` **then**
  $\beta_1 = \gamma\alpha$
  $\beta_2 = \frac{\alpha - \beta_1}{1 - \beta_1/2}$
  $[l, u] =$ `PostProcessUnion`$(\mathscr{R}_{\text{discrete}}, \{(x, n\tilde{C}(x), n\sigma_x^2\}_{x \in \mathscr{R}_{\text{discrete}}}, \mathtt{N}_L^{\beta_1}, \mathtt{N}_U^{\beta_1}, \beta_2)$ `// Algorithm 7`
else
  $[l, u] =$ `PostProcess`$(n, \mathscr{R}_{\text{discrete}}, (x, \tilde{C}(x), \sigma_x)_{x \in \mathscr{R}_{\text{discrete}}})$
return $[l, u]$

We now need to show that our algorithm is differentially private and that the intervals that Algorithm 13 returns are valid confidence intervals.

Lemma D.2. *Mechanism* `CDFPostProcess(Union)` *(Algorithm 13) is* ρ*-CDP.*

Proof. Note that the only step in `CDFPostProcess(Union)` that touches the dataset d is the call to `DPTree`, which creates a tree of m differentially private histograms. Each histogram is ρ/m-CDP, and by composition ([Bun and Steinke, 2016, Proposition 1.6]), `DPTree` is a ρ-CDP algorithm. The rest of the computations in `CDFPostProcess(Union)` apply post-processing to the output of `DPTree`, so they do not affect the privacy guarantee. □

Lemma D.3. *For any dataset* $d \overset{iid}{\sim} P$*, where* $P \in \Delta_{\mathscr{C}}(\mathbb{R})$*, and any hyperparameters* $\theta, \mathscr{R} = [r_\ell, r_u], \gamma \in (0,1)$*, failure rate* α *and privacy parameter* ρ*, let* `CDFPostProcess`$(d, \rho, (\alpha, \texttt{Union} = 0, \mathscr{R}, \theta, \gamma))$ *return an interval* $[\widetilde{ci}_L^{\alpha}(d), \widetilde{ci}_U^{\alpha}(d)]$*. If* $\texttt{med}(P) \in \mathscr{R}$*, then with probability at least* $1 - \alpha$*,*

$$\texttt{med}(P) \in [\widetilde{ci}_L^{\alpha}(d), \widetilde{ci}_U^{\alpha}(d)]$$

where the probability is taken over the randomness of both the dataset d *and the mechansim* `CDFPostProcess`*.*

Proof. Let us consider the upper endpoint of the interval. First, given a set of DP measurements $\tilde{C}(x) = \hat{C}(x) +$

$\mathcal{N}(0,\sigma_x^2)$, for all $x \in \mathscr{R}_{\text{discrete}}$, recall that we define a_x^u as follows.

$$a_x^u = \min\{a \mid \int_q \Pr_d(\hat{C}(\texttt{med}(P)) = q) \cdot \Pr_{N\sim\mathcal{N}(0,\sigma_x^2)}(q+N > a) \le \alpha/2\}$$

Then, recall that the post-processing algorithm (`PostProcess`) outputs $\widetilde{\texttt{ci}_U^\alpha}(d) = \min\{x \in \mathscr{R}_{\text{discrete}} \mid \forall x' \ge x, \tilde{C}(x') > a_j^u\}$. Let $x^* = \max\{x \in \mathscr{R}_{\text{discrete}} \mid x < \texttt{med}(P)\}$, with corresponding σ_{x^*} and $a_{x^*}^u$. Then, using the subscript A to denote randomness of the DP mechanism, we have that

$$\begin{aligned}
\Pr_{A,d}(\widetilde{\texttt{ci}_U^\alpha}(d) < \texttt{med}(P)) &= \Pr_{A,d}(\min\{x \in \mathscr{R}_{\text{discrete}} \mid \forall x' \ge x\ \ \tilde{C}(x') > a_x^u\} < \texttt{med}(P)) \\
&\le \Pr_{A,d}(\tilde{C}(x^*) > a_{x^*}^u) \\
&= \int_q \Pr_d(\hat{C}(\texttt{med}(P)) = q) \cdot \Pr_A(\tilde{C}(x^*) > a_{x^*}^u \mid \hat{C}(\texttt{med}(P)) = q) \\
&\le \int_q \Pr_d(\hat{C}(\texttt{med}(P)) = q) \cdot \Pr_{N\sim\mathcal{N}(0,\sigma_{x^*}^2)}(q+N > a_{x^*}^u)) \\
&\le \alpha/2
\end{aligned}$$

where the last line follows by definition of $a_{x^*}^u$. A similar argument holds for $\widetilde{\texttt{ci}_L^\alpha}(d)$, so we are done. □

E Details: Range-robust estimator based on CDF estimator, `BinSearch + CDF`

Recall that `NoisyBinSearch` (Algorithm 5) is useful for finding the dataset when it lies within a large range $\mathscr{R}_{\text{large}}$, while `CDFPostProcess` (Algorithm 13) offers highly optimized estimates of the CDF within a small range $\mathscr{R}_{\text{small}}$. The combination `BinSearch + CDF` leverages the strengths of both of these algorithms: it uses `NoisyBinSearch` to narrow down the search space from $\mathscr{R}_{\text{large}} = [r_l, r_u]$ to $\mathscr{R}_{\text{small}} = [r'_l, r'_u]$, clips the data to within $\mathscr{R}_{\text{small}}$, and runs `CDFPostProcess` with the remaining privacy budget within this smaller range to obtain a confidence interval for the population median. The pseudocode for `BinSearch + CDF` is given in Algorithm 14.

The privacy budget ρ and coverage failure probability α both need to be partitioned between the two stages of the algorithm. We expect the optimal split to be distribution dependent. In particular, it likely depends on how large a region the data occupies within the range $\mathscr{R}$. We found experimentally, for the parameter regimes we studied, using $\rho/4$ for the first step, and $3\rho/4$ for the second step ($\gamma = 1/4$) seemed to be a good choice. Similarly, we ensure that the region found in the first step contains the median with probability $1-\alpha/4$, and the second step finds a $1-3\alpha/4$-confidence interval within that region.

Algorithm 14: `BinSearch + CDF`: ρ-CDP Algorithm

Data: $d = (d_1, \ldots, d_n) \in \mathbb{R}^n$
Privacy params: $\rho \in \mathbb{R}_+$
Hyperparams: $\alpha \in (0,1)$, $\mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}$, $\theta \in \mathbb{R}_+, r, r_1, \gamma \in (0,1)$
$n = |d|$
$\rho_{\text{BinSearch}} = \gamma \cdot \rho$
$\alpha_{\texttt{NoisyBinSearch}} = r_1 \cdot \alpha$
$\rho_{\text{CDF}} = (1-\gamma) \cdot \rho$
$\alpha_{\text{CDF}} = (1-r_1) \cdot \alpha$
$[r'_l, r'_u] = \texttt{NoisyBinSearch}(d, \rho_{\text{BinSearch}}, (\alpha_{\texttt{NoisyBinSearch}}, \mathscr{R}, \theta, \gamma, 0.25, 0.75))$
return $\texttt{CDFPostProcess}(d, \rho_{\text{CDF}}, (\alpha_{\text{CDF}}, [r'_l, r'_u], \theta, \gamma))$

Lemma E.1. *Mechanism* `BinSearch + CDF` *(Algorithm 14) is ρ-CDP.*

Proof. BinSearch + CDF is a composition of two algorithms – NoisyBinSearch which by Lemma C.1 is $\gamma\rho$-CDP, and CDFPostProcess which by Lemma D.2 is $(1-\gamma)\rho - CDP$. By Lemma 2.2, this means BinSearch + CDF satisfies ρ-CDP. □

The coverage analysis of BinSearch + CDF follows immediately from Lemma D.3 and Lemma C.2, and a union bound.

Lemma E.2. *Given any dataset $d \overset{i.i.d}{\sim} P^n$, where $P \in \Delta_{\mathscr{C}}(\mathbb{R})$, any hyperparameters $\theta \in \mathbb{R}_+, \mathscr{R} = [r_\ell, r_u] \subset \mathbb{R}, \gamma, r_1, \gamma$, failure rate $\alpha \in (0,1)$ and privacy parameter $\rho \in \mathbb{R}_+$, if $\mathtt{med}(P) \in \mathscr{R}$ then $\mathtt{BinSearch} + \mathtt{CDF}(d, \rho, (\alpha, \theta, \mathscr{R}, \gamma, r_1, \gamma))$ is a valid $1-\alpha$-confidence interval for $\mathtt{med}(P)$.*

Proof. By Lemma C.2, if $\mathtt{med}(P) \in \mathscr{R}$ then $\Pr(\mathtt{med}(P) \in [r'_l, r'_u]) \geq 1 - \alpha_{\mathtt{NoisyBinSearch}}$. Then, by Lemma D.3, $\Pr(\mathtt{med}(P) \in \mathtt{BinSearch} + \mathtt{CDF}(d) \mid \mathtt{med}(P) \in [r'_l, r'_u]) \geq 1 - \alpha_{\mathtt{CDFPostProcess}}$. Therefore,

$$\Pr(\mathtt{med}(P) \in \mathtt{BinSearch} + \mathtt{CDF}(d)) \geq 1 - \alpha_{\mathtt{NoisyBinSearch}} - \alpha_{\mathtt{CDFPostProcess}} = 1 - \alpha.$$

where the probability is over the randomness of both the dataset d and the mechanism $BSCDF$. □

F Details: Other Algorithms Explored

In this section we give a brief overview of additional CDP confidence intervals and CDP median estimators that we explored. These algorithms were not included in the main body of this paper since they are outperformed by other algorithms in every parameter regime we studied. The additional CDP confidence interval algorithms were:

- CDF+BS CI computes a CDP estimate to the empirical CDF in the same way as CDFPostProcess. However, instead of using the post-processing algorithm described in Algorithm 12, it performs binary search using the noisy CDF measurements.
- BinSearch is the same as NoisyBinSearch except it uses the same privacy budget at every iteration. We expect this algorithm to perform strictly worse than NoisyBinSearch which uses its budget more carefully.

Figure 13 shows the performance of CDF+BS CI and BinSearch, as well as the naive estimators ExpMechUnion and CDFPostProcessUnion and the four CDP estimators we presented in the main body. We can see that for all values of ρ, at least one of the four main CDP estimators outperforms each of the other algorithms.

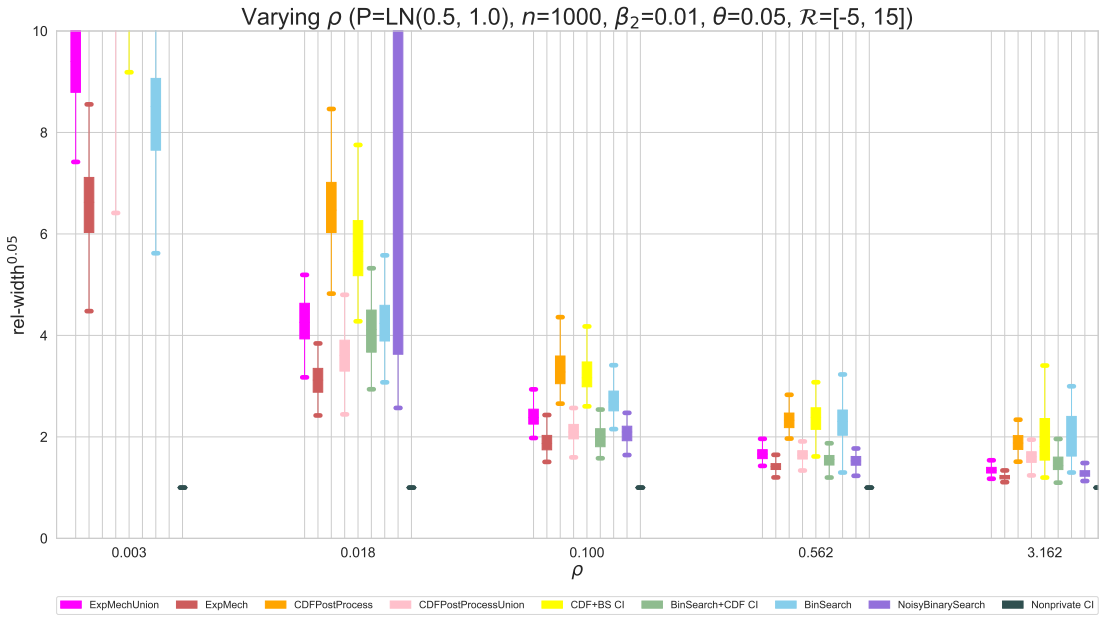


Figure 13: Performance of various CDP confidence intervals for the median as we vary ρ. Performance is measured in terms of the relative width with $\alpha = 0.05$. Box plots are computed using 100 random datasets of 1000 data points drawn i.i.d. from Lognormal$(\ln(1.5), 1)$. Each CDP algorithm is run 5 times on each dataset.

We also explored several CDP median estimators. For point estimators that directly correspond to analogues of our CDP confidence intervals, we use the same name to denote both. However, note that these point estimators for the median are different to the mid-point of the confidence interval estimators that we used in the main body. The algorithms presented in Figure 14 are all directly estimating the median, and do not additionally release a confidence interval for the median.

- `ExpMech` is the point estimator version of our confidence interval algorithm `ExpMech`. It uses the exponential mechanism with target quantile $n/2$ to estimate the median.
- `SmoothSens` releases the median using the smooth sensitivity framework Nissim et al. [2007], Bun and Steinke [2019]. This algorithm Gaussian noise to the empirical median where the standard deviation of the noise is data dependent, and carefully calibrated to ensure differential privacy.
- `BinSearch` is the point estimator version of `BinSearch` described above. It uses binary search with target quantile $n/2$.
- `NoisyStartBinarySearch` was a preliminary version of altering the privacy budget through the iterations of the algorithm, with little budget initially then increasing through the search process.
- `NoisyBinSearch`is the point estimator version of our confidence interval algorithm `NoisyBinSearch`. It uses noisy binary search to search for the quantile $n/2$.
- `FancyBinarySearch` is similar to `BinSearch`. However, instead of halving the range at each iterate, it makes more conservative steps when it is not confident whether the median is to the left or right.
- `CDFPostProcess` is the point estimator version of our confidence interval algorithm `CDFPostProcess`. It computes a CDP estimate the CDF in the same way, then computes the median based on the CDP CDF. An important note is that since both this algorithm and `CDFPostProcess` are post-processing on the CDP CDF estimate, they can be performed at the same time without additional privacy budget.

- `GradDescent` uses CDP gradient descent to solve the optimisation problem $\arg\min \sum_{i=1}^{n} |m - d_i|$. We use the private stochastic gradient descent technique proposed by Bassily et al. [2014].

Figure 14 shows the performance of each of our point estimators on log-normal data. We can see that `SmoothSens`, the only unbiased estimator, has among the highest variability in all regimes, and particularly poor performance for small ρ. It has comparable performance to the other algorithms for large ρ, but extending this point estimator to a confidence interval algorithm remains an open problem. All the variants of binary search perform similarly as point estimators for the median. Even as a point estimator, `ExpMech` slightly outperforms the other algorithms, except `GradDescent`. Extending `GradDescent` to a confidence interval remains an open problem.

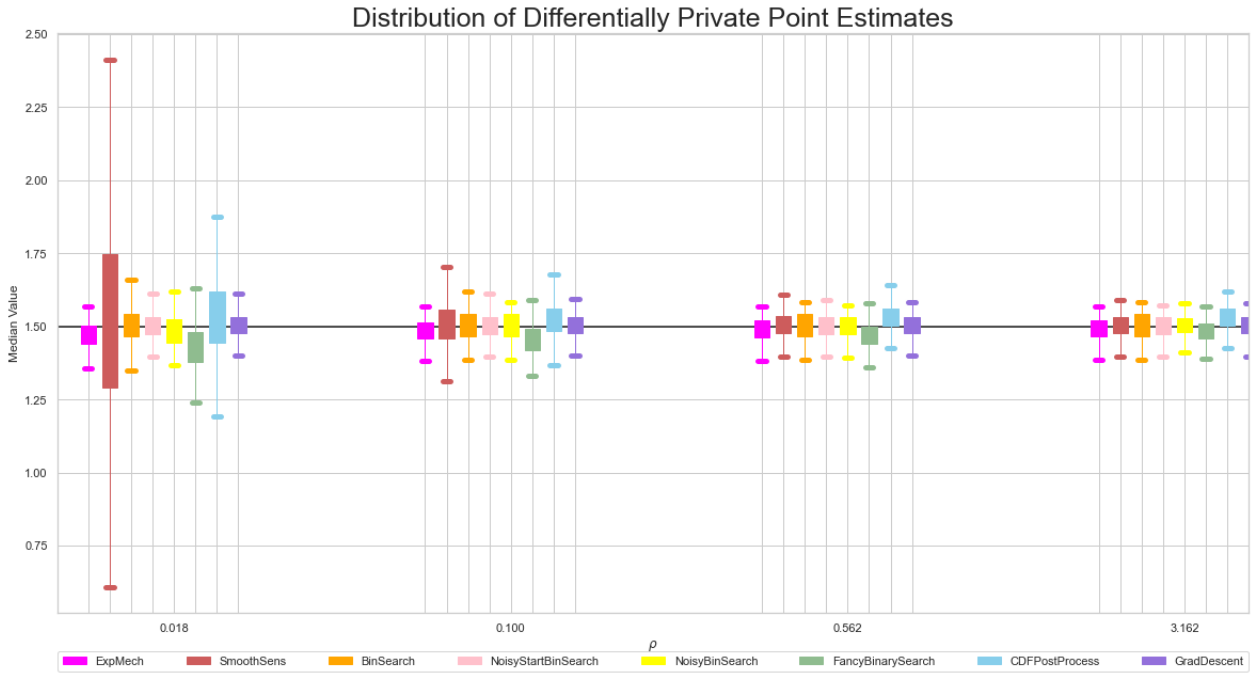
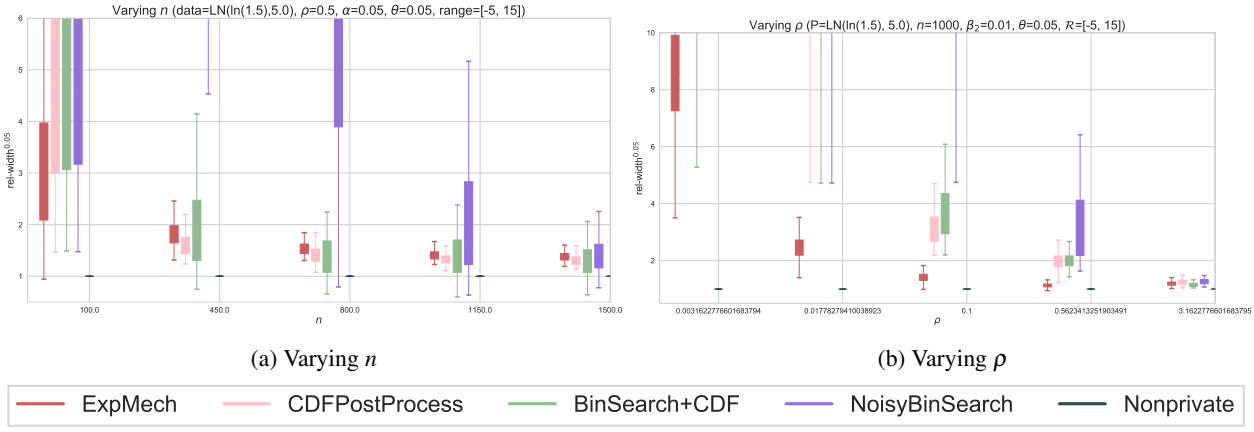


Figure 14: Performance of various CDP point estimators for the median as we vary ρ. Box plots are computed using 100 random datasets of 1000 data points drawn i.i.d. from Lognormal$(\ln(1.5), 1)$. Each CDP algorithm is run 5 times on each dataset.

G Other Regimes

The relative ordering of algorithms can depend on the scale of the data (σ_d) relative to the range. In the figures below, we display the relative widths of the algorithms on data sampled from a Lognormal$(\ln(1.5, \sigma_d^2)$ distribution, where $\sigma_d = 5.0$, as we vary the size of the dataset n and the privacy loss parameter ρ. Note that although we have drastically increased the scale of the data, the range is left the same as in Figure 6: $\mathscr{R} = [-5, 15]$. From these plots, we can see that when n, ρ, and σ_d are large, `CDFPostProcess` performs slightly better than `ExpMech`. In Figure 6, we saw that when σ_d is small, `ExpMech` remains the best performing algorithm in both the large n and large ρ regimes. Hence we conjecture that σ_d needs to be large, and either ρ or n need to be large for `CDFPostProcess` begins outperforming `ExpMech`. This conjecture is supported by Figure 15, where we see `CDFPostProcess` only beginning to outperform `ExpMech` when either n and ρ are large.



(a) Varying n (b) Varying ρ

Figure 15: Relative width of DP confidence intervals on well-spread data ($\sigma_d = 5.0$) as we vary (a) n and (b) ρ.