Noname manuscript No.
(will be inserted by the editor)

Differentially Private Histogram Publication

**Jia Xu · Zhenjie Zhang · Xiaokui Xiao ·
Yin Yang · Ge Yu · Marianne Winslett**

Received: date / Accepted: date

Abstract Differential privacy (DP) is a promising scheme for releasing the results of statistical queries on sensitive data, with strong privacy guarantees against adversaries with arbitrary background knowledge. Existing studies on differential privacy mostly focus on simple aggregations such as counts. This paper investigates the publication of DP-compliant histograms, which is an important analytical tool for showing the distribution of a random variable, e.g., hospital bill size for certain patients. Compared to simple aggregations whose results are purely numerical, a histogram query is inherently more complex, since it must also determine its *structure*, i.e., the ranges of the bins. As we demonstrate in the paper, a DP-compliant histogram with finer bins may actually lead to significantly lower accuracy than a coarser one, since the former requires stronger perturbations in order to satisfy DP. Moreover, the histogram structure itself may reveal sensitive information, which further complicates the problem.

Motivated by this, we propose two novel mechanisms, namely *NoiseFirst* and *StructureFirst*, for computing DP-compliant histograms. Their main difference lies in the relative order of the noise injection and the histogram structure computation steps. NoiseFirst has the additional benefit that it can improve the accuracy of an *already published* DP-complaint histogram computed using a naive method. For each of proposed mechanisms, we design algorithms for computing the optimal histogram structure with two different objectives: minimizing the mean square error and the mean absolute error, respectively. Going one step further, we extend both mechanisms to answer arbitrary range queries. Extensive experiments, using several real data sets, confirm that our two proposals output highly accurate query answers, and consistently outperform existing competitors.

Keywords Differential privacy · query processing · Histogram

J. Xu (✉), G. Yu
College of Information Science and Engineering,
Northeastern University, Shenyang, China
E-mail: {xujia,yuge}@ise.neu.edu.cn

Z. Zhang, Y. Yang
Advanced Digital Sciences Center,
Illinois at Singapore Pte. Ltd, Singapore
E-mail: zhenjie@adsc.com.sg

X. Xiao
School of Computer Engineering,
Nanyang Technological University, Singapore
E-mail: xkxiao@ntu.edu.sg

M. Winslett
Department of Computer Science,
University of Illinois at Urbana-Champaign, Illinois, USA
E-mail: winslett@illinois.edu

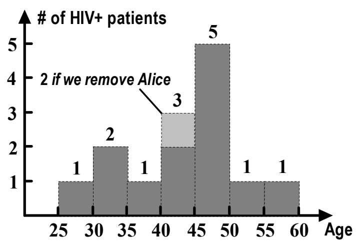
1 Introduction

Digital techniques have enabled various organizations to easily gather vast amounts of personal information, such as medical records, web search history, etc. Analysis on such data can potentially lead to valuable insights, including new understandings of a disease and typical consumer behaviors in a community. However, currently privacy concerns is a major hurdle for such analysis, in two aspects. First, it increases the difficulty for third-party data analyzers to access their input data. For instance, medical researchers are routinely required to obtain the approval of their respective institutional review boards, which is tedious and time-consuming, before they can even look at the data they need. Second, privacy concerns complicate the publication of analysis results. A notable example is the dbGaP[1] database, which contains results of genetic studies. Such results used to be publicly available, until a recent paper [14] describes an attack that infers whether a person has participated in a certain

[1] `http://www.ncbi.nlm.nih.gov/gap`

Name	Age	HIV+
Alice	42	Yes
Bob	31	Yes
Carol	32	Yes
Dave	36	No
Ellen	43	Yes
Frank	41	Yes
Grace	26	Yes
...	...	...

(a) Example sensitive data



(b) Unperturbed histogram

Fig. 1 Example sensitive dataset and its corresponding histogram

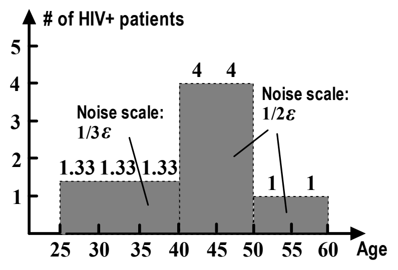
study (e.g., on patients with diabetes) from its results; thereafter, access to such results is strictly controlled. Furthermore, a strengthened version of this attack [22] threatens the publication of any research paper on genome-wide association studies, which currently is an active field in biomedical research.

The recently proposed concept of differential privacy (DP) [7–9, 13, 18, 24] addresses the above issues by injecting a small amount of random noise into statistical results. DP is rapidly gaining popularity, because it provides rigorous privacy guarantees against adversaries with arbitrary background information. This work focuses on the computation of DP-compliant histograms, which is a common tool for presenting the distribution of a random variable. Fig. 1(a) shows sample records in an imaginary sensitive dataset about HIV-positive patients, and Fig. 1(b) illustrates its histogram showing the age distribution of such patients. Such histograms are commonly found, e.g., in the published statistics by Singapore's Ministry of Health[2]. The application of DP to such histograms guarantees that changing or removing any record from the database has negligible impact on the output histogram. This means that the adversary cannot infer whether a specific patient (say, Alice) is infected by HIV, even if s/he knows the HIV status of all the remaining patients in the database.

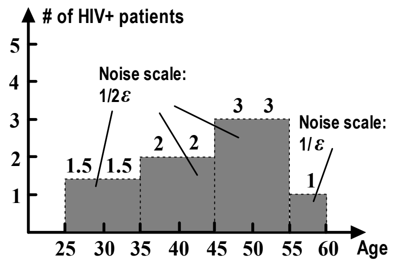
A histogram with a given structure reduces to a set of disjoint range-count queries, one for each bin. The state-of-the-art method [7] (called the Laplace Mechanism, or LM) for perturbing the output of such counts to satisfy DP works as follows. First, LM determines the *sensitivity* Δ of these counts, which is the maximum possible changes in the query results if we remove one record from (or add one into) the database. In our example, we have $\Delta = 1$, since each patient affects the value of exactly one bin in the histogram by at most 1[3]. For instance, removing Alice decreases the number of HIV+ patients aged 40-45 by 1. Then, LM adds to every bin a random value following the Laplace distribution with mean 0 and scale Δ/ϵ, where ϵ is a parameter indicating the level of privacy. For instance, when $\epsilon = 1$, the noise added to each bin has a variance of 2 [17], which intuitively covers the impact (i.e., 1) of any individual in the database.

A key observation made in this paper is that the accuracy of a DP-compliant histogram depends heavily on its structure. In particular, a coarser histogram can sometimes lead to higher accuracy than a finer one, as shown in the example below.

Example 1 Fig. 2(a) exhibits a different histogram of the dataset in Fig. 1(a) with 3 bins 25-40, 40-50 and 50-60, respectively. In the following, we use the term "unit-length range" to mean the range corresponding to a bin in the histogram in Fig. 1(b), e.g., 25-30. In the histogram in Fig. 2(a), each bin covers multiple unit-length ranges, and the numbers on top of each bin correspond to the *mean* count of each unit-length range, e.g., 1.33 above range 25-30 is calculated by dividing the *total* number of patients (i.e., 4) in the bin 25-40 by the number of unit-length ranges it covers (i.e. 3). As we prove later in the paper, such averaging decreases the amount of noise therein. Specifically, the Laplace noise added to each unit-length range inside a bin covering b such ranges has a scale of $1/b \cdot \epsilon$, compared to the $1/\epsilon$ scale in the histogram of Fig. 1(b).



(a) Optimal histogram with 3 bins



(b) Poor histogram structure

Fig. 2 Impact of histogram structures on noise scales

The above example demonstrates that a coarser histogram can lead to a lower amount of *noise* added to each bin. However, the use of larger bins also introduces *information loss*,

[2] http://www.moh.gov.sg/content/moh_web/home/statistics.html

[3] An alternative definition of sensitivity [7] concerns the maximum changes in the query results after *modifying* a record in the database. In our example this leads to $\Delta = 2$, since in the worst case, changing a person's age can affect the values in two different bins by 1 each.

as the mean statistic (e.g., 1.33 for bin 25-30 in Fig. 2(a)) replaces their respective original values (1). Accordingly, the quality of the histogram structure depends on the balancing between the information loss and the error reduction. Fig. 2(b), for example, shows yet another histogram for the same data set in Example 1, which intuitively has poor accuracy because (i) it merges unit-length ranges with very different values, e.g., ranges 35-40 and 40-45, leading to high information loss; and (ii) it contains a very small bin, i.e., 55-60, that requires a high noise scale $1/\epsilon$, especially for small values of ϵ. This example implies that the best structure depends on the data distribution as well as ϵ. A further complication is that if we build the optimal histogram structure on the original count sequence, as shown in Example 1, the optimal structure itself may reveal sensitive information. This is because removing a record from the original database may cause the optimal structure to change, which can be exploited by the adversary to infer sensitive information. Thus, simply selecting the best structure with an existing histogram construction technique violates DP, regardless of the amount of noise injected to the counts.

Facing these challenges, we propose two effective solution frameworks for DP-compliant histogram computation, namely *NoiseFirst* and *StructureFirst*. The former determines the histogram structure *after* the injecting random noise, and the latter derives the histogram structure *before* the addition of the Laplace noise. In particular, NoiseFirst can be used to improve the accuracy of an already published histogram using an existing method [7]. We discuss both methods under both *mean-histograms*, which publishes the mean value for each bin, and *median-histograms*, which reports the median value for each bin. Although median-histograms are rarely used in conventional databases, we show that it often obtains a more accurate representation of the original data under DP requirements.

Furthermore, we adapt DP-histograms to answer arbitrary range-count queries, which has drawn considerable research attention (e.g., [7,13]). For such queries, NoiseFirst achieves better accuracy for short ranges, whereas StructureFirst is more suitable for longer ones. Extensive experiments using several real data sets demonstrate that our two proposals, namely NoiseFirst and StructureFirst output highly accurate histograms, and significantly outperform existing methods for range count queries.

In the remainder of this paper, Section 2 provides necessary background on Differential Privacy(DP). Section 3 and 4 present the solution frameworks of NoiseFirst and StructureFirst, respectively, for mean-histograms . Section 5 extends our solutions to median-histograms. Section 6 discusses the range-count query processing for our proposed methods. Section 7 contains a thorough experimental study. Section 8 overviews existing DP techniques. Finally, Section 9 provides concluding remarks.

2 Preliminaries

2.1 Histogram Construction

Given a series of n counts $D = \{x_1, x_2, \ldots, x_n\}$ and a parameter k, a histogram H merges neighboring counts into k bins $H = \{B_1, B_2, \ldots, B_k\}$, and uses a representative count for each bin. Each bin $B_j = (l_j, r_j, c_j)$ contains an interval $[l_j, r_j] \subseteq [1, n]$, and a count c_j that approximates the counts in D that fall into the interval $[l_j, r_j]$, i.e., $\{x_i \mid l_j \leq i \leq r_j\}$. The bins in a histogram must be disjoint and yet collectively cover all the counts in D. Since a histogram uses fewer counts than the original sequence D, it inevitably introduces *error*. This error is often measured by *Sum of Squared Error* (SSE) between the histogram H and the original sequence D, as follows.

$$SSE(H, D) = \sum_j \sum_{l_j \leq i \leq r_j} (c_j - x_i)^2. \qquad (1)$$

$SSE(H, D)$ can also be interpreted as the sum of squared error for all unit-length range count queries. Given the structure of the histogram, the interval $[l_j, r_j]$ for each bin B_j, the optimal value of c_j for each B_j that minimizes $SSE(H, D)$ is simply the mean value of the counts in $[l_j, r_j]$, i.e., $c_j = \frac{\sum_{i=l_j}^{r_j} x_i}{r_j - l_j + 1}$. Accordingly, the problem of conventional histogram construction [15,12] aims to identify the optimal histogram structure H^* containing k bins (k is a given parameter) that minimizes $SSE(H, D)$.

Jagadish et al. [15] propose a dynamic programming solution, with time complexity $O(n^2k)$ and space complexity $O(nk)$. Fig. 3 lists the intermediate results of the dynamic programming process, using the data sequence in Fig. 1(b) and setting $k = 3$. Each entry in the i-th column and k-th row, denoted as $T(i, k)$, represents the minimum error of any histogram with k bins covering the prefix $D_i = \{x_1, \ldots, x_i\}$. Let $SSE(D, l, r)$ to denote the sum of squared error incurred by merging a partial sequence $\{x_l, \ldots, x_r\}$ into a single bin. The mean count for this merged bin is then $\bar{x}(l, r) = \sum_{i=l}^{r} x_i/(r - l + 1)$. Therefore, $SSE(D, l, r) = \sum_{i=l}^{r}(x_i - \bar{x}(l, r))^2$.

i=1	2	3	4	5	6	7	
0	**0.5**	**0.67**	**2.75**	**11.2**	**12.8**	**14**	***k*=1**
	0	**0.5**	**0.67**	**2.67**	**8.67**	**11.2**	**2**
		0	**0.5**	**0.67**	**2.67**	**2.67**	**3**

Fig. 3 Building the optimal histogram for the dataset in Figure 1(b)

The dynamic programming algorithm in [15] recursively computes the minimum SSE $T(n, k)$ of the optimal histogram,

Table 1 Summary of frequent notations

Δ	query sensitivity
$D = \{x_1, \dots, x_i, \dots, x_n\}$	a count sequence
$D' = \{x_1, \dots, x_i \pm 1, \dots, x_n\}$	a neighbor count sequence of D
$\hat{D} = \{\hat{x}_1, \dots, \hat{x}_n\}$	a noisy count sequence of D
$D_i = \{x_1, \dots, x_i\}$	a partial sequence of D
H_k^*	the optimal k-bin histogram on D
$\hat{H}_k^*$	the optimal k-bin histogram on the noisy count sequence $\hat{D}$
$H^*(D_i, k)$	the optimal k-bin histogram on the partial sequence D_i
$SSE(D, l, r)/SAE(D, l, r)$	the SSE/SAE error if we merge a partial sequence $\{x_l, \dots, x_r\}$ into a single bin
$SSE(H, D)/SAE(H, D)$	the SSE/SAE error if we build histogram H on D

using the following equation:

$$T(i, k) = \min_{k-1 \le j \le i-1} (T(j, k-1) + SSE(D, j+1, i)) .$$

Given the minimum SSE values, we can determine the optimal histogram structure by tracing back the selections of the optimal bin boundaries (shown as gray cells in Figure 2(a)). In our example, the optimal histogram is $H^* = \{$(1, 3, 1.33), (4, 5, 4.5), (6, 7, 1.0)$\}$.

2.2 Differential Privacy

Given a count sequence $D = \{x_1, x_2, \dots, x_n\}$, another sequence D' is a neighbor sequence to D, if and only if D' differs from D in only one count, and the difference in that count is exactly 1. Formally, there exists an integer $1 \le m \le n$, such that $D' = \{x_1, x_2, \dots, x_{m-1}, x_m \pm 1, x_{m+1}, \dots, x_n\}$. A histogram publication mechanism Q satisfies ϵ-differential privacy (ϵ-DP), if it outputs a randomized histogram H, such that

$$\forall D, D', H : \quad \Pr(Q(D) = H) \le e^{\epsilon} \times \Pr(Q(D') = H),$$

where D and D' denote two arbitrary neighbor sequences, and $\Pr(Q(D) = H)$ denotes the probability that Q outputs H with input D. The first and most commonly used mechanism for differential privacy is the Laplace mechanism [7], which relies on the concept of *sensitivity*. In particular, the sensitivity Δ of the query (e.g., a histogram query in our problem) is defined as the maximum L_1-distance between the exact answers of the query Q on any neighbor databases D and D', i.e.,

$$\Delta = \max_{D, D'} \|Q(D) - Q(D')\|_1$$

Dwork et al. [7] prove that differential privacy can be achieved by adding random noise to the each output of Q that follows a zero-mean Laplace distribution with scale $b = \frac{\Delta}{\epsilon}$. In our problem, note that the simple solution that simply injects Laplace noise to each bin count of the optimal histogram (e.g., computed with the algorithm in [15]) does not satisfy differential privacy, because the *structure* of the optimal histogram depends on the original data. Consequently, the adversary can infer sensitive information based on the optimal histogram structure. For instance, consider again the example in Fig. 1(b), and assume that the adversary knows all the AIDS patients except for Alice. If Alice were not an HIV+ patient, there would be only two patients between 40 and 45 years old, which leads to a different optimal 3-bin histogram as shown in Figure 4. Since the published optimal histogram structure is the one shown in Figure 2(a), the adversary infers that *Alice must be an HIV+ patient*, leading to a privacy breach.

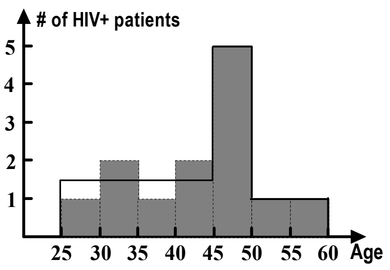


Fig. 4 Optimal histogram when Alice is excluded from the dataset

Table 1 summarizes frequently used notations throughout the paper.

3 NoiseFirst

In this and the next section, we present the proposed solutions under the mean-histogram, in which each bin outputs the mean value of the unit-length counts therein. Our first solution, NoiseFirst, involves two steps. First, it computes a differentially private histogram with the finest granularity, i.e., with unit-length bins, using the Laplace Mechanism (LM) [7]. Clearly, the sensitivity of this step is exactly 1, since adding or removing a record can change the count of one bin by 1. Therefore, it suffices to inject Laplace noise with magnitude $b = \frac{1}{\epsilon}$ into each bin to satisfy ϵ-DP. The result is a noisy sequence $\hat{D} = \{\hat{x}_1, \dots, \hat{x}_n\}$.

In the second step, NoiseFirst computes the optimal histogram structure based on the noisy sequence $\hat{D}$, using the dynamic programming algorithm [15]. The complete pseudocode of NoiseFirst is listed in Algorithm 1. Clearly, NoiseFirst can be used as a post-processing step for optimizing an already published histogram $\hat{D}$ computed by LM, by merging adjacent noisy counts.

Algorithm 1 Mean-NoiseFirst (count sequence D, the number of bins k, privacy guarantee ϵ)

```
1: Generate a new database $\hat{D}$ by adding independent $Lap(\frac{1}{\epsilon})$ on every count $x_i \in D$.
2: Build the optimal $k$-bin histogram on $\hat{D}$, denoted as $\hat{H}_k^*$, with dynamic programming method.
3: Return histogram $\hat{H}_k^* = \{(l_1, r_1, c_1), \ldots, (l_k, r_k, c_k)\}$, in which every $c_j$ is the mean of the noisy data $\{\hat{x}_{l_j}, \ldots, \hat{x}_{r_j}\}$.
```

Since NoiseFirst computes the histogram structure based on the noisy counts, a natural question is whether the resulting histogram indeed has high quality. To answer this, in the rest of the section, we provide some theoretical analysis on the expected Sum of Squared Error(SSE) incurred by NoiseFirst. The main objective in the analysis is to derive the connection between (i) the error of the histogram built with the noisy count sequence $\hat{D}$ and (ii) the error of the optimal histogram built with the original data D. Note that (ii) is only used in the analysis; NoiseFirst does not compute or report such a histogram. First, we analyze the impact of merging consecutive noisy counts into a single bin. The following lemmata quantify the expected errors of a bin on the noisy sequence and original sequence, respectively.

Lemma 1 *Given subsequence of counts $\{x_l, x_{l+1}, \ldots, x_r\}$. Let $\{\hat{x}_l, \ldots, \hat{x}_r\}$ be the noisy counts after the first step in Algorithm 1. If (l, r, c) is the result bin by merging all the counts, the expected squared error of the bin with respect to $\{\hat{x}_l, \ldots, \hat{x}_r\}$ is*

$$\mathbb{E}\left(Error((l,r,c), \{\hat{x}_l, \ldots, \hat{x}_r\}\right) = SSE(D,l,r) + \frac{2(r-l)}{\epsilon^2}$$

Proof Assume that X_i is the variable of the count x_i after adding noise following $Lap(\frac{1}{\epsilon})$. Let C be the variable of the average count c, i.e., $C = \frac{\sum_{i=l}^{r} X_i}{r-l+1}$. We use δ_i to denote the residual variable on X_i, i.e., $\delta_i = X_i - x_i$, and let the size of the bin $s = r - l + 1$. The expected error of the result bin (l, r, c) on the noisy count sequence $\{\hat{x}_l, \ldots, \hat{x}_r\}$ is thus derived as follows.

$$\mathbb{E}\left\{\sum_{i=l}^{r}(X_i - C)^2\right\} = \mathbb{E}\left\{\sum_{i=l}^{r}(X_i)^2 - \frac{1}{s}\left(\sum_{i=1}^{r} X_i\right)^2\right\}$$
$$= \mathbb{E}\left\{\sum_{i=l}^{r}(x_i)^2 + \sum_{i=l}^{r}(\delta_i)^2 - \frac{1}{s}\left(\sum_{i=l}^{r} x_i\right)^2 - \frac{1}{s}\left(\sum_{i=l}^{r}\delta_i\right)^2\right\}$$
$$= \sum_{i=l}^{r}(x_i)^2 - \frac{1}{s}\left(\sum_{i=l}^{r} x_i\right)^2 + \mathbb{E}\left\{\sum_{i=l}^{r}(\delta_i)^2 - \frac{1}{s}\left(\sum_{i=l}^{r}\delta_i\right)^2\right\}$$
$$= SSE(D,l,r) + \mathbb{E}\left\{\sum_{i=l}^{r}(\delta_i)^2 - \frac{1}{s}\sum_{i=l}^{r}(\delta_i)^2\right\}$$
$$= SSE(D,l,r) + \frac{2(s-1)}{\epsilon^2}$$

Since $s = r - l + 1$, we reach the conclusion in the lemma. □

The above is the sum of squared error computed using the the average count in the noisy data $\hat{D}$. The following lemma shows how to estimate the error of a bin with respect to the original counts in D.

Lemma 2 *Given subsequence of counts $\{x_l, x_{l+1}, \ldots, x_r\}$. Let $\{\hat{x}_l, \ldots, \hat{x}_r\}$ be the noisy counts after the first step in Algorithm 1. If (l, r, c) is the result bin by merging all the counts, the expected squared error of the bin with respect to $\{x_l, \ldots, x_r\}$ is*

$$\mathbb{E}\left(Error(\{(l,r,c\}, \{x_l, \ldots, x_r\})\right) = SSE(D,l,r) + \frac{2}{\epsilon^2}$$

Proof Similar to the proof of Lemma 1, we derive the error as follows. Notations X_i, C and s have the same meanings as defined in the proof for Lemma 1.

$$\mathbb{E}\left\{\sum_{i=l}^{r}(x_i - C)^2\right\}$$
$$= \sum_{i=l}^{r}(x_i)^2 - \frac{2}{s}\left(\sum_{i=l}^{r} x_i\right)^2 + \frac{1}{s}\mathbb{E}\left\{\left(\sum_{i=l}^{r} X_i\right)^2\right\}$$
$$= SSE(D,l,r) - \frac{1}{s}\left(\sum_{i=l}^{r} x_i\right)^2 + \frac{1}{s}\mathbb{E}\left\{\left(\sum_{i=l}^{r}(x_i + \delta_i)\right)^2\right\}$$
$$= SSE(D,l,r) + \frac{1}{s}\mathbb{E}\left\{\sum_{i=l}^{r}(\delta_i)^2\right\}$$
$$= SSE(D,l,r) + \frac{2}{\epsilon^2}$$

This completes the proof. □

The expected SSE of the histogram with the finest granularity (i.e., with unit-length bins)is exactly $\frac{2(r-l+1)}{\epsilon^2}$, i.e.,

$$\mathbb{E}\left(Error(\{(l,l,\hat{x}_l), \ldots, (r,r,\hat{x}_r)\}, \{x_l, \ldots, x_r\})\right) \quad (2)$$
$$= \frac{2(r-l+1)}{\epsilon^2}$$

Lemma 2 shows that the expected error of histogram with a single bin is $SSE(D, l, r) + \frac{2}{\epsilon^2}$. This implies that the accuracy of the coarsest (i.e., single-bin) histogram is more accurate than the finest one (i.e., unit-bin), when ϵ is sufficiently small, i.e., $\epsilon < \sqrt{\frac{2(r-l)}{SSE(D,l,r)}}$. Next we extend the analysis of Lemma 1 and Lemma 2 to multiple bins.

Let H_k denote the k-bin histogram with the same structure as $\hat{H}_k^*$ but with different counts in the bins. Instead of using noisy counts $\{\hat{x}_{l_j}, \ldots, \hat{x}_{r_j}\}$, H_k calculates the mean count for bin B_j with the original counts $(x_{l_j} + \ldots + x_{r_j})/(r_j - l_j + 1)$. It is straightforward to see that $Error(H_k, D) = \sum_j^k SSE(D, l_j, r_j)$, which helps proving the following theorem.

Theorem 1 *Given H_k and $\hat{H}_k^*$ as defined above, the expected error of the histogram on $\hat{D}$ and D are respectively,*

$$\mathbb{E}(Error(\hat{H}_k^*, \hat{D})) = Error(H_k, D) + \frac{2(n-k)}{\epsilon^2}$$

$$\mathbb{E}(Error(\hat{H}_k^*, D)) = Error(H_k, D) + \frac{2k}{\epsilon^2}$$

Since k is fixed before running the algorithm, the theorem above implies that optimizing the histogram by minimizing on $Error(\hat{H}_k^*, \hat{D})$ leads to the solution that minimizes $Error(\hat{H}_k^*, D)$, i.e.,

$$\arg\min_{\hat{H}_k^*} \mathbb{E}(Error(\hat{H}_k^*, \hat{D})) = \arg\min_{\hat{H}_k^*} \mathbb{E}(Error(\hat{H}_k^*, D)) \tag{3}$$

The equation provides the intuition behind the correctness of NoiseFirst method in Algorithm 1, which runs the dynamic programming on the noisy data sequence $\hat{D}$.

It remains to clarify how to select an appropriate value for parameter k. Since the noisy data already satisfies ϵ-DP, and NoisyFirst does not require any additional privacy budget, we simply execute it n times, with parameter values $k = 1, \ldots, n$ and returning the result with the minimum expected SSE. Although it is impossible to directly evaluate the expected error on D, we can utilize Theorem 1 again. If $\hat{H}^*$ is the optimal histogram returned by Algorithm 1 with $k = 1, \ldots n$ bins, the final result histogram is selected based on the following optimization objective:

$$\hat{H}^* = \arg\min_{\hat{H}_k^*} \mathbb{E}\left(Error(\hat{H}_k^*, \hat{D}) - \frac{2n-4k}{\epsilon^2}\right) \tag{4}$$

Algorithm 2 details the computation of the optimal parameter k^* for NoiseFirst. We use the notation $\hat{T}\mathrm{SSE}[i, j]$ to represent the minimum SSE of merging the partial noisy count sequence $\hat{D}_i$ into j bins. The algorithm first fills the table $\hat{T}\mathrm{SSE}$ (Lines 1-6) using dynamic programming. Then, k^* is obtained by comparing the expected SSE under different ks based on the Equation 3 (Lines 7-12). Since Algorithm 2 runs entirely on the noisy count sequence $\hat{D}$ rather than the original sequence D, it does not consumes any privacy budget, either.

Algorithm 2 ComputeOptimalK (noisy count sequence $\hat{D}$ with $|\hat{D}| = n$, privacy guarantee ϵ)

```
1: for each $i$ from 1 up to $n$ do
2:     $\hat{T}\mathrm{SSE}[i, 1] := SSE(D, 1, i)$
3:     for each $j$ from 2 to $n$ do
4:         $\hat{T}\mathrm{SSE}[i, j] := +\infty$
5:         for each $l := j - 1$ down to 1 do
6:             $\hat{T}\mathrm{SSE}[i, j] :=$ min ($\hat{T}\mathrm{SSE}[i, j]$,
                       $\hat{T}\mathrm{SSE}[l, j-1] + SSE(\hat{D}, l+1, i)$)
7: $minErr = MAX$
8: for each $k$ from 1 up to $n$ do
9:     $Err = \hat{T}\mathrm{SSE}[n, k] - \frac{2n-4k}{\epsilon^2}$
10:    if $Err < minErr$ then
11:        $minErr = Err$
12:        $k^* = k$
13: Return $k^*$
```

NoiseFirst obtains improved accuracy by canceling out the zero-mean Laplace noise injected into different bins. On the other hand, NoiseFirst has the same sensitivity as LM, and injects $Lap(\frac{1}{\epsilon})$ in each bin during its first step. As shown in the Example 1 of Section 1, the sensitivity of the histogram can be further reduced, if we add noise *after* the histogram construction. Next we describe a novel algorithm based on this idea.

4 StructureFirst

Unlike NoiseFirst, the StructureFirst algorithm computes the histogram structure using the original data, before adding the noise to each of its bins. Note that the optimal histogram structure is sensitive information itself; hence, StructureFirst spends a portion of the privacy budget ϵ to protect it. In the rest of the section, we use D_i to denote the partial database $D_i = \{x_1, x_2, \ldots, x_i\}$, and $H^*(D_i, j)$ to denote the optimal histogram with j bins on D_i.

Algorithm 3 presents the StructureFirst algorithm. It first constructs the optimal histogram H^* on the basis of the original data D. All the intermediate results are stored in a table, as is done in Figure 3. To protect the structure of the optimal histogram, StructureFirst randomly moves the boundaries between the bins. When choosing the boundary between B_j and B_{j+1}, it picks up r_j for bin B_j at the count $x_q \in D$ with probability proportional to

$$\exp\left\{-\frac{\epsilon_1 E_{SSE}(q, j, r_{j+1})}{2(k-1)(2F+1)}\right\}$$

Here, $E_{SSE}(q, j, r_{j+1})$ is the SSE of the histogram, which consists of the optimal histogram with j bins covering $[1, q]$ and the $j+1$th bin covering $[q+1, r_{j+1}]$. If $(q+1, r_{j+1}, c)$ is a new bin constructed by merging counts $\{x_{q+1}, \ldots, x_{r_{j+1}}\}$

Algorithm 3 Mean-StructureFirst (count sequence D, the number bins k, privacy parameter ϵ, count upper bound F)

```
1: Partition $\epsilon$ into two parts $\epsilon_1$ and $\epsilon_2$ that $\epsilon_1 + \epsilon_2 = \epsilon$.
2: Build the optimal histogram $H_k^*$ and keep all intermediate results,
   i.e. $Error(H^*(D_i, j), D_i)$ for all $1 \le i \le n$ and $1 \le j \le k$
3: Set right boundary of $k$th bin at $r_k = n$
4: for each $j$ from $k-1$ down to $1$ do
5:     for each $q$ from $j$ up to $r_{j+1} - 1$ do
6:         Calculate $E_{SSE}(q, j, r_{j+1}) = Error(H^*(D_q, j), D_q) + SSE(D, q+1, r_{j+1})$
7:     select $r_j = q$ from $k-1 \le q < r_{j+1}$ with probability
       proportional to $\exp\left\{-\frac{\epsilon_1 E_{SSE}(q,j,r_{j+1})}{2(k-1)(2F+1)}\right\}$
8:     Set $l_{j+1} = r_j + 1$
9: Calculate average count $c_j$ for every bin on interval $[l_j, r_j]$.
10: Add noise to counts as $\hat{c}_j = c_j + Lap\left(\frac{1}{\epsilon_2(r_j - l_j + 1)}\right)$
11: Return histogram $\{(l_1, r_1, \hat{c}_1), \dots, (l_k, r_k, \hat{c}_k)\}$
```

and c is the average count, it is straightforward to verify that

$$\begin{aligned} & E_{SSE}(q, j, r_{j+1}) \\ & = Error(H^*(D_q, j) \cup \{(q+1, r_{j+1}, c)\}, D_{r_{j+1}}) \\ & = Error(H^*(D_q, j), D_q) + SSE(D, q+1, r_{j+1}) \end{aligned} \tag{5}$$

In the probability function, F is some prior knowledge on the upper bound on the maximum count in the sequence, which is assumed to be independent of the database. After setting all the boundaries, Laplace noise is added on count in each bin. For bin (l_j, r_j, c_j), the magnitude of the Laplace noise on c_j is $\frac{1}{\epsilon_2(r_j - l_j + 1)}$.

Next we analyze the correctness and expected accuracy of the algorithm, and discuss how to set the values of ϵ_1 and ϵ_2. Unlike NoiseFirst, StructureFirst does not decide on the value of k itself; instead, the user must specify the desired k before running the algorithm. In Section 7, we provide some empirical guidelines on how to choose generally good k value on real data sets.

4.1 Correctness

To pave the way for the correctness proof, we first derive some worst case sensitivity analysis on the intermediate results during the computation of the histogram structure.

Lemma 3 *Let F be an upper bound of the maximum count in any of the bins. Given two neighbor database D and D', the error of the optimal histograms on D and D' with respect to the first i counts changes by at most*

$$|Error(H^*(D'_i, j), D'_i) - Error(H^*(D_i, j), D_i)| \le 2F + 1$$

Proof Given two neighbor databases D and D', there is exactly one count x_m changes by 1, i.e., $x_m - 1 \le x'_m \le x_m + 1$. Assume that x_m is in bin (l_z, r_z, c_z) in $H^*(D_i, j)$. Since $H^*(D'_i, j)$ is the optimal histogram for D'_i with j bins, it is straightforward to know that given any histogram H' covering interval $[1, i]$, we have

$$Error(H^*(D'_i, j), D'_i) \le Error(H', D'_i) \tag{6}$$

In particular, we construct a special histogram H' by reusing all bins in $H^*(D_i, j)$, except that $c'_z = \frac{\sum_{q=l_z}^{r_z} x_q + (x'_m - x_m)}{r_z - l_z + 1}$ replaces $c_z = \frac{\sum_{q=l_z}^{r_z} x_q}{r_z - l_z + 1}$. Let $s = r_z - l_z + 1$. In the following, we derive some upper bound on the error of such H' on D'_i.

$$\begin{aligned} & Error(H', D'_i) - Error(H^*(D_i, j), D_i) \\ & = (x'_m)^2 - (x_m)^2 - s(c'_z)^2 + s(c_z)^2 \end{aligned} \tag{7}$$

When $x'_m = x_m + 1$, the difference on the error above is

$$\begin{aligned} & Error(H', D'_i) - Error(H^*(D_i, j), D_i) \\ & = (x_m + 1)^2 - (x_m)^2 - s\left(c_z + \frac{1}{s}\right)^2 + s(c_z)^2 \\ & = 2x_m + 1 - 2c_z - \frac{1}{s} \\ & \le 2x_m + 1 \end{aligned} \tag{8}$$

When $x'_m = x_m - 1$, the difference can be estimated similarly as

$$\begin{aligned} & Error(H', D'_i) - Error(H^*(D_i, j), D_i) \\ & = (x_m - 1)^2 - (x_m)^2 - s\left(c_z - \frac{1}{s}\right)^2 + s(c_z)^2 \\ & = -2x_m + 1 + 2c_z - \frac{1}{s} \\ & \le 2c_z + 1 \\ & \le 2 \max_{l_s \le q \le r_s} x_q + 1 \end{aligned} \tag{9}$$

Combining Equations 8 and 9, we have
$Error(H^*(D'_i, j), D'_i) \le Error(H', D'_i)$
$\le Error(H^*(D_i, j), D_i) + 2F + 1$
This reaches the conclusion of the lemma. □

Given the above worst case analysis on the change of error, we next prove that the bin-boundary perturbation scheme satisfies differential privacy.

Lemma 4 *The selection of r_j on line 7 of Algorithm 3 satisfies $\frac{\epsilon_1}{k-1}$-differential privacy.*

Proof Let $E_{SSE}(q, j, r_{j+1})$ be the SSE cost of setting $r_j = q$, we use $E'_{SSE}(q, j, r_{j+1})$ to denote the cost when the same histogram is constructed on D' instead. Based on Lemma 3, we have

$$|E'_{SSE}(q, j, r_{j+1}) - E_{SSE}(q, j, r_{j+1})| \le 2F + 1$$

According to the pseudocodes of StructureFirst, when r_{j+1} is fixed, the probability of selecting $r_j = q$ on D' is

$$\begin{aligned}\Pr(r_j = q) &= \frac{\exp\left\{-\frac{\epsilon_1 E'_{SSE}(q,j,r_{j+1})}{2(k-1)(2F+1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1 E'_{SSE}(z,j,r_{j+1})}{2(k-1)(2F+1)}\right\}}\\ &\le \frac{\exp\left\{-\frac{\epsilon_1 (E_{SSE}(q,j,r_{j+1})-2F-1)}{2(k-1)(2F+1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1 (E_{SSE}(z,j,r_{j+1})+2F+1)}{2(k-1)(2F+1)}\right\}}\\ &= \exp\left(\frac{\epsilon_1}{k-1}\right)\frac{\exp\left\{-\frac{\epsilon_1 E_{SSE}(q,j,r_{j+1})}{2(k-1)(2F+1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1 E_{SSE}(z,j,r_{j+1})}{2(k-1)(2F+1)}\right\}}\end{aligned}$$

Hence, the selection of r_j follows $(\frac{\epsilon_1}{k-1})$-differential privacy. □

Based on the above lemmata, we prove the correctness of Algorithm 3 in the following theorem.

Theorem 2 *Algorithm 3 satisfies ϵ-differential privacy.*

Proof In Algorithm 3, the output histogram relies on the results on Line 7 and Line 12, which are all independent of each other. Line 7 is run k times and Line 12 is run exactly once. According to Lemma 4, each execution on line 7 costs the privacy budget $\frac{\epsilon_1}{k}$. Based on the concept of generalized sensitivity used in [24], the privacy cost of line 12 is ϵ_2. It is because we can define $W_j = (r_j - l_j + 1)$ to meet the requirement of generalized sensitivity. Consider two databases D and D' with the same boundary structure calculated in the StructureFirst algorithm, we have $\hat{c}_j$ and $\hat{c}'_j$ being the noisy average counts. They satisfy the following function in form of weighted sensitivity.

$$\sum_j W_j \cdot \hat{c}_j - \sum_j W_j \cdot \hat{c}'_j \le 1$$

The total privacy budget spent in the algorithm is thus $k\frac{\epsilon_1}{k} + \epsilon_2 = \epsilon$, since $\epsilon_1 + \epsilon_2 = \epsilon$. □

4.2 Accuracy Analysis

Next we quantify the expected error incurred by StructureFirst.

Lemma 5 *Given any non-negative real number x and positive constant c, we have $x \cdot e^{-cx} \le \frac{1}{2c}$.*

Lemma 6 *The SSE of the histogram increases by no more than $\frac{8(k-1)(2F+1)^2}{\epsilon_1(8(k-1)(2F+1)-\epsilon_1 nF^2)}$, every time we move the boundary by line 7 in Algorithm 3.*

Proof Assume that the optimal histogram with $j+1$ bins is supposed to selection q^* as the boundary between j^{th} bin and $(j+1)^{th}$ bin, to minimize the error $Error(H^*(D_q,j),D_q) + SSE(q+1,r_{j+1})$. We are interested in the expected increase on the error when the randomized algorithm fails to select q^*. Note that the probabilities remain the same if we reduce each $E_{SSE}(q,j,r_{j+1})$ by $E_{SSE}(q^*,j,r_{j+1})$. Since each $E_{SSE}(q) - E(q^*) \ge 0$ and there is at least one $z = q^*$ that $E_{SSE}(z,j,r_{j+1}) - E_{SSE}(q^*,j,r_{j+1}) = 0$, the following inequalities must be valid, using the well known fact $e^{-x} \ge 1 - x$ for any positive x.

$$\begin{aligned}&\sum_z \exp\left\{-\frac{\epsilon_1(E_{SSE}(z,j,r_{j+1}) - E_{SSE}(q^*,j,r_{j+1}))}{2(k-1)(2F+1)}\right\}\\ &\ge \sum_z \left(1 - \frac{\epsilon_1(E_{SSE}(z,j,r_{j+1}) - E_{SSE}(q^*,j,r_{j+1}))}{2(k-1)(2F+1)}\right)\\ &\ge n\left(1 - \frac{\epsilon_1 nF^2}{8(k-1)(2F+1)}\right) \qquad (10)\end{aligned}$$

To simplify the notation, we use $E(q)$ to replace $E_{SSE}(q,j,r_{j+1})$, when j and r_{j+1} are clear in the context. Thus, the expectation on the additional error over the optimal one with q^* is

$$\begin{aligned}&\sum_q \{(E(q))\Pr(r_j = q)\} - E(q^*,j,r_{j+1})\\ &= \sum_q \{(E(q) - E(q^*))\Pr(r_j = q)\}\\ &= \sum_q \left\{(E(q) - E(q^*))\frac{\exp\left\{-\frac{\epsilon_1(E(q)-E(q^*))}{2(k-1)(2F+1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1(E(z)-E(q^*))}{2(k-1)(2F+1)}\right\}}\right\}\\ &\le \sum_q \left\{(E(q) - E(q^*))\frac{\exp\left\{-\frac{\epsilon_1(E(q)-E(q^*))}{2(k-1)(2F+1)}\right\}}{n - \frac{\epsilon_1 n^2F^2}{8(k-1)(2F+1)}}\right\}\\ &\le \sum_q \frac{(k-1)(2F+1)}{\epsilon_1}\left(n - \frac{\epsilon_1 n^2F^2}{8(k-1)(2F+1)}\right)^{-1}\\ &\le \frac{8(k-1)^2(2F+1)^2}{\epsilon_1(8(k-1)(2F+1) - \epsilon_1 nF^2)} \qquad (11)\end{aligned}$$

Here, the first inequality is due to Equation 10. The second inequality applies to Lemma 5. The last inequality is due to the fact that the number of candidate q is no larger than the size of the database. □

Note that the cost analysis in Lemma 6 does not rely on j or r_{j+1}. Therefore, the total additional SSE over the optimal histogram without noise is simply $k-1$ times of the error in Lemma 6. Combined with the error introduced on line 10 in Algorithm 3, the total expected SSE is bounded, as stated in the following theorem.

Theorem 3 *The output histogram of Algorithm 3, the expected SSE is at most*

$$SSE(H^*(D,k),D) + \frac{8(k-1)^3(2F+1)^2}{\epsilon_1(8(k-1)(2F+1) - \epsilon_1 nF^2)} + \frac{2k}{(\epsilon_2)^2}$$

The error analysis in the theorem above shows the advantages of StructureFirst method. We can interpret the expected SSE of StructureFirst as $Error(H^*(D,k),D)+O(\frac{k}{\epsilon^2})$, by ignoring all constant items. This is a significant theoretical improvement over all previous methods, including NoiseFirst.

4.3 Budget Assignment

Given k, the expected SSE of the histogram obtained by StructureFirst depends upon the values of ϵ_1 and $\epsilon_2 = \epsilon - \epsilon_1$. In the error analysis of Theorem 3, the total error consists of three parts. The first part $Error(H^*(D,k),D)$ relies on k and the original data D. The other two parts are independent of the data, but are decided by n, k, F, ϵ_1 and ϵ_2. Therefore, we minimize the expected error by finding the optimal combination of ϵ_1 and ϵ_2:

$$\text{Minimize} \quad \left(\frac{8(k-1)^3(2F+1)^2}{\epsilon_1\left(8(k-1)(2F+1) - \epsilon_1 n F^2\right)} + \frac{2k}{(\epsilon_2)^2} \right)$$
$$\text{s.t.} \quad \epsilon_1 + \epsilon_2 = \epsilon$$

The optimization problem above does not always has a closed-form solution. Hence, we employ numerical methods to identify a near-optimal assignment on ϵ_1 and ϵ_2. Specifically, we apply a simple sampling technique that tries different $0 < \epsilon_1 < \epsilon$ and returns the best ϵ_1 and $\epsilon_2 = \epsilon - \epsilon_1$ encountered. Since it takes constant time to verify the cost when a specific pair of ϵ_1 and ϵ_2 are given, the computational overhead of the budget assignment optimization is insignificant.

5 Publishing Median-Histograms

So far, we have discussed NoiseFirst and StructureFirst for mean-histograms, which reports the mean value for each of its bins. One drawback of using the mean is that it is sensitive to outliers with extreme values, e.g., a single very high or very low unit count can significant distort the mean value. In contract, the *median* value is more robust in the presence of such outliers. Thus, reporting the median for each bin is more appropriate for some applications, e.g., salary statistics. Besides application requirements, there is also a technical motivation for using median instead of mean: that the sensitivity of StructureFirst discussed in Section 4 depends on the maximum possible unit-count F. As we show soon, reporting median value for each bin eliminates this dependence, leading to a more robust version of StructureFirst.

One difference between mean and median is that the former minimizes the expected sum of squared error with respect to the original data values, whereas the latter minimizes the *sum of absolute error* (SAE). Hence, minimizing overall SAE is a more appropriate objective for median-histograms. Specifically, given an ordered count sequence $D = \{x_1, x_2, \ldots, x_n\}$, the sum of absolute error (SAE) of a histogram $H = \{B_1, B_2, \ldots, B_k\}$ on D is calculated by Equation 12:

$$SAE(H,D) = \sum_{j}^{k} \sum_{l_j \le i \le r_j} |m_j - x_i| \tag{12}$$

where m_j, l_j and r_j denote the median, the left boundary and the right boundary in the j^{th} bin, respectively.

The computation of the optimal structure of a median-histogram is more complicated, in the sense that the median value cannot be incrementally computed. Algorithm 4 shows the procedure for building an optimal median-histogram, without considering the privacy issue. Let $H^*(D_i,j)$ be the optimal j-bin histogram representation on prefix sequence D_i with minimum SAE, and table TSAE$[i,j]$ records the best SAE value corresponding to the $H^*(D_i,j)$. Then, the optimum median-histogram construction problem is to find the optimal histogram $H^*(D,k))$ with the minimum error TSAE$[|D|;k]$. Here, D is the original data and k is the number of bins in the histogram.

Algorithm 4 Median-Histogram Construction (count sequence D with $|D| = n$, the number of bins k)

```
1: for each $i$ from 1 up to $n$ do
2:     TSAE$[i,1] := SAE(D,1,i)$
3:     for each $j$ from 2 to $k$ do
4:         TSAE$[i,j] := +\infty$
5:         for each $l := j-1$ down to 1 do
6:             TSAE$[i,j]$ := min (TSAE$[i,j]$,
                       TSAE$[l,j-1] + SAE(D,l+1,i)$)
7:             update $H^*(D_i,j)$ based on the recent TSAE$[i,j]$
8: Return the optimal histogram $H^*(D,k)$ w.r.t. TSAE$[|D|,k]$.
```

Compared with the mean-histogram construction algorithm, the major bottleneck of Algorithm 4 lies in the calculation of the SAEs. Since SAE is an order statistic, the calculation of SAE is naturally more time-consuming than that of SSE. Fortunately, the median value can still be computed in linear time, using the algorithm in [4]. Next we adapt NoiseFirst and StructureFirst to median histograms.

5.1 Median-NoiseFirst

Algorithm 5 describes the Median-NoiseFirst framework.

Median-NoiseFirst (Median-NF) follows similar ideas as Mean-NoiseFirst described in Algorithm 1. The major difference between Algorithms 5 and 1 is that Median-NF builds the optimal histogram over the noisy count sequence $\hat{D}$ based on the median statistic and the SAE metric, rather than the mean statistic and the SSE metric used in Algorithm 1. Next we analyze the expected SAE incurred by Median-NF. The key is to make a connection between (i) the SAE

Algorithm 5 Median-NoiseFirst (count sequence D, the number of bins k, privacy guarantee ϵ)

```
1: Generate a noisy database $\hat{D}$ by adding independent Laplace noise $Lap(\frac{1}{\epsilon})$ on each count $x_i \in D$.
2: Run Algorithm 4 to build the median-based optimal $k$-bin histogram on $\hat{D}$, denoted as $\hat{H}_k^*$.
3: Return histogram $\hat{H}_k^* = \{(l_1, r_1, m_1), \ldots, (l_k, r_k, m_k)\}$, in which every $m_j$ is the median of the noisy data subsequence $\{\hat{x}_{l_j}, \ldots, \hat{x}_{r_j}\}$.
```

of the histogram towards the noisy count sequence $\hat{D}$ and (ii) the SAE of the same histogram on the original count sequence D. To do this, we first derive Lemmata 7 and 8 to quantify the expected SAE of a single bin on noisy sequence $\hat{D}$ and original sequence D respectively. After that, we extend their conclusions to multiple bins in Theorem 4.

Lemma 7 *Given subsequence of counts $\{x_l, x_{l+1}, \ldots, x_r\}$. Let $\{\hat{x}_l, \hat{x}_{l+1}, \ldots, \hat{x}_r\}$ be the noisy counts after the first step in Algorithm 5. If (l, r, m) represents the result bin by using the median as the representative of all the noisy counts, the lower bound on the expected SAE of the bin with respect to $\{\hat{x}_l, \hat{x}_{l+1} \ldots, \hat{x}_r\}$ is*

$$\mathbb{E}\left(Error((l,r,m), \{\hat{x}_l, \ldots, \hat{x}_r\}\right) \leq SAE(D,l,r) + \frac{2(r-l)}{\epsilon}$$

Proof Let X_i represent the variable of the count x_i after adding the noise following $Lap(\frac{1}{\epsilon})$. Assume that the median of the noisy sequence $\{X_l, \ldots, X_r\}$ equals to X_e. Meanwhile, x_d denotes the median of the original count sequence $\{x_l, \ldots, x_r\}$. In practice, x_e may not equal to x_d. Let δ_i denote the residual variable on X_i, i.e., $\delta_i = X_i - x_i$, and let the size of the bin be $s = r - l + 1$. Then the upper bound on the expected SAE of the result bin (l, r, m) with respect to the noisy count sequence $\{\hat{x}_l, \ldots, \hat{x}_r\}$ is as follows.

$$\begin{aligned}
&\mathbb{E}\left\{\sum_{i=l, i\neq m}^{r} |X_i - X_e|\right\} \\
&\leq \mathbb{E}\left\{\sum_{i\neq e} (|x_i - x_e| + |x_i - X_i| + |x_e - X_e|)\right\} \\
&= \sum_{i\neq e} |x_i - x_e| + \mathbb{E}\left\{\sum_{i\neq e} |x_i - X_i| + \sum_{i\neq e} |x_e - X_e|\right\} \\
&= SAE(D,l,r) + \mathbb{E}\left\{\sum_{i\neq e} |\delta_i|\right\} + \mathbb{E}\left\{\sum_{i\neq e} |\delta_e|\right\} \\
&= SAE(D,l,r) + \frac{2(s-1)}{\epsilon}
\end{aligned}$$

The above follows the triangle inequality and the property of the median. The last equality is because each 0-mean Laplace variable, e.g., δ_i, satisfies the exponential distribution with an expected value of $\frac{1}{\epsilon}$. Since $s = r - l + 1$, this leads to the conclusion of this lemma. □

Lemma 7 shows the upper bound on the expected SAE with respect to the noisy count sequence $\hat{D}$ if we merge some consecutive noisy counts into a single bin. Next, in Lemma 8, we discuss the expected SAE of merging consecutive noisy counts with respect to the original count sequence D.

Lemma 8 *Given subsequence of counts $\{x_l, x_{l+1}, \ldots, x_r\}$. Let $\{\hat{x}_l, \hat{x}_{l+1}, \ldots, \hat{x}_r\}$ be the noisy counts after the first step in Algorithm 5. If (l, r, m) is the result bin by using the median as the representative of all the noisy counts, the lower bound on the expected SAE of the bin with respect to $\{x_l, x_{l+1}, \ldots, x_r\}$ is*

$$\mathbb{E}\left(Error(\{(l,r,m\}, \{x_l, \ldots, x_r\})\right) \geq SAE(D,l,r) - \frac{r-l}{\epsilon}$$

Proof Similar to the proof of Lemma 7, we again use the notations X_i and X_e to represent the random variable of count x_i after the injection of Laplacian noise and the median in the noisy count sequence $\{X_l, X_{l+1}, \cdots, X_r\}$, respectively.

$$\begin{aligned}
&\mathbb{E}\left\{\sum_{i=l, i\neq m}^{r} |x_i - X_m|\right\} \\
&\geq \mathbb{E}\left\{\sum_{i\neq m} (|x_i - x_m| - |x_m - X_m|)\right\} \\
&= \sum_{i\neq m} |x_i - x_m| - \mathbb{E}\left\{\sum_{i\neq m} |x_m - X_m|\right\} \\
&= SAE(D,l,r) - \mathbb{E}\left\{\sum_{i\neq m} |\delta_m|\right\} \\
&= SAE(D,l,r) - \frac{s-1}{\epsilon}
\end{aligned}$$

This completes the proof. □

Extending the conclusions in Lemma 7 and Lemma 8 to k bins, we obtain Theorem 4. In the following analysis, $\hat{H}_k^*$ denotes the optimal k-bin histogram built on the noisy count sequence $\hat{D}$, and H_k represents the histogram with the same k-bin structure as $\hat{H}_k^*$ but with different median values in the bins. Specifically, instead of using noisy counts $\{\hat{x}_{l_j}, \ldots, \hat{x}_{r_j}\}$, H_k calculates the median for each bin B_j on the original count sequence $\{x_{l_j}, \ldots, x_{r_j}\}$. It can be understood that $Error(H_k, D) = \sum_j^k SAE(D, l_j, r_j)$, which leads to the Theorem below.

Theorem 4 *Given H_k and $\hat{H}_k^*$, the expected SAE of the histogram on $\hat{D}$ and D are*

$$\mathbb{E}(Error(\hat{H}_k^*, \hat{D})) \leq Error(H_k, D) + \frac{2(n-k)}{\epsilon}$$
$$\mathbb{E}(Error(\hat{H}_k^*, D)) \geq Error(H_k, D) - \frac{n-k}{\epsilon}$$

Note that $Error(H_k, D)$ depends on the sensitive data, and, thus, cannot be published or used in the algorithm. Theorem 4 builds a connection between $\mathbb{E}(Error(\hat{H}_k^*, D))$ and $\mathbb{E}(Error(\hat{H}_k^*, \hat{D}))$. Specifically, based on Theorem 4, we estimate the expected SAE lower bound of $\hat{H}_k^*$ on D based on the expected SAE upper bound of $\hat{H}_k^*$ on $\hat{D}$, as follows.

$$\mathbb{E}(Error(\hat{H}_k^*, D)) \geq \mathbb{E}\left(Error(\hat{H}_k^*, \hat{D}) - \frac{3(n-k)}{\epsilon}\right) \tag{13}$$

To minimize the expected value of $Error(\hat{H}_k^*, D)$, we minimize its upper bound shown in the Equation 13. While the analysis above assumes that parameter k is given by the users, we can also automatically derive the optimal k following the idea of Algorithm 2 in Section 3. Equation 14 summarizes this optimization problem.

$$\hat{H}^* \approx \arg\min_{\hat{H}_k} \mathbb{E}\left(Error(\hat{H}_k^*, \hat{D}) - \frac{3(n-k)}{\epsilon}\right) \tag{14}$$

5.2 Median-StructureFirst

Algorithm 6 shows the StructureFirst algorithm for building a median-histogram under ϵ-differential privacy. The notation $E_{SAE}(q, j, r_{j+1})$ represents the SAE of the histogram, which corresponds to the optimal histogram $H^*(D_q, j)$ covering $[1, q]$ using at most j bins and the $(j+1)^{th}$ bin covering the following subsequence of $[q+1, r_{j+1}]$.

Although Median-StructureFirst (median-SF) resembles Mean-StructureFirst (Mean-SF) described in Algorithm 3, the former involves some modifications that further improve the accuracy of the output histogram. First, SSE calculation is replaced by the computation of SAE in Lines 2 and 6. Second, in Line 7, the probability of setting the j^{th} boundary of the histogram to q is changed from $\exp\left\{-\frac{\epsilon_1 E_{SSE}(q,j,r_{j+1})}{2(k-1)(2F+1)}\right\}$ to $\exp\left\{-\frac{\epsilon_1 E_{SAE}(q,j,r_{j+1})}{2(k-1)}\right\}$. F is the upper bound of the maximum possible count in the original data. Finally, the output histogram (Line 10) consists of medians instead of means. Each median value is added a Laplace noise with a scale of $\frac{1}{\epsilon_2}$ rather than the $\frac{1}{\epsilon_2(r_j - l_j + 1)}$ in Mean-SF. Although in Median-SF, the scale of Laplace noise added to each bin increases, Median-SF still has advantages compared to Mean-SF. This is because the boundary adjustment

Algorithm 6 Median-StructureFirst (count sequence D, the number of bins k, privacy parameter ϵ)

```
1: Partition $\epsilon$ into two parts $\epsilon_1$ and $\epsilon_2$ that $\epsilon_1 + \epsilon_2 = \epsilon$.
2: Build the optimal median-based $k$-bin histogram $H_k^*$ and keep all
   the intermediate results, i.e. $\text{TSAE}(i, j)$ for all $1 \leq i \leq n$ and
   $1 \leq j \leq k$
3: Set the right boundary of the $k^{th}$ bin at $r_k = n$
4: for each $j$ from $k-1$ down to $1$ do
5:     for each $q$ from $j$ up to $r_{j+1} - 1$ do
6:         Calculate $E_{SAE}(q, j, r_{j+1}) = \text{TSAE}(q, j) + SAE(D, q+1, r_{j+1})$
7:     select $r_j = q$ from $k-1 \leq q < r_{j+1}$ with a probability
       proportional to $\exp\left\{-\frac{\epsilon_1 E_{SAE}(q,j,r_{j+1})}{2(k-1)}\right\}$
8:     Set $l_{j+1} = r_j + 1$
9: Calculate the median $m_j$ for every bin on interval $[l_j, r_j]$.
10: Add Laplace noise to median counts as $\hat{m}_j = m_j + Lap\left(\frac{1}{\epsilon_2}\right)$
11: Return histogram $\{(l_1, r_1, \hat{m}_1), \ldots, (l_k, r_k, \hat{m}_k)\}$
```

procedure in Median-SF does not depend on the factor $(2F+1)$, which decreases the randomness in the adjusted boundaries, as shown in the experiments.

In the following, we analyze the correctness and accuracy of Algorithm 6 in Section 5.2.1 and Section 5.2.2. Section 5.2.3 presents the strategy for finding a good assignment for privacy budgets ϵ_1 and ϵ_2 is proposed based on the cost analysis results in the Section 5.2.2.

5.2.1 Correctness

The worst case sensitivity analysis on the intermediate dynamic programming with respect to Median-StructureFirst is given in Lemma 9, where $H^*(D_i, j)$ represents the optimal SAE-based histogram constructed on the subsequence D_i using at most j bins.

Lemma 9 *Given two neighbor database D and D' which differs from each other by at most 1 on a certain count, the SAE of the optimal histograms on D and D' with respect to the first i counts changes by at most*

$$|E_{SAE}(H^*(D'_i, j), D'_i) - E_{SAE}(H^*(D_i, j), D_i)| \leq 1$$

Proof Given two neighbor databases as D and D', there is one and only one count x_e changes by 1, i.e., $x_e - 1 \leq x'_e \leq x_e + 1$. Assume that x_e is in the bin (l_z, r_z, m_z) of the histogram $H^*(D_i, j)$, where m_z denotes the median of the bin (l_z, r_z). Since $H^*(D'_i, j)$ is the optimal j-bin histogram for subsequence D'_i, it is straightforward to know that given any histogram H' with j bins covering the interval $[1, i]$, we have

$$E_{SAE}(H^*(D'_i, j), D'_i) \leq E_{SAE}(H', D'_i) \tag{15}$$

Assume that H' is a special histogram on D'_i which employs all bins in $H^*(D_i, j)$, except that m'_z may not equal to m_z due to the difference between x'_e and x_e. According to the property of the median on the integer sequence, $|m'_z - m_z| \leq 1$. In the following, we derive an upper bound for the SAE of H' on D'_i.

$$\begin{aligned} & E_{SAE}(H', D'_i) - E_{SAE}(H^*(D_i, j), D_i) \\ & = m'_z - m_z \leq 1 \end{aligned} \tag{16}$$

Based on Equations 15 and 16, we have

$$\begin{aligned} & E_{SAE}(H^*(D'_i, j), D'_i) \\ & \leq E_{SAE}(H', D'_i) \\ & \leq E_{SAE}(H^*(D_i, j), D_i) + 1 \end{aligned} \tag{17}$$

This reaches the conclusion of the lemma. □

Given the worst case analysis on the change of SAE between the two optimal histograms of two neighbor databases, we further prove that the boundary readjustment strategy in Algorithm 6 satisfies differential privacy.

Lemma 10 *The selection of r_j on line 7 of Algorithm 6 satisfies $\frac{\epsilon_1}{k-1}$-differential privacy.*

Proof Let $E_{SAE}(q, j, r_{j+1})$ be the SAE cost of setting the r^{th} boundary r_j to q, and let the notation $E'_{SAE}(q, j, r_{j+1})$ represent the SAE cost when the same histogram structure is used on D'. On the basis of the conclusion in Lemma 9, we have

$$|E'_{SAE}(q, j, r_{j+1}) - E_{SAE}(q, j, r_{j+1})| \leq 1$$

$$\begin{aligned} \Pr(r_j = q) &= \frac{\exp\left\{-\frac{\epsilon_1 E'_{SAE}(q,j,r_{j+1})}{2(k-1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1 E'_{SAE}(z,j,r_{j+1})}{2(k-1)}\right\}} \\ &\leq \frac{\exp\left\{-\frac{\epsilon_1(E_{SAE}(q,j,r_{j+1})-2F-1)-1}{2(k-1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1(E_{SAE}(z,j,r_{j+1})+2F+1)+1}{2(k-1)}\right\}} \\ &= \exp\left(\frac{\epsilon_1}{k-1}\right) \frac{\exp\left\{-\frac{\epsilon_1 E_{SAE}(q,j,r_{j+1})}{2(k-1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1 E_{SAE}(z,j,r_{j+1})}{2(k-1)}\right\}} \end{aligned}$$

Hence, the selection of r_j follows $(\frac{\epsilon_1}{k-1})$-differential privacy. □

The derivation of Lemma 10 follows the Exponential Mechanism.

Lemma 10 tells us that each execution of the line 7 in Algorithm 6 consumes a privacy budget of $\frac{\epsilon_1}{k-1}$. Based on similar analysis in Theorem 2, we prove the correctness of the following theorem.

Theorem 5 *Algorithm 6 satisfies ϵ-differential privacy.*

5.2.2 Accuracy Analysis

Based on the analysis of SAE in the expectation case, we first derive Lemma 11 below, which discusses the SAE incurred by every time we randomly adjust a boundary in the optimal histogram structure.

Lemma 11 *The SAE of the histogram increments by no more than $\frac{2(k-1)^2}{\epsilon_1(2(k-1)-\epsilon_1 nF)}$, every time we move the boundary by line 7 in Algorithm 6.*

Proof We also assume the optimal $(j+1)$-bin histogram utilizes the optimal location q^* as the boundary between the j^{th} bin and the $(j+1)^{th}$ bin to minimize the error of $E_{SAE}(H^*(D_q, j), D_q) + SAE(q+1, r_{j+1})$. In the following, we discuss the expected growth on the SAE when the randomized algorithm fails to select the optimal location q^*. We employ the notation $E_{SAE}(z, j, r_{j+1})$ to represent the SAE of the histogram, that consists of the optimal histogram with j bins covering $[1, z]$ and the $(j+1)$th bin covering $[q+1, r_{j+1}]$. Based on the fact that has been proposed in Lemma 5 of Section 4, the following inequalities hold.

$$\begin{aligned} & \sum_z \exp\left\{-\frac{\epsilon_1(E_{SAE}(z,j,r_{j+1}) - E_{SAE}(q^*,j,r_{j+1}))}{2(k-1)}\right\} \\ & \geq \sum_z \left(1 - \frac{\epsilon_1(E_{SAE}(z,j,r_{j+1}) - E_{SAE}(q^*,j,r_{j+1}))}{2(k-1)}\right) \\ & \geq n\left(1 - \frac{\epsilon_1 nF}{2(k-1)}\right) \end{aligned} \tag{18}$$

As we have mentioned in Section 4, F is some priori knowledge on the upper bound of the maximum count on the sequence D. Given F, the second inequality of Equation 18 holds since $E_{SAE}(z, j, r_{j+1}) - E_{SAE}(q^*, j, r_{j+1}) < nF$.

To simplify our symbolic system, we use $E(q)$ to replace $E_{SAE}(q, j, r_{j+1})$ if the variables j and r_{j+1} are clear in the context. The expection on the additional SAE towards the optimal j^{th} boundary, i.e., q^*, is,

$$\begin{aligned} & \sum_q \{(E(q))\Pr(r_j = q)\} - E(q^*, j, r_{j+1}) \\ & = \sum_q \{(E(q) - E(q^*))\Pr(r_j = q)\} \\ & = \sum_q \left\{(E(q) - E(q^*)) \frac{\exp\left\{-\frac{\epsilon_1(E(q)-E(q^*)}{2(k-1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1(E(z)-E(q^*)}{2(k-1)}\right\}}\right\} \\ & \leq \sum_q \left\{(E(q) - E(q^*)) \frac{\exp\left\{-\frac{\epsilon_1(E(q)-E(q^*)}{2(k-1)}\right\}}{n - \frac{\epsilon_1 n^2 F}{2(k-1)}}\right\} \end{aligned}$$

$$\leq \sum_{q} \frac{k-1}{\epsilon_1} \left(n - \frac{\epsilon_1 n^2 F}{2(k-1)} \right)^{-1}$$
$$\leq \frac{2(k-1)^2}{\epsilon_1 \left(2(k-1) - \epsilon_1 nF\right)} \tag{19}$$

The first inequality is the result of applying the Equation 18. The second inequality is due to the Lemma 5 and the last inequality is because the number of candidate locations for q is smaller than the database cardinality. This completes the proof of the Lemma 11. □

As being indicated in Lemma 11 that the error upper bound of each boundary adjustment never depends on j or r_{j+1}. Consequently, the total incremental SAE after $(k-1)$ times of boundary adjustment (see Line 7 in Algorithm 6) is $(k-1)$ times of the error in Lemma 11. Taking into account the error produced by Laplacian noise injection (see Line 10 in Algorithm 6) together, we can answer the question proposed at the beginning of this section by putting forward Theorem 6.

Theorem 6 *For the output histogram of Algorithm 6, the expected SAE is at most*

$$E_{SAE}(H^*(D,k), D) + \frac{2(k-1)^3}{\epsilon_1 \left(2(k-1) - \epsilon_1 nF\right))} + \frac{2k}{(\epsilon_2)^2}$$

5.2.3 Budget Assignment

When parameter k is fixed, we find a good budget assignment scheme between ϵ_1 and ϵ_2 by solving the following optimization problem which minimizes the expected SAE derived in Section 5.2.2. We use the same numerical methods as in Section 4.3 to obtain a near-optimal assignment between ϵ_1 and ϵ_2.

$$\text{Minimize} \quad \left(\frac{2(k-1)^3}{\epsilon_1 \left(2(k-1) - \epsilon_1 nF\right))} + \frac{2k}{(\epsilon_2)^2} \right)$$
$$\text{s.t.} \quad \epsilon_1 + \epsilon_2 = \epsilon$$

6 Answering Arbitrary Range Counts

So far, we construct a histogram to minimize either the *Sum of Squared Error* (SSE) or the *Sum of Absolute Error* (SAE), which correspond to the error of unit-length queries. When the user queries on the sum of consecutive counts, the histogram synopsis may not be optimal in terms of query accuracy. To understand the underlying reason behind the ineffectiveness of histogram synopsis on large range queries, let us revisit the example in Figure 2(a). If the user queries for patients with age between 25 and 35, the average count of the first bin is not sufficient to return accurate result, even when there is no Laplace noise added on the histogram. Therefore, in some cases, more accurate results can be obtained if a bin adopts a non-uniform scheme of count computation.

In this section, we discuss how to implement such non-uniform scheme in both NoiseFirst and StructureFirst respectively. Two different strategies are employed in NoiseFirst and StructureFirst because of the different properties of the algorithms.

6.1 Range Counting Using NoiseFirst

In Algorithm 1, after finding the structure of the histogram, Mean-NoiseFirst uses the average count to replace the original noisy counts. Another option is to keep using the noisy data instead of the average counts in some of the bins. To ensure that a better selection, we propose to compare the estimations on the errors of the two options and adopt the one with smaller expected SSE error. In particular, for every bin B_i, we assume that $[l, r]$ is the interval B_i covers in the data domain, and c is the mean count over the noisy counts. Mean-NoiseFirst uses the average count for B_i only when

$$SSE(\hat{D}, l, r) < \frac{4(r-l)}{\epsilon^2} \tag{20}$$

Otherwise, the noisy counts $\{\hat{x}_l, \ldots, \hat{x}_r\}$ are kept as the original values. The Equation 20 is derived by combining the conclusions of Equations 2 and 4 (with $k = 1$ and $n = r-l+1$) in Section 3. Intuitively, when the structural error of the histogram is small, our scheme prefers to use the average counts in the bin. It is because the average count is capable of reducing the noise in consecutive noisy counts. When the original counts are very different from each other in the bin, averaging over all counts does not help reduce the errors. In such situations, the noisy counts may perform better.

Regarding Median-NoiseFirst, the median statistic takes the place of the mean statistic to represent a merged bin, the condition that we output the median count for bin B_i is

$$SAE(\hat{D}, l, r) < \frac{4(r-l)+1}{\epsilon} \tag{21}$$

Otherwise, the original noisy counts $\{\hat{x}_l, \ldots, \hat{x}_r\}$ are employed to represent the bin B_i. The Equation 21 is obtained on the basis of Equation 14 in Section 5 and Equation 22 below. Equation 22 denotes the expected SAE of the noisy counts $\{\hat{x}_l, \cdots, \hat{x}_r\}$ with respect to the original data sequence $\{x_l, \cdots, x_r\}$.

$$\mathbb{E}\left\{\{(l, l, \hat{x}_l), \cdots, (r, r, \hat{x}_r)\}, \{x_l, \cdots, x_r\}\right\}$$
$$= \frac{(r-l+1)}{\epsilon} \tag{22}$$

6.2 Range Counting Using StructureFirst

Unfortunately, the non-uniform strategy used for NoiseFirst method is not applicable in StructureFirst, since StructureFirst calculates the histogram on the original counts. To apply the non-uniform scheme, we mainly borrow the ideas from [13]. Generally speaking, after constructing all the bins, StructureFirst runs the boosting algorithm [13] on every bin, with differential privacy parameter ϵ_2. The following scheme is applicable to both of Mean-StructureFirst and Median-StructureFirst.

Given a bin B_i covering $[l, r]$, Laplace noise with magnitude $Lap\left(\frac{2}{\epsilon_2}\right)$ is added on every count x_j for $l \le j \le r$, as well as the sum $s_i = \sum_{l \le j \le r} x_j$. Assume $\overline{x}_j$ is the result noisy count and $\overline{s}_i$ is the noisy sum. Our algorithm runs normalization on $\{\overline{x}_l, \ldots, \overline{x}_r, \overline{s}_i\}$ and returns a new group of counts $\{x'_l, \ldots, x'_r, s'_i\}$ satisfying that $s'_i = \sum_{l \le j \le r} x'_j$. These counts are used to approximate the original counts in the database. By [13], this scheme is always consistent with ϵ_2-differential privacy. Therefore, the complete modified algorithm of StructureFirst is still fulfilling ϵ-different privacy.

There is some connection between such method and [13]. StructureFirst with this non-uniform count assignment scheme can be considered as an adaptive version of the boosting algorithm in [13]. In the original boosting algorithm, the user must specify the fan-out of the tree structure before running the algorithm. This fan-out decides how the algorithm partitions the count sequence and how the normalization is run bottom-up and top-down. In our histogram technique, an adaptive partitioning is used instead, which is supposed to better capture the distribution of the data. This is the underlying reason that our StructureFirst method outperforms the algorithms [13]. In the following section, some empirical studies verify the advantages of our proposal.

7 Experiments

This section experimentally compares the effectiveness of the four proposed algorithms, namely Mean-NoiseFirst, Median-NoiseFirst, Mean-StructureFirst and Median-StructureFirst for range count queries with three state-of-the-art methods, referred to as *Dwork*[7], *Privlet*[24] and *Boost*[13]. Specifically, in our implementation of StructureFirst, the parameter F, i.e., maximal possible count in a histogram bin, is fixed to $\frac{10^4 * NumOfRecords}{DomainSize}$. Meanwhile, the implementation of Boost follows the same settings in [13], and employs binary trees as the synopsis structure as in [13]. Similarly, both StructureFirst solutions also utilize a binary tree inside each bin of the histogram, in order to ensure the fairness in the comparisons. Akin to [13], we evaluate the accuracy of all algorithms over range queries with varying lengths and locations. In particular, given a query length L, we test all its possible range queries and report the *average squared error average absolute error*, and *average relative error* for each method. We run experiments with three popular values of ϵ: 0.01, 0.1 and 1. All methods are implemented in C++ and tested on an Intel Core 2 Duo 2.33 GHz CPU with 2GB RAM running Windows XP. In each experiment, every method is executed for 10 times, and its average accuracy is reported. The experiments involve four real-world datasets:

- *Age*[24] contains 100,078,675 records, each of which corresponds to the age of an individual, extracted from the IPUM's census data of Brazil. The ages range from 0 to 101. Differential privacy guarantees the hardness for adversaries to infer any individual's true age from published statistics.
- *Search Logs*[13] is a synthetic dataset generated by interpolating *Google Trends* data and *America Online search logs*. It contains $32,768$ records, each of which stores the frequency of searches (ranging from 1 to 496) with the keyword "Obama" within a 90 minutes interval, from Jan. 1, 2004 to Aug. 9, 2009. DP ensures that the adversary cannot derive if any specific person has searched the keyword "Obama" at a particular time.
- *NetTrace*[13] contains the IP-level network trace at a border gateway in a major university. Each record reports the number of external hosts connected to an internal host. There are $65,536$ records with connection number ranging from 1 to 1423. The application of DP protects the information of individual connections.
- *SocialNetwork*[13] records the friendship relations among 11K students from the same institution, derived from an online social network website. Each record contains the number of friends of certain student. There are $32,768$ students, each of which has at most 1678 friends. DP protects the sensitive information of individuals' connections from adversaries.

To generate histograms, we transform each of the four datasets to statistical counts on every possible value. On *Age*, for example, each x_i indicates the number of individuals aged i ($1 \le i \le 101$). We observed that the distribution of *Age* is very different from the other three, in that the counts therein are more evenly distributed over the entire domain, while all the other datasets exhibit a high degree of skewness.

In the following, we use *NoiseFirst* to represent both Mean-NoiseFirst and Median-NoiseFirst, and *StructureFirst* for both Mean-StructureFirst and Median-StructureFirst. Meanwhile, *mean-based solutions* refer to Mean-NoiseFirst and Mean-StructureFirst, and *median-based solutions* refer to Median-NoiseFirst and Median-StructureFirst. Since in StructureFirst, the parameter k (i.e., the number of bins in the resulting histogram) can not be derived by the theoretical analysis, in Section 7.1, we first test the effect of different ks on the accuracy of the algorithm. These results provide guidelines on the selection of k in StructureFirst. After that, Section 7.2 compares the mean-based solutions with their corresponding median-based solutions. For example, Mean-NoiseFirst is compared with Median-NoiseFirst and then the better one of the two is selected to participate into the comparisons in Section 7.3. Finally, Section 7.3 compares the performance of NoiseFirst and StructureFirst against state-of-the-art solutions mentioned above.

7.1 Impact of Parameter k on StructureFirst

This subsection evaluates the effectiveness of StructureFirst with different the number of bins k in the resulting histogram. Specifically, for each dataset with domain size n, we test five different values of k: 1, $\frac{n}{15}$, $\frac{n}{10}$, $\frac{n}{5}$, n. Since Mean-StructureFirst and Median-StructureFirst are designed to minimize SSE and SAE, respectively, we report query accuracy in terms of squared error for Mean-StructureFirst, and absolute error for Median-StructureFirst.

Figures 5-8 illustrate the average squared error of Mean-StructureFirst with different values of k and different query range sizes on all the four datasets. Figures 9-12 repeat the same experiments for Median-StructureFirst, reporting absolute rather than squared error. From the experimental results, we obtain the following important observations. First, when k equals to the domain size n, StructureFirst reduces to the Laplace Mechanism[7] that adds random Laplace noise with magnitude $\frac{1}{\epsilon}$ to every count in the dataset. In this case, the error incurred by StructureFirst increases linearly with the query range size, for all values of ϵ. Note that the histogram structure is fixed to the finest one with one bin per count; hence, there is no need to spend privacy budget to protect the histogram structure.

Second, when $k = 1$, StructureFirst is equivalent to Boost [13]. Similar to the case for $k = n$, the histogram structure is fixed, and there is no need to spend privacy budget on it. StructureFirst's performance follows the properties of Boost, with more accurate results on queries with very small or very large ranges.

Third, when $1 < k < n$, StructureFirst achieves generally better performance compared to the cases with extreme values. This shows that StructureFirst successfully constructs histograms adaptively to the data distribution, reducing the error for queries covering different bins in the data domain.

Finally, on all datasets, both Mean-StructureFirst and Median-StructureFirst simultaneously achieve their highest accuracy when k is around $n/10$. In such cases, each bin consists of 10 counts by average, offering a balanced results for all types of queries with different lengths. Moreover, the normalization technique from [13] also works well, since every bin only involves 10 random counts in the processing by average. Therefore, we employ $k = n/10$ for StructureFirst in the rest of the paper.

7.2 Comparison of mean-based and median-based solutions

In this section, we compare the the proposed mean-based and median-based methods. In particular, Mean-NoiseFirst is tested against Median-NoiseFirst (Figure 13a-d), and Mean-StructureFirst is compared with Median-StructureFirst (Figure 14a-d). As the SSE can also reflect the value of SAE, we uniformly exhibit the tendency changes of all the above mentioned methods in consideration of the SSE error metric. We observe the following from the results shown in Figure 13. First, in both Mean-NoiseFirst (Mean-NF) and Median-NoiseFirst (Median-NF), the error increases linearly with the size of the query range. This is because these two methods spend the whole privacy budget on the noise injected to each bin. Therefore, they incur a noise variance of $O(\frac{m}{\epsilon^2})$, where m is a variable corresponding to the query range size.

Second, Median-NF performs apparently better than Mean-NF on all datasets when the degree of privacy protection is high (i.e., for $\epsilon = 0.1$ and $\epsilon = 0.01$). The error reduction in Median-NF is due to the fact that with a higher amount of noise injected to each count, there is a higher probability for the algorithm to produce extreme counts in each merged bin. Considering that the mean statistic is susceptible to those extreme counts, and consequently the estimation accuracy of Mean-NoiseFirst is decreased, while Median-NoiseFirst performs well since those outlier counts have little or no effect on the median statistic.

Third, Mean-NF outperforms Median-NF for all datasets when $\epsilon = 1$. To explain this, we restate that a larger number of privacy budget ϵ leads to a smaller scale of noise added to each count. When the noise scale is small, the mean statistic has its inherent advantage in summarizing all counts dropping into the same merged bin. Up to now, we can conclude that Median-NF has advantage in handling data with higher amount of noise and Mean-NF performs well when the noise added to each bin is not so big.

From Figure 14, we make the following observations.

First, Median-StructureFirst (Median-SF) has higher accuracy than Mean-StructureFirst (Mean-SF) when the noise scale is relatively small (i.e., for $\epsilon = 1$ and $\epsilon = 0.1$ here). This is because the Median-SF does not rely on the parameter F, which represents the prior knowledge on the upper bound of the maximal count in the original data. Therefore, during the boundary random adjustment phase, Median-SF adjust boundaries in the optimal histogram structure according to a probability of $\exp\left\{-\frac{\epsilon_1 E_{SAE}(q,j,r_{j+1})}{2(k-1)}\right\}$, which does not depend on F. On the other hand, Mean-SF must consider F into its boundary adjustment. The absence of parameter F in such a probability equation reduce the randomness of the

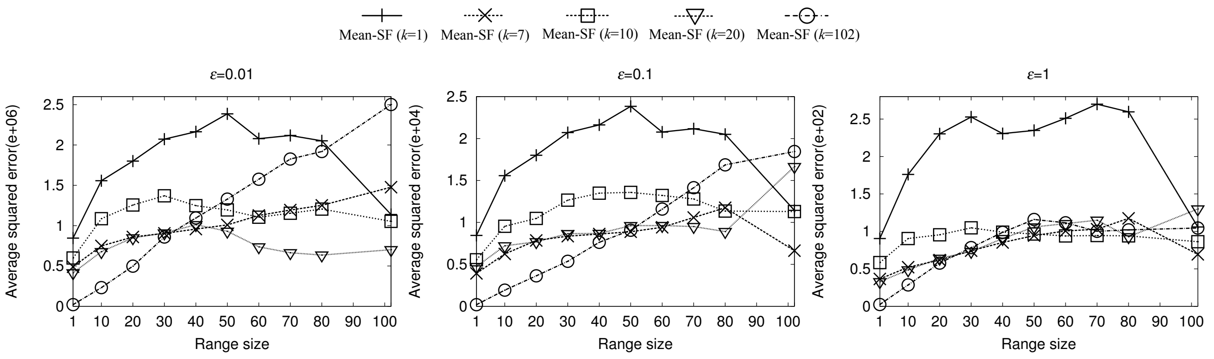


Fig. 5 Varying k on *Age*

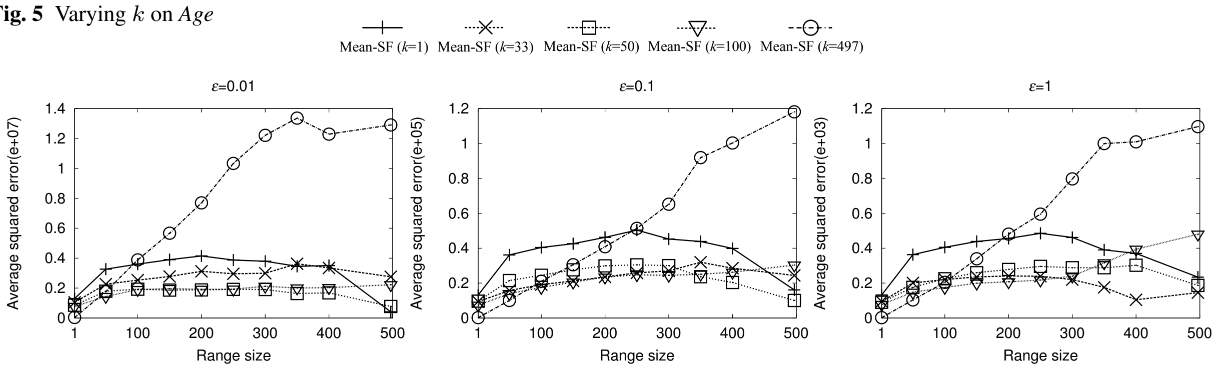


Fig. 6 Varying k on *Search Logs*

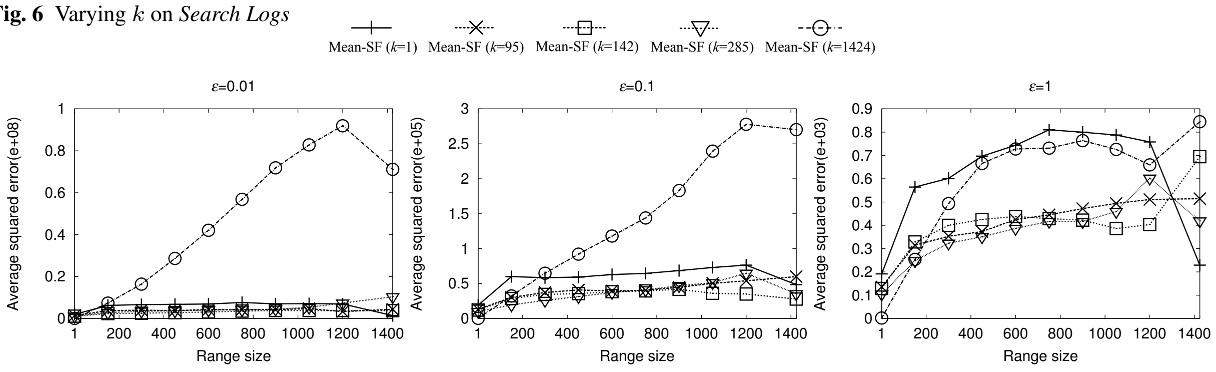


Fig. 7 Varying k on *NetTrace*

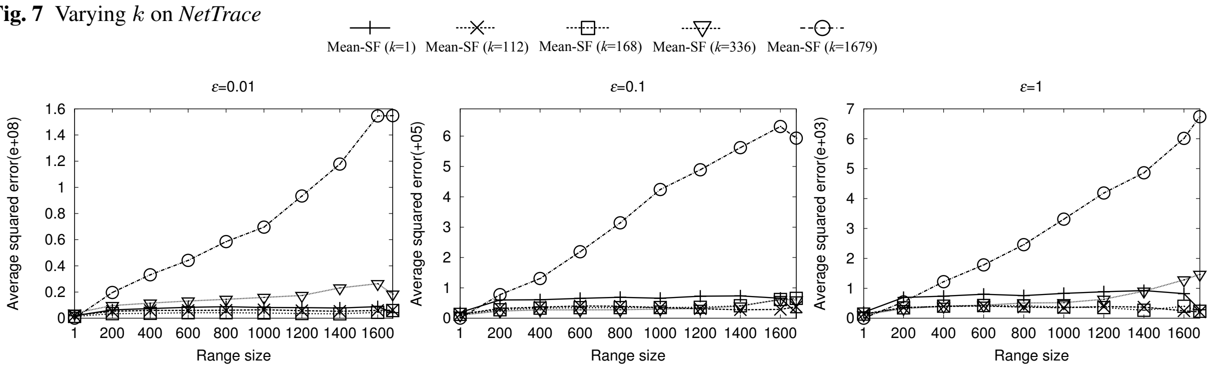


Fig. 8 Varying k on *Social Network*

adjustment and hence produce a more accurate histogram structure. This is due to the accurate histogram structure, which improves the accuracy of Median-SF.

Second, when the privacy budget $\epsilon = 0.01$, Mean-SF outperforms Median-SF on most datasets. Based on the above analysis, Median-SF has advantages in producing more accurate histogram structure after the boundary adjustment phase. Such an advantage can be ascribed to the fact that it is based on the median statistic. However, when the privacy budget is small, the randomness of noisy counts increased, and therefore there is little gain in producing a more accurate histogram structure. On the other hand, as we have stated in the Section 1, the utilizing of the mean statistic decreases the sensitivity in each merged bin and hence cuts down the

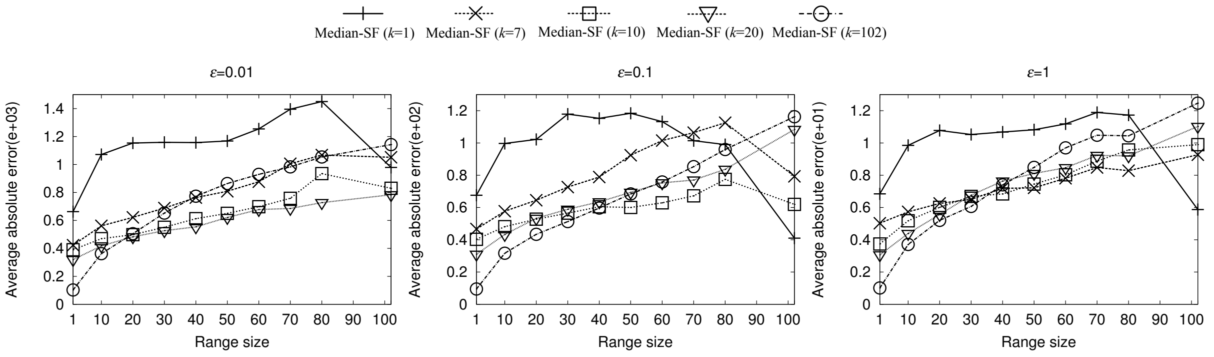


Fig. 9 Varying k on *Age*

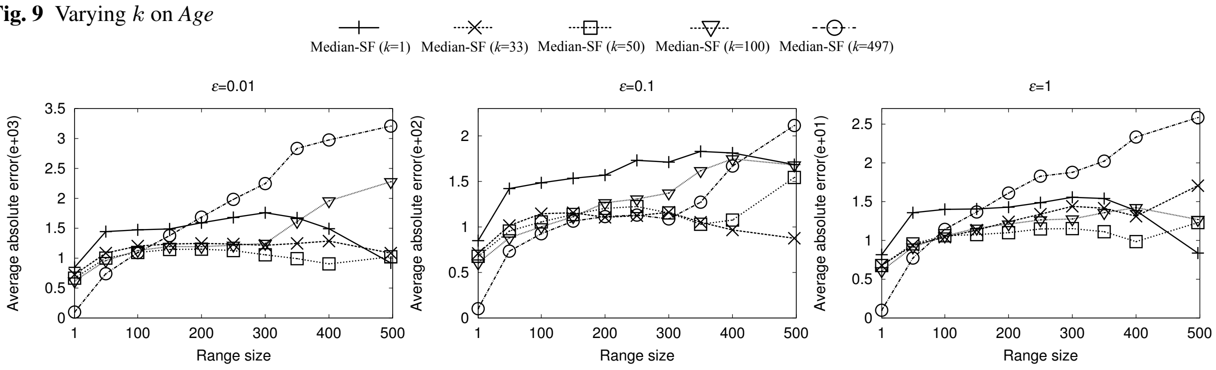


Fig. 10 Varying k on *Search Logs*

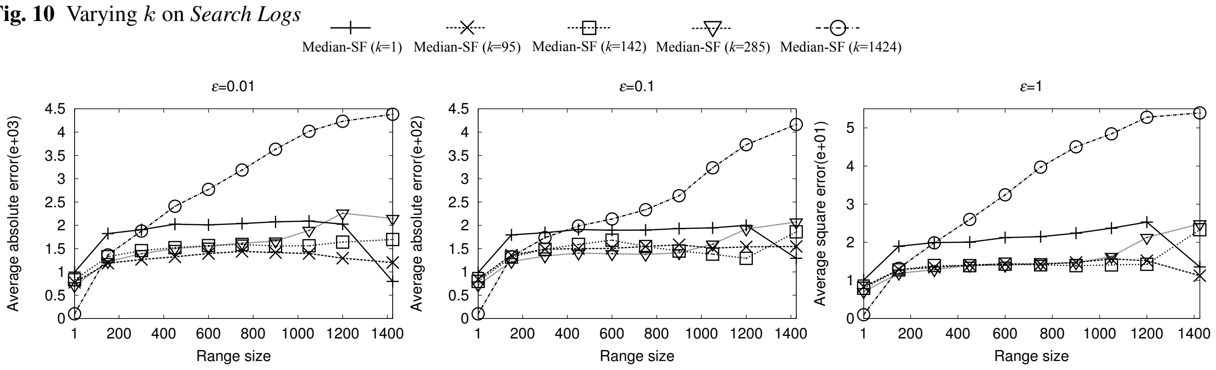


Fig. 11 Varying k on *NetTrace*

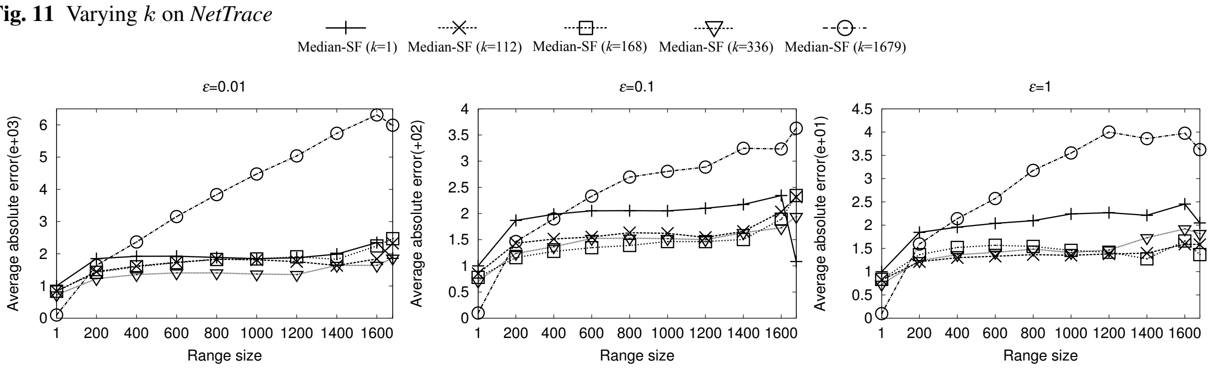


Fig. 12 Varying k on *Social Network*

scale of noise added to each mean count. When the privacy budget is small, the noise scale becomes rather large. Under such conditions, a method that reduces noise scale has significant advantage in producing more an accurate histogram. Although Mean-SF outperforms Median-SF on most datasets, a special case happens on the *Age* dataset. This is because *Age* is different from the other three datasets in that it contains a series of large counts. The noise injected to each count is relatively low although the absolute noise scale is as large as $\frac{1}{0.01}$. For this reason, Median-SF still outperforms Mean-SF due to its good histogram structure.

Dataset	*Age*			*Search Logs*			*NetTrace*			*Social Network*		
ϵ	0.01	0.1	1	0.01	0.1	1	0.01	0.1	1	0.01	0.1	1
Dwork	**18,865**	**187**	2.38	19,986	204	1.98	19,927	204	1.99	20,153	198	2.05
Boost	730,466	7,666	76	1,264,010	12,037	127	1,774,689	16,956	178	1,701,026	17,769	172
Privlet	360,774	4,745	42	678,763	6,975	62	978,884	9,987	91	972,103	9,426	92
Median-NF	20,825	197	**1.88**	**4,944**	**65**	**1.47**	**3,394**	**34**	1.75	**4,028**	**49**	**0.79**
Mean-NF	21,178	205	1.98	12,231	131	1.79	13,198	128	**1.34**	12,677	135	1.40
Median-SF	285,923	2,929	26	876,980	9,091	88	1,351,249	12,412	121	1,383,363	11,694	138
Mean-SF	598,750	5,541	58	854,311	9,668	90	1,156,542	12,154	129	1,362,108	12,687	123

Table 2 Comparison of average squared errors on unit-length queries

Dataset	*Age*			*Search Logs*			*NetTrace*			*Social Network*		
ϵ	0.01	0.1	1	0.01	0.1	1	0.01	0.1	1	0.01	0.1	1
Dwork	**99**	9.99	1.04	100	10	1	101	10	1	100	9.97	1
Boost	653	67	7	851	83	8	1008	99	10	991	102	10
Privlet	453	51	4.90	615	62	5.89	732	74	7.16	730	73	7.21
Median-NF	104	**9.87**	**0.95**	**26**	**3.42**	**0.67**	**19**	**1.72**	**0.97**	**22**	**2.49**	**0.30**
Mean-NF	104	10	0.96	62	6.86	0.88	63	6.32	0.67	62	6.50	0.71
Median-SF	390	40	3.74	666	68	6.76	832	80	7.94	836	78	8.36
Mean-SF	535	56	5.59	662	68	6.80	263	79	8.20	831	81	8.08

Table 3 Comparison of the average absolute errors on unit-length queries

7.3 Evaluation on all methods

Next we compare the proposed methods with existing solutions. Tables 2 and 3 show the comparison results on unit-length queries in terms of squared error and absolute error, respectively. Each value in the tables equals to the corresponding SSE/SAE divided by the domain size n. The best results on each dataset and ϵ value are marked in bold. We made the following observations from these results.

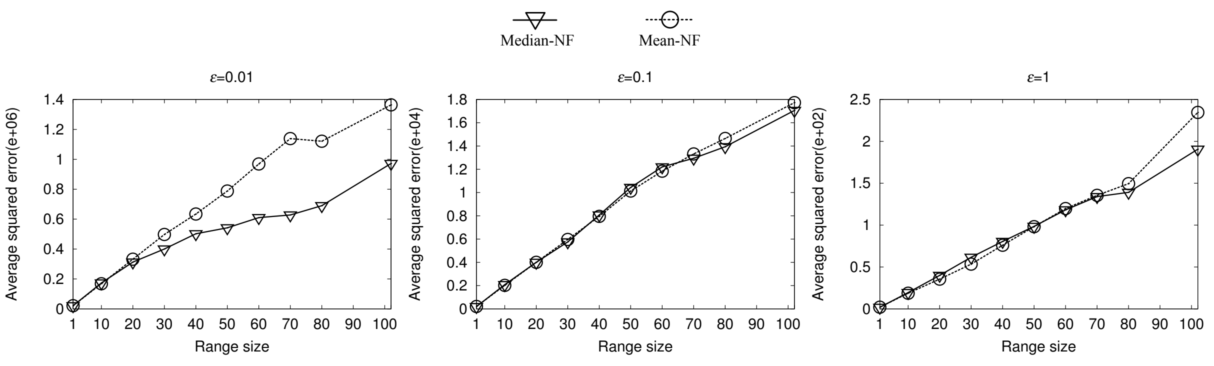
First, Dwork and NoiseFirst have significant advantages compared to the other solutions on unit-length range count queries. The main reason for their good performance is that both Dwork and NoiseFirst spend the whole privacy budget ϵ on the injected Laplace noise, leading to a smaller noise scale. On the other hand, the remaining methods (i.e., Boost, Privlet, and StructureFirst) all adopt a hierarchy tree structure to reorganize the original data in order to improve the query accuracy of range queries with large ranges. The utilization of such tree structures leads to the partition of the privacy budget ϵ to each tree level, resulting in a higher amount of noise added to each single count. Therefore, all these tree-based solutions are outperformed by Dwork and NoiseFirst on unit-length queries.

Second, NoiseFirst achieves better accuracy on *Search Log*, *NetTrace* and *Social Network* than Dwork, while the latter sometimes performs better than NoiseFirst on the *Age* dataset. This is because data values *Age* are more evenly distributed; consequently, it is more difficult for NoiseFirst to find a good histogram structure after adding random noise to the unit-length counts. On the other hand, the remaining three datasets have a large number of small unit counts, and NoiseFirst is more effective in merging consecutive small unit counts into bins, and eliminating the impact of the random noise in the bins. Especially, on *NetTrace* with $\epsilon = 0.1$, Median-NF is over 6 times better than Dwork in terms of squared error. To the best of our knowledge, NoiseFirst is the first solution outperforming Dwork for unit-length range count queries.

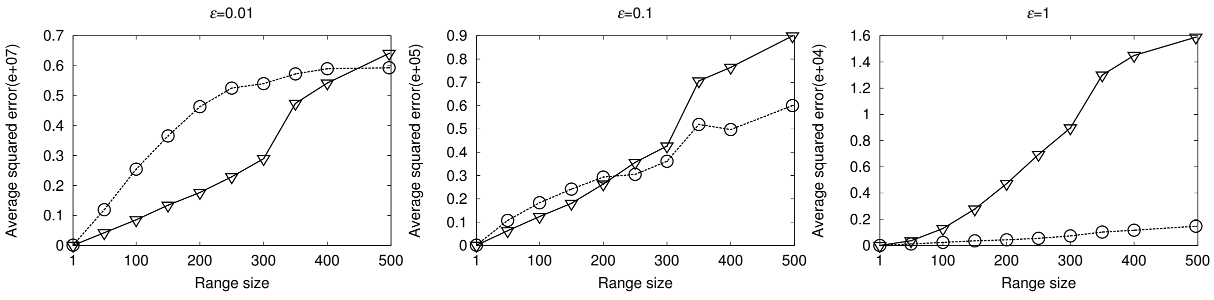
Third, Median-NF is better than Mean-NF in answering unit-length queries under most of the settings. This is because there exist some extreme counts after the injection of Laplace noise to each unit count. Median-NF is designed based on the median statistic is more robust to those extreme counts, whereas Mean-NF is more easily affected by those extreme values, leading to unstable mean values, and, consequently lower accuracy.

Finally, Median-NF shows better performance in terms of absolute errors. This is expected since Median-NF builds the histogram with the optimization objective of minimizing the SAE error.

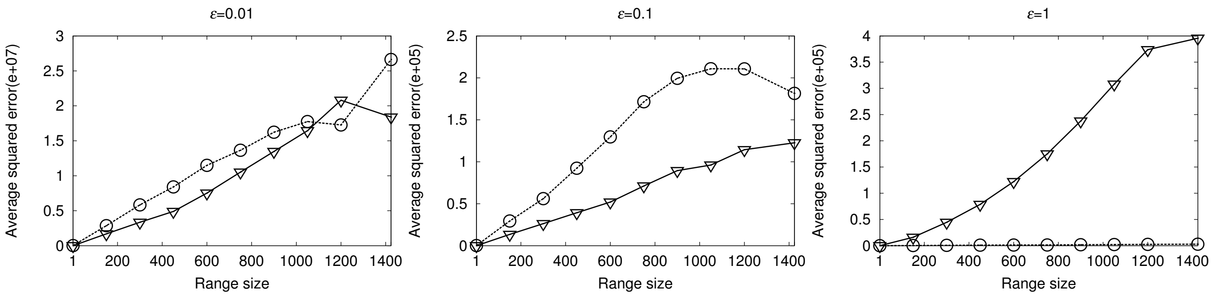
Next we evaluate the performance of all methods on queries with arbitrary ranges, reporting their accuracy in terms of average squared error, average absolute error as well as average relative error. Recall that StructureFirst sets $k = \frac{n}{10}$ in all settings, based on the conclusion drawn in Section 7.1. We select the best result for both NoiseFirst (NF) and StructureFirst (SF) based on the experimental results shown in Section 7.2. For example, the Mean-NF is chosen to participate into this round of comparison with $\epsilon = 1$ on *Social Network*. On the *Age* dataset with $\epsilon = 0.1$ Median-SF is picked up to compared with all the other state-of-the-art methods. For simplicity, regardless whether Mean-NF or Median-NF is used, we denote it as *NF* in the figures. Similarly, Mean-SF and Median-SF in the comparison are denoted as *SF*. In order to clearly display the relative performance of all methods, some points on large range sizes are discarded for Dwork and NoiseFirst in Figures 15-16. This is because these two methods have a linear growth trend with the increasing query range sizes.



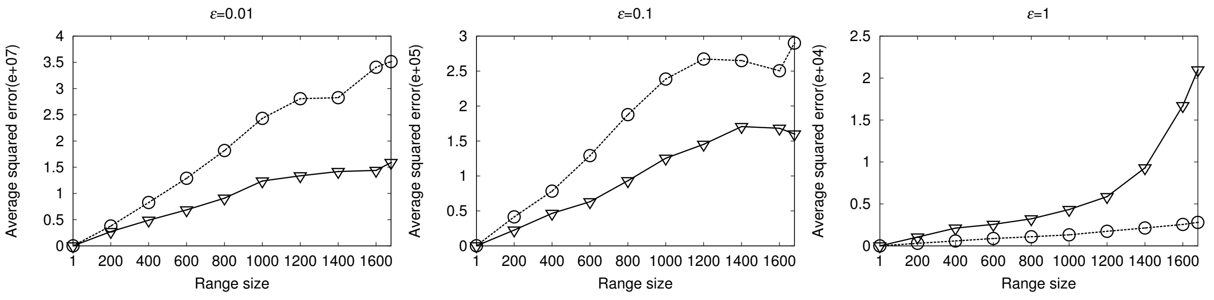
(a) On *Age*



(b) On *Search Log*



(c) On *NetTrace*



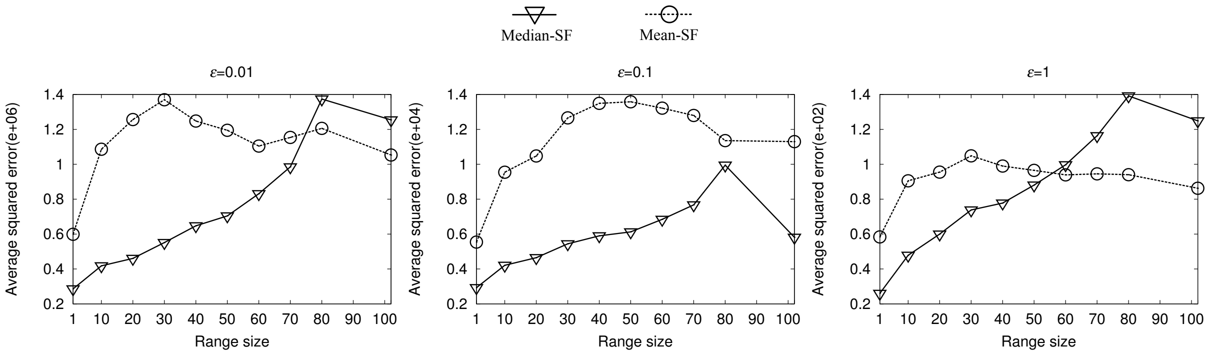
(d) On *Social Network*

Fig. 13 Average squared error of Mean-NoiseFirst and Median-NoiseFirst

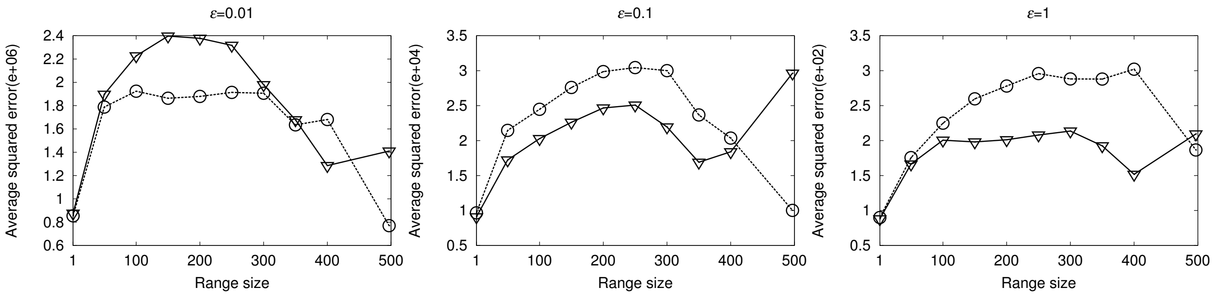
The results shown in Figures 15 and 16 lead to the following conclusions.

First, on all datasets, the average squared error of NoiseFirst and Dwork increases linearly with the query range. Both methods significantly outperform the other three algorithms on small range sizes. As the query length grows, the performance gap between NoiseFirst and Dwork expands, and NoiseFirst is better than Dwork by a certain margin on most of the query ranges. These results again verify the advantage of NoiseFirst for short ranges, which is consistent with the results in Tables 2 and 3.

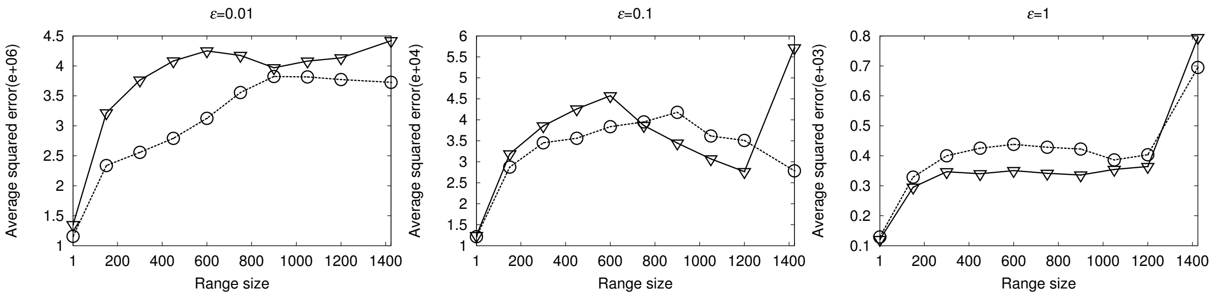
Second, the accuracy of both Privlet and Boost is high for queries with very long or very short ranges; for other queries, they tend to incur rather high errors. This is due to the binary tree structures used in their algorithms (note that the Haar wavelet used in Privlet is essentially a binary tree).



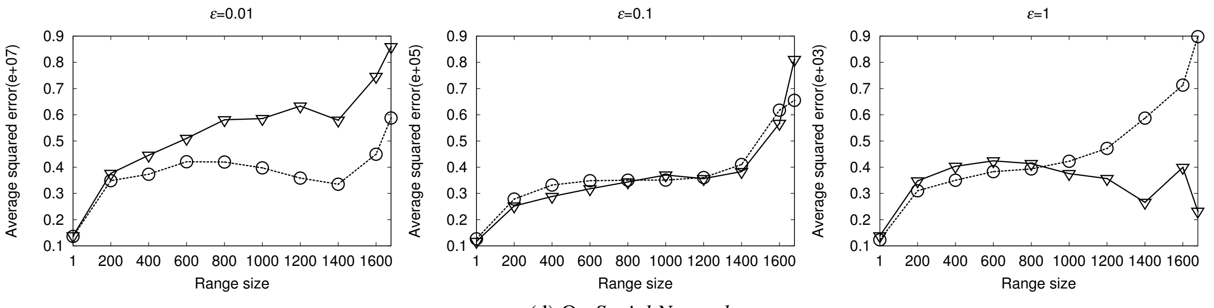
(a) On *Age*



(b) On *Search Log*



(c) On *NetTrace*



(d) On *Social Network*

Fig. 14 Average squared error of Median-StructureFirst and Mean-StructureFirst

Hence, very short queries favor both methods, as they only need to access a small number of nodes close to the leaf level. Very long queries, on the other hand, are answered mainly using a small number of nodes close to the top of the tree, leading to high accuracy. When queries with length in neither extremes, these methods return poorer results, since a large number of nodes from different levels of the tree are involved in query processing. In addition, we also observe that Privlet outperforms Boost on most of the datasets with most of the settings of ϵ. This is because after reorganizing the original count sequence using a Haar Wavelet Tree, Privlet decides the amount of noise for each wavelet coefficient on each tree level based on its weight, and a larger weight leads to a smaller noise. Such a strategy ensures that the main trend of the original count sequence is not altered significantly by the noise. However, in Boost, all intermediate counts on different tree levels are treated equally and added with the same scale of Laplace noise. It is the strategy of adding noise that helps Privlet outperform Boost significantly.

Third, StructureFirst produces more accurate estimates than both Privlet and Boost on all datasets under most settings of range sizes. Specifically, StructureFirst performs on average twice as well as those two methods. The main reasons are (i) that StructureFirst utilizes the count similarities in consecutive bins; (ii) that StructureFirst avoids building a large tree over the whole data domain; reducing the number of nodes required to answer a range query; and (iii) that bins in a domain partition tend to have similar counts, and, thus, building trees separately on each partition benefits from both the consistency and the similarities amongst bins. Another interesting observation concerning StructureFirst is that when the range size is as large as the domain size, Boost and Privlet have better accuracy than StructureFirst. This is because both Boost and Privlet build hierarchical trees to summarize data on the whole domain, and hence, have a better overview on range counts on the complete domain.

Figure 17 exhibits the relative error of all methods on the four datasets. Similar to [24], the relative error of a variable C is computed as $\frac{|C-C_{act}|}{max\{c_{act},S\}}$, where C_{act} denotes the actual query result, and s is a *sanity bound* that alleviates the effects of the queries with extremely small results. In our experiment, parameter s is set to a value that equals to 0.1% of the number of tuples in the dataset. From the figure, we see that the error of StructureFirst is consistently lower than that of the other methods on larger range sizes. NoiseFirst is often the best method in handling the queries with small range sizes. On the *Age* dataset, all methods tend to have a relatively small error. This is because the bin counts of *Age* are apparently larger than that of the other three datasets due to its largest tuple cardinality (i.e., 100,078,675). Therefore, when the same scale of noise is added to *Age*, its relative error is not large due to its high count values. Another observation in Figure 17 is that when the privacy budget ϵ is as small as 0.01, all methods incur relative errors higher than 20. This indicates the trade-off between the utility of the query results and the degree of privacy protection.

7.4 Summary

The experimental results show that compared to the state-of-the-art methods, NoiseFirst usually returns more accurate results for range count queries with short ranges, especially for unit-length queries. Since the error of unit-length queries has direct impacts on the histogram shape, the histogram produced by NoiseFirst provides better visualization of the data distribution. StructureFirst, on the other hand, has apparent advantages with queries with larger ranges. Thus, a query executer can provide the users more precise results by querying the DP-complaint histogram published by StructureFirst. To sum up, NoiseFirst and StructureFirst complement each other, and they can be simultaneously embedded into a DBMS for more accurate query processing while protecting data privacy.

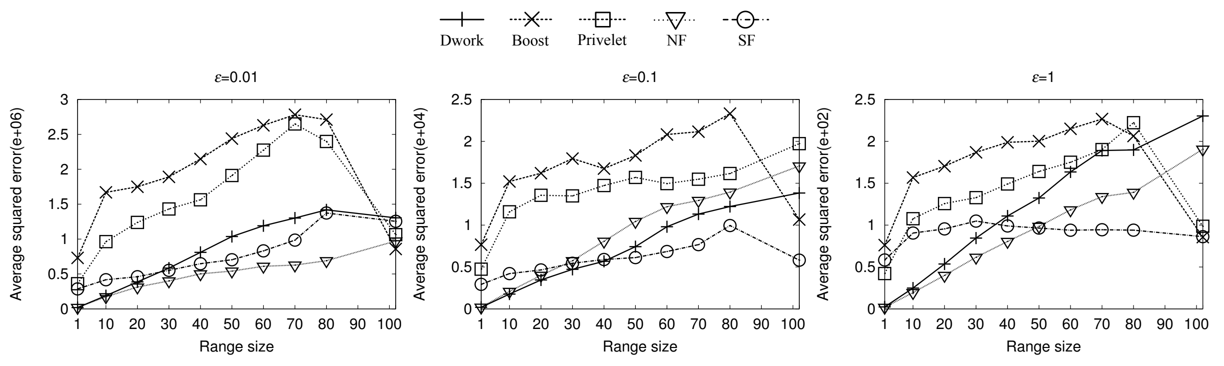
8 Related Work

Numerous techniques have been proposed for publishing various types of data while achieving differential privacy (see [6] for a survey). For example, Bhaskar et al. [2] investigate how frequent itemsets from transaction data can be published. Friedman et al. [10] devise methods for constructing decision trees. Korolova et al. [16] and Götz et al. [11] present methods for publishing statistics in search logs, while McSherry and Mahajan [19] develop techniques for network trace analysis. Zhang et al. [26] study the problem of differentially private regression analysis. Finally, Mohan et al. build a general-purpose system GUPT [20] for applications where the level of required privacy protection change with time.

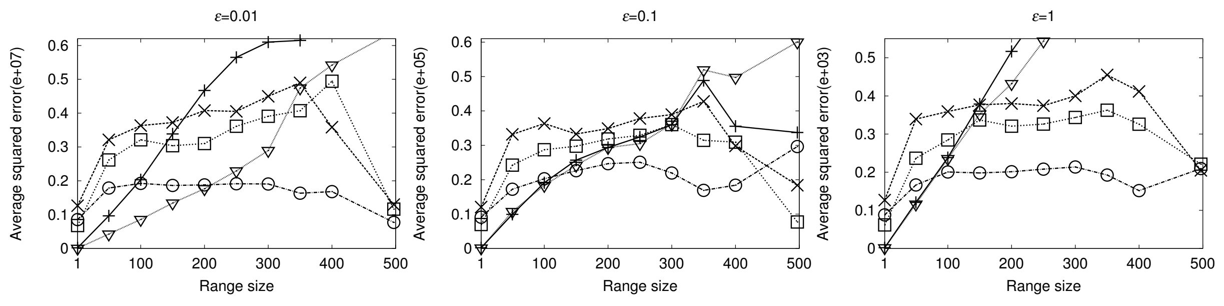
Among the existing approaches, the ones most related to ours are by Blum et al. [3], Hay et al. [13], Xiao et al. [24], and Li et al. [18, **?**]. Specifically, Blum et al. [3] propose to construct one-dimensional histograms by dividing the input counts into several bins, such that the sum of counts in each bin is roughly the same. The bin counts are then published in a differentially private manner. This approach, however, is shown to be inferior to the method by Hay et al. [13] in terms of the variance of the noise in range count query results.

Hay et al.'s method works by first (i) computing the results of a set of range count queries (with Laplace noise injected), and then (ii) refining the noisy results by exploiting the correlations among the queries. The results obtained thus can then be used to answer any range count queries, and the variance of noise in the query answers is $O(\log^3 n)$, where n is the number of counts in the input data. Hay et al.'s method, as with our solutions, is designed only for one-dimensional data. Meanwhile, Xiao et al. [24] develop a wavelet-based approach that can handle multi-dimensional datasets, and it achieves a noise variance bound of $O(\log^{3d} n)$, where d is the dimensionality of the data set. As shown in our experiments, however, both Hay et al.'s and Xiao et al.'s approaches are outperformed by our techniques in terms of query accuracy.

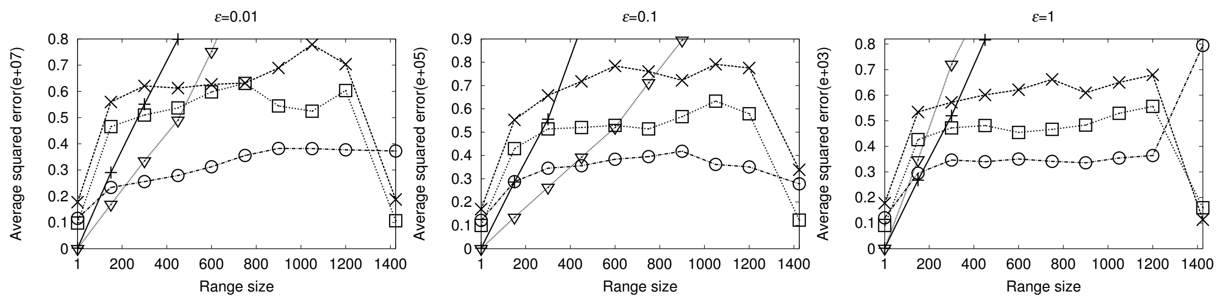
Li et al. [18, **?**] propose an approach that generalizes both Hay et al.'s and Xiao et al.'s techniques in the sense that it can achieve optimal noise variance bound for a large spectrum of query workloads, which is later improved in [25]. In contrast, Hay et al.'s and Xiao et al.'s techniques only optimizes the accuracy of range count queries. Nevertheless, Li et al.'s approach incurs significant computation cost, and hence is inapplicable on large data sets.



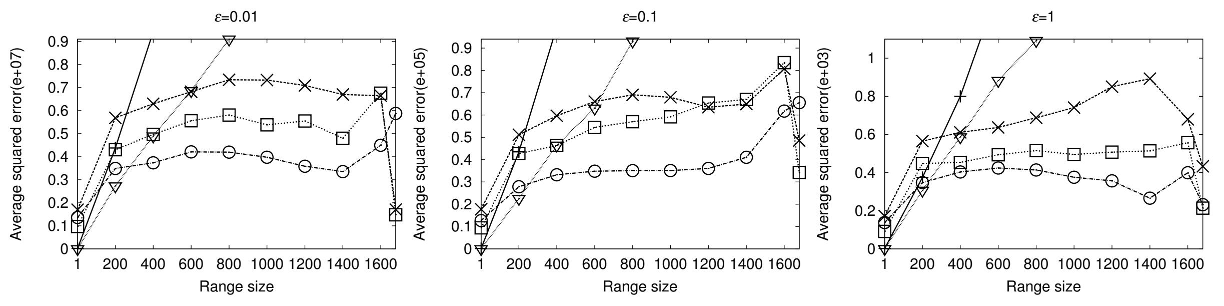
(a) On *Age*



(b) On *Search Log*



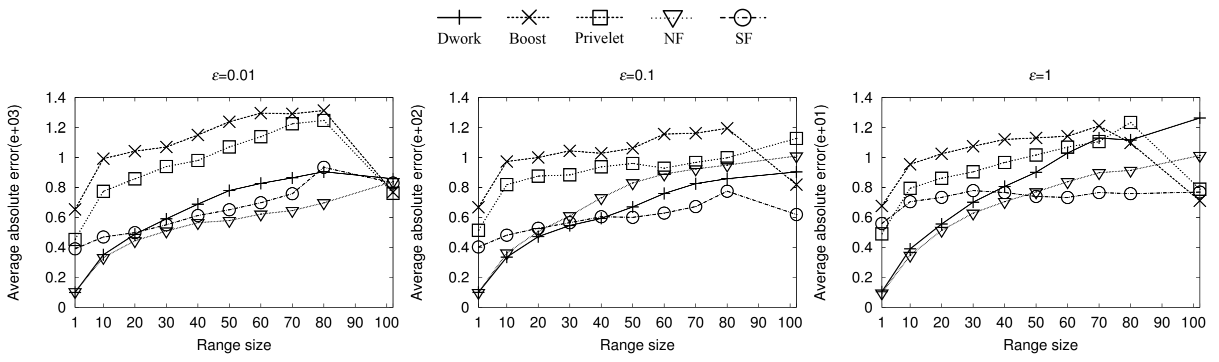
(c) On *NetTrace*



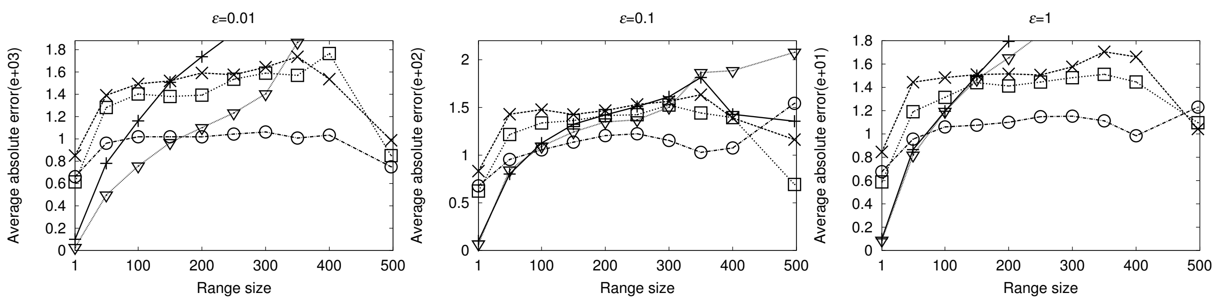
(d) On *Social Network*

Fig. 15 Average squared error of different methods

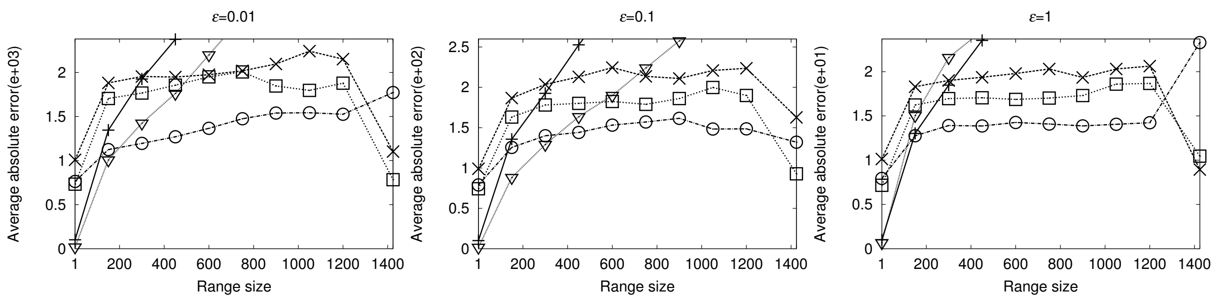
There are also quite a lot works handling the multi-dimensional data in a differentially private way [**?**][**?**] [**?**]. Cormode et al. [**?**] develop a general framework to privately release the multi-dimensional sparse data in an efficient way. Their efficiency stems from the utilization of a certain compact summary for the noisy data with the same privacy guarantee. The main difference between our proposals and Cormode et al.'s work is that they focus on optimizing the cost of computing the private summaries while we target at improving the query accuracy at full stretch. Li et al. in [**?**] present a couple of multidimensional partitioning strategies for differentially private histogram release. Their first method constructs a fine-grained grid on multi-dimensional data. They then consider the data uniformity in consecutive grids and discuss the problem of partitioning the noisy count tables using the classical kd-tree. At first glance, their kd-tree based solu-



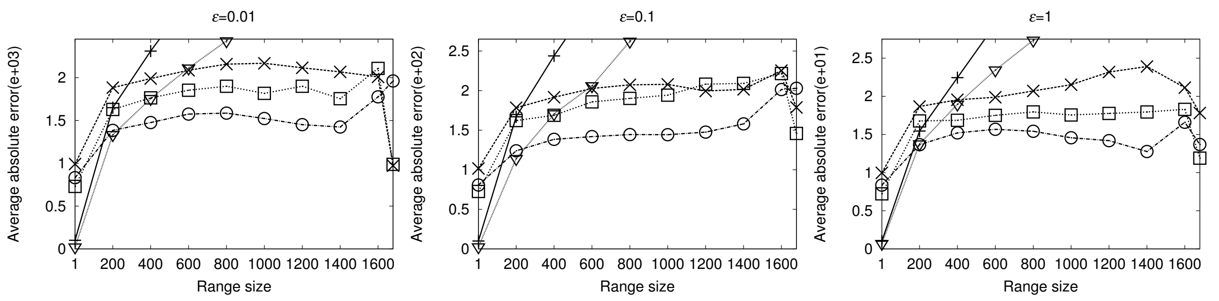
(a) On *Age*



(b) On *Search Log*



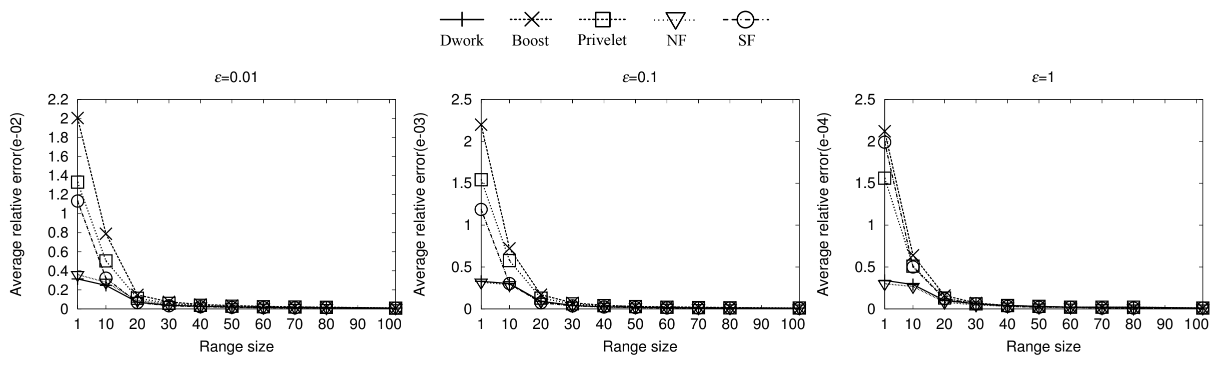
(c) On *NetTrace*



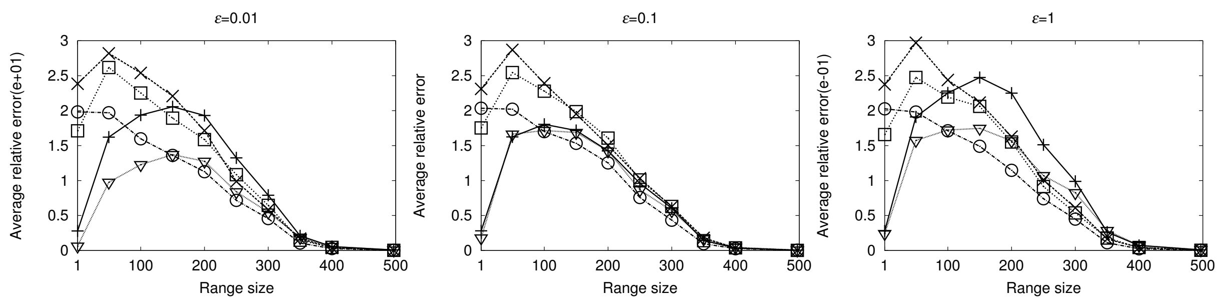
(d) On *Social Network*

Fig. 16 Average absolute error of different methods

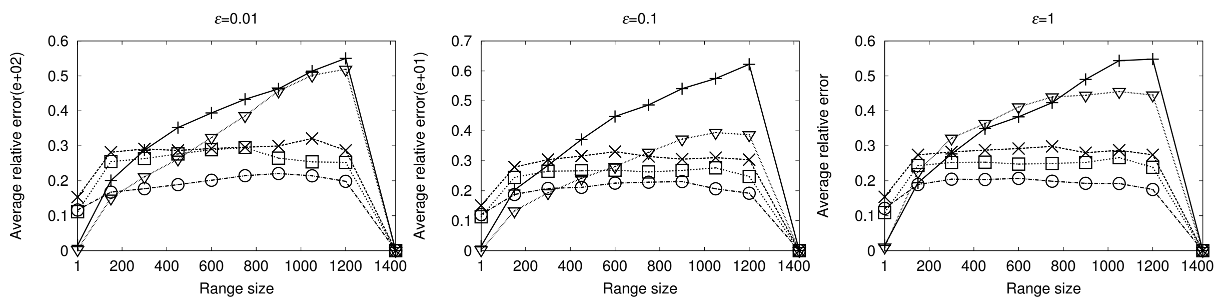
tion is seemingly similar to our StructureFirst method, since we both at first merge consecutive similar counts into different partitions and then consume a certain amount of privacy budget to hind the optimal partition structure which may compromise users' privacy. However, their kd-tree based solution simply assigns $\frac{\epsilon}{2}$ budget to partition adjustment while we theoretically give a near-optimal budget assignment strategy for users. In [**?**], Cormode et al. focus on the problem of multi-dimensional query processing on spatial data. In particular, their query processing is strongly supported by the "private spatial decomposition" indices which take into account both of the data-dependent partitioning (e.g., quadtree) and data-dependent partitioning (e.g., kd-tree). Although they show that their spatial decomposition indices highly improve the data utility of the query results than Hay et al. and Xiao et al.'s methods, their problem is different



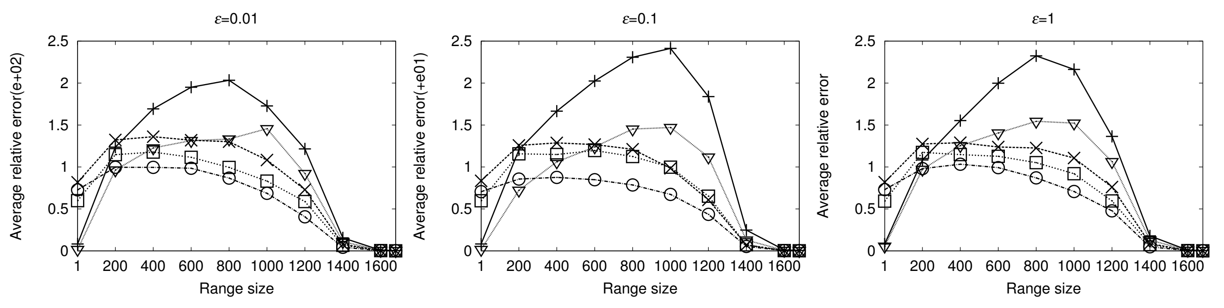
(a) On *Age*



(b) On *Search Log*



(c) On *NetTrace*



(d) On *Social Network*

Fig. 17 Average relative error of different methods

from that of ours. For we are centered on publishing the differentially private histogram to users while they concern with accurately answering users' queries by accessing different level of nodes in the index structures.

In addition, there exist several techniques that address problems similar to (but different from) ours. Barak et al. [1] and Ding et al. [5] propose methods for releasing *marginals*, i.e., projections of a data set onto subsets of its attributes. The core ideas of their methods are to exploit the correlations among the marginals to reduce the amount of noise required for privacy protection. However, neither Barak et al.'s nor Ding et al.'s method can be applied for our problem, as we consider the release of *one* histogram instead of *multiple* marginals.

Xiao et al. [23] devise a differentially private approach that optimizes the relative errors of a given set of count queries. The approach targets the scenario where the count queries overlap with each other (i.e., there exist at least one tuple that satisfies multiple queries), in which case adding less noise in one query result may necessitate a larger amount of noise for another query, so as to ensure privacy protection. Under this setting, Xiao et al.'s approach calibrates the amount of noise in each query result, such that queries with smaller (larger) answers are likely to be injected with less (more) noise, which leads to reduced relative errors. This approach, however, is inapplicable for our problem, since the count queries concerned in a histogram are mutually disjoint.

Rastogi and Nath [21] develop a technique for releasing aggregated results on time series data collected from distributed users. The technique injects noise into a time series by first (i) deriving an approximation of the time series and then (ii) perturbing the approximation. Our histogram construction algorithm is similar in spirit to Rastogi and Nath's technique, in the sense that our algorithm (i) approximates a set D of counts with several bins and (ii) perturbs the bin counts. One may attempt to apply Rastogi and Nath's technique for histogram construction, by regarding the set D of counts as a time series. This approach, however, would lead to highly suboptimal results, since Rastogi and Nath's technique assumes that all information in a time series concerns the same user, in which case changing one user's information could completely change all counts in D.

9 Conclusion

In this paper, we present two new differential privacy mechanisms, namely NoiseFirst and StructureFirst, supporting arbitrary range count queries on numeric domains. Utilizing commonly used histogram techniques, our mechanisms generate randomized estimations on the counts in the database. By summarizing similar consecutive counts using aggregates (mean of median), our mechanisms dramatically improve the query accuracy. Experimental results on four real data sets show that NoiseFirst outperforms the Laplace Mechanism, which is the best solution supporting the range count queries with small lengths, by up to 6 times. Our second solution, namely StructureFirst, performs on average twice as good as the state-of-the-art methods on queries with longer ranges.

References

1. B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*, pages 273–282, 2007.
2. R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *KDD*, pages 503–512, 2010.
3. A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.
4. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms, Second Edition.*, pages 185–192. MIT Press and McGraw-Hill, 2001.
5. B. Ding, M. Winslett, J. Han, and Z. Li. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD*, pages 217–228, 2011.
6. C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.
7. C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.
8. C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of LP decoding. In *STOC*, pages 85–94, 2007.
9. C. Dwork, G. N. Rothblum, and S. P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60.
10. A. Friedman and A. Schuster. Data mining with differential privacy. In *KDD*, pages 493–502, 2010.
11. M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Publishing search logs - a comparative study of privacy guarantees. In *TKDE*, in press.
12. S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM TODS*, 31(1):396–438, 2006.
13. M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.
14. N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genetics*, 4(8), 2008.
15. H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *VLDB*, pages 275–286, 1998.
16. A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180, 2009.
17. S. Kotz, T. Kozubowski, and K. Podgórski. *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*, pages 23–23. Birkhäuser Publication, 2001.
18. C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134, 2010.
19. F. McSherry and R. Mahajan. Differentially-private network trace analysis. In *SIGCOMM*, pages 123–134, 2010.
20. P. Mohan, A. Thakurta, E. Shi, D. Song, and D. E. Culler. Gupt: privacy preserving data analysis made easy. In *SIGMOD*, pages 349–360, 2012.
21. V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, pages 735–746, 2010.
22. R. Wang, Y. Li, X. Wang, H. Tang, and X. Zhou. Learning your identity and disease from research papers: Information leaks in genome wide association study. In *ACM CCS*, 2009.
23. X. Xiao, G. Bender, M. Hay, and J. Gehrke. ireduct: differential privacy with reduced relative errors. In *SIGMOD*, pages 229–240, 2011.
24. X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, pages 225–236, 2010.
25. G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: Optimizing batch queries under differential privacy. In *PVLDB*, 2012.
26. J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional mechanism: Regression analysis under differential privacy. In *PVLDB*, 2012.