The VLDB Journal (2013) 22:797–822
DOI 10.1007/s00778-013-0309-y

REGULAR PAPER

Differentially private histogram publication

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

Received: 22 June 2012 / Revised: 26 January 2013 / Accepted: 27 February 2013 / Published online: 20 April 2013

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-compliant 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 datasets, confirm that our two proposals output highly accurate query answers and consistently outperform existing competitors.

Keywords Differential privacy · Database query processing · Histogram

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

J. Xu
e-mail: xujia@ise.neu.edu.cn

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

Y. Yang
e-mail: yin.yang@adsc.com.sg

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

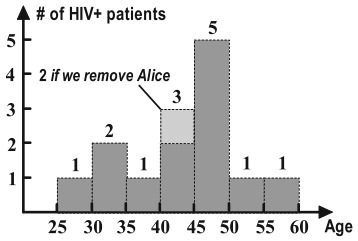
M. Winslett
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, USA
e-mail: winslett@illinois.edu

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 become 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

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



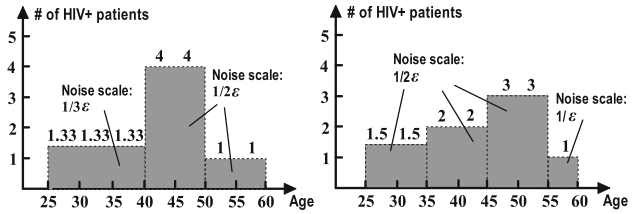
(b) Unperturbed histogram

Fig. 1 Example sensitive dataset and its corresponding histogram

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 [16] describes an attack that infers whether a person has participated in a certain 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 [26] 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) [9–11,15,21,28] 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 a DP-compliant histogram, which is a common tool for presenting the distribution of a random variable. Figure 1a shows sample records in an imaginary sensitive dataset about HIV-positive patients, and Fig. 1b illustrates its corresponding histogram showing the age distribution of such patients. Such a histogram can be commonly found, e.g., in the published statistics by Singapore's Ministry of Health.[2] The application of DP to such a histogram 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 [9] (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



(a) Optimal histogram structure **(b)** Poor histogram structure

Fig. 2 Impact of histogram structures ($k = 3$) on noise scales

counts, which is the maximum possible changes in the query results if we remove one record from (or add one record 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 [20], 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 Figure 2a exhibits a different histogram of the data set in Fig. 1a 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 an original bin in the histogram in Fig. 1b, e.g., 25–30. In the histogram in Fig. 2a, 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. 1b.

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*, as the mean statistic (e.g., 1.33 for bin 25–30 in Fig. 2a) replaces their respective original values (1). Accordingly, the quality

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

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

[3] An alternative definition of sensitivity [9] 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.

of the histogram structure depends on the balancing between the information loss and the error reduction. Figure 2b, for example, shows yet another histogram for the same dataset 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 solutions for DP-compliant histogram computation, namely *NoiseFirst* and *StructureFirst*. The former determines the histogram structure *after* the noise injection, and the latter derives the histogram structure *before* the addition of the noise. In particular, NoiseFirst can be used to improve the accuracy of an already published histogram using an existing method, e.g., [9]. We discuss both of the approaches above under two generic histogram construction strategies, i.e., *mean-histograms* and *median-histograms*. Specifically, mean-histograms adopt *means* of all counts in each histogram bin as the representative count. In median-histograms, medians are used instead means to summarize the set of counts for all histogram bins. Although median-histograms are rarely used in conventional databases, we show that it is a better match to differential privacy, thus improving the accuracies of the histograms.

Furthermore, we adapt DP histograms to answer arbitrary range count queries, which has drawn considerable research attention (e.g., [9,15]). For such queries, NoiseFirst achieves better accuracy for short ranges, whereas StructureFirst is more suitable for longer ones. Extensive experiments using several real datasets demonstrate that our two proposals, namely NoiseFirst and StructureFirst output highly accurate histograms, and significantly outperform existing methods for range count queries. In following, Sect. 2 provides necessary background on histogram construction and differential privacy. Sections 3 and 4 present NoiseFirst and StructureFirst, respectively, for mean-histograms. Section 5 extends our solutions to median-histograms. Section 6 discusses the range count query processing. Section 7 contains a thorough experimental study. Section 8 overviews existing DP techniques. Finally, Sect. 9 concludes the paper.

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 Fig. 1

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 count sequence D, as follows.

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

$\mathrm{SSE}(H, D)$ can also be interpreted as the SSE for all unit-length range count queries. Given the structure of the histogram, i.e., the interval $[l_j, r_j]$ for each bin B_j, the optimal value of c_j for B_j minimizing $\mathrm{SSE}(H, D)$ 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 [14,17] aims to identify the optimal histogram structure H^* containing k bins (k is a given parameter) that minimizes $\mathrm{SSE}(H, D)$.

Jagadish et al. [17] propose a dynamic programming solution, with time complexity $O(n^2 k)$ and space complexity $O(nk)$. Figure 3 lists the intermediate results of the dynamic programming process, using the data sequence in Fig. 1b 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 sequence $D_i = \{x_1, \ldots, x_i\}$. Let $SSE(D, l, r)$ 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, $\mathrm{SSE}(D, l, r) = \sum_{i=l}^{r} (x_i - \bar{x}(l, r))^2$.

The dynamic programming algorithm in [17] recursively computes the minimum SSE, i.e., $T(n,k)$, for building the optimal histogram, using the following equation:

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

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 Fig. 3). In our example, the optimal histogram is $H^* = \{(1, 3, 1.33), (4, 5, 4), (6, 7, 1.0)\}$.

2.2 Differential privacy

Given a count sequence $D = \{x_1, x_2, \ldots, x_n\}$, another sequence D' is a neighboring 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 $D' = \{x_1, x_2, \ldots, x_{m-1}, x_m \pm 1, x_{m+1}, \ldots, x_n\}$. A histogram publication mechanism Q satisfies ϵ-differential privacy (ϵ-DP), if it outputs a randomized histogram H, such that $\forall D, D', H: \ \Pr(Q(D) = H) \le e^{\epsilon} \times \Pr(Q(D') = H)$, where D and D' denote two arbitrary neighboring 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 [9], 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 two neighboring databases D and D', i.e., $\Delta = \max_{D,D'} \|Q(D) - Q(D')\|_1$.

Dwork et al. [9] prove that differential privacy can be achieved by perturbing each output of Q by an independent random noise that follows a zero-mean Laplace distribution with a scale of $b = \frac{\Delta}{\epsilon}$. In our problem, note that the simple solution that simply adds Laplace noise to each bin count of the optimal histogram (e.g., computed with the algorithm in [17]) 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. 1b 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 Fig. 4. Since the published optimal histogram structure is the one shown in Fig. 2a), the adversary infers that *Alice must be an HIV+ patient*, leading to a privacy breach.

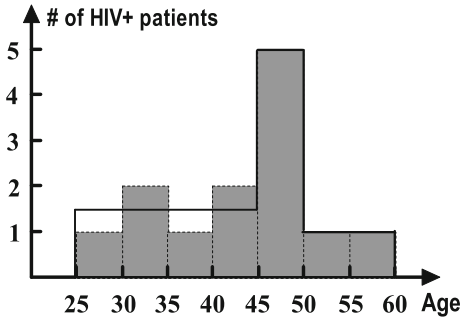


Fig. 4 Optimal histogram when Alice is excluded from the dataset

Table 1 summarizes frequently used notations throughout the paper.

3 NoiseFirst

In this and the next section, we present our proposals under the context of the mean-histogram, in which each bin outputs the mean value of the set of counts in it. 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) [9]. Clearly, the sensitivity of this step is 1, since adding or removing any record can change the count of one bin by at most 1. Therefore, it suffices to inject Laplace noise

Table 1 Summary of frequent notations

Notation	Meaning
Δ	Query sensitivity
$D = \{x_1, \ldots, x_i, \ldots, x_n\}$	A count sequence
$D' = \{x_1, \ldots, x_i \pm 1, \ldots, x_n\}$	A neighboring count sequence of D
$\hat{D} = \{\hat{x}_1, \ldots, \hat{x}_n\}$	A noisy count sequence of D
$D_i = \{x_1, \ldots, x_i\}$	A partial count sequence of D
$B_i = (l_i, r_i, c_i)$	The ith histogram bin with left boundary l_i, right boundary r_i and representative value c_i
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 count sequence D_i
$\mathrm{SSE}(D, l, r)/\mathrm{SAE}(D, l, r)$	The SSE/SAE error if we merge a partial count sequence $\{x_l, \ldots, x_r\}$ into a single bin
$\mathrm{SSE}(H, D)/\mathrm{SAE}(H, D)$	The SSE/SAE error if we build histogram H on D

with magnitude $\frac{1}{\epsilon}$ into each bin to satisfy ϵ-DP. The result is a noisy sequence $\hat{D} = \{\hat{x}_1, \ldots, \hat{x}_n\}$. In the second step, NoiseFirst computes the optimal histogram structure based on the noisy count sequence $\hat{D}$, using the dynamic programming algorithm [17]. The pseudocode of NoiseFirst is listed in Algorithm 1. Apparently, NoiseFirst can be used as a post-processing step to optimize a 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 count subsequence $\{\hat{x}_{l_j}, \ldots, \hat{x}_{r_j}\}$.

Since NoiseFirst computes the histogram structure based on the noisy counts, a natural question is whether the computed histogram indeed has high quality. To answer this, in the rest of the section, we conduct a theoretical analysis on the expected SSE incurred by NoiseFirst. The main objective in the analysis is to derive a connection between (i) the error of the optimal histogram built on the noisy count sequence $\hat{D}$ and (ii) the error of the same histogram built on the original count sequence D. First, we analyze the impact of merging consecutive noisy counts into a single bin. The following two lemmata quantify the expected errors of a single merged bin on the noisy count sequence and the original count sequence, respectively.

Lemma 1 *Given a 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.

$$\mathbb{E}\left\{\sum_{i=l}^{r}(X_i - C)^2\right\} = \mathbb{E}\left\{\sum_{i=l}^{r}(X_i)^2 - \frac{1}{s}\left(\sum_{i=l}^{r} X_i\right)^2\right\}$$

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

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

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

$$= \mathrm{SSE}(D, l, r) + \frac{2(s-1)}{\epsilon^2}$$

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

The above lemma evaluates the SSE of a single bin built on the noisy count sequence $\hat{D}$. The following lemma shows how to estimate the error of a single bin with respect to the original count sequence D.

Lemma 2 *Given a 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 expected error as follows. Notations X_i, C and s have the same meanings as defined in the proof of Lemma 1.

$$\mathbb{E}\left\{\sum_{i=l}^{r}(x_i - C)^2\right\}$$

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

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

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

$$= \mathrm{SSE}(D, l, r) + \frac{2}{\epsilon^2}$$

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.,

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

Lemma 2 shows that the expected error of histogram with a single bin is $\mathrm{SSE}(D, l, r) + \frac{2}{\epsilon^2}$. 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., $\epsilon < \sqrt{\frac{2(r-l)}{\text{SSE}(D,l,r)}}$. 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. Specifically, instead of using noisy counts $\{\hat{x}_{l_j}, \ldots, \hat{x}_{r_j}\}$, H_k calculates the mean count for bin B_j using the original counts, i.e., $(x_{l_j} + \cdots + x_{r_j})/(r_j - l_j + 1)$. It is straightforward to see that $\text{Error}(H_k, D) = \sum_{j=1}^{k} \text{SSE}(D, l_j, r_j)$, which helps to prove the following theorem.

Theorem 1 *Given H_k and $\hat{H}_k^*$ as defined above, the expected sum of squared 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 NoiseFirst, the above theorem indicates us that optimizing the histogram by minimizing the $\text{Error}(\hat{H}_k^*, \hat{D})$ leads to the minimization of the $\text{Error}(\hat{H}_k^*, D)$, i.e.,

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

Equation 3 provides the intuition behind the correctness of NoiseFirst method in Algorithm 1, which runs the dynamic programming on the noisy count sequence $\hat{D}$.

It remains to clarify how to select an appropriate value for the parameter k. Since the noisy data already satisfy the ϵ-DP, we simply execute the algorithm n times, with parameter values $k = 1, \ldots, n$ and return the best k that minimizes the expected SSE. To calculate the expected SSE, although it is impossible to directly evaluate the expected error on D, we can utilize Theorem 1 again. Specifically, if $\hat{H}_k^*$ is the optimal histogram returned by the Algorithm 1 with k ($k = 1, \ldots n$) bins, the final result histogram $\hat{H}_{k^*}^*$ is selected based on the following optimization objective:

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

Algorithm 2 details the selection of the optimal parameter k^* for NoiseFirst. The notation $\hat{T}_{\text{SSE}}[i, j]$ represents the minimum SSE of merging the partial noisy count sequence $\hat{D}_i$ into j bins. The algorithm first fills the table $\hat{T}_{\text{SSE}}$ (Lines 1-6) by running dynamic programming. Then, k^* is obtained by comparing the expected SSE under different ks based on the Eq. 4 (Lines 7-12). Since Algorithm 2 runs entirely on the noisy count sequence $\hat{D}$ rather than the original sequence D, it does not consume any privacy budget.

By averaging noisy counts, the zero-mean Laplace noise added to each count is smoothed, and therefore, NoiseFirst can obtain an improved accuracy compared with the LM solution. On the other hand, NoiseFirst has the same sensitivity as LM and injects $\text{Lap}(\frac{1}{\epsilon})$ noise to each count during its first step. As shown in the Example 1 of Sect. 1, the sensitivity of the histogram can be further reduced, if we inject noise *after* the histogram construction. Next we describe a novel algorithm based on this idea.

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

4 StructureFirst

Unlike NoiseFirst, the StructureFirst algorithm computes the optimal histogram structure on the original count sequence, before adding the Laplace noise to each count. Note that the optimal histogram structure itself is sensitive; 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 count sequence $D_i = \{x_1, x_2, \ldots, x_i\}$, 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 k-bin histogram H_k^* on the basis of the original count sequence D. All the intermediate results are stored in a table, as is done in Fig. 3. To protect the structure of the optimal histogram, StructureFirst randomly moves the boundaries between histogram bins. Specifically, when choosing the boundary between the bin B_j and the bin B_{j+1}, namely r_j here, r_j is reset to the location $q \in [1, |D| + 1]$ with a probability proportional to:

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

Here, $E_{\text{SSE}}(q, j, r_{j+1})$ 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 $\{x_{q+1}, \ldots, x_{r_{j+1}}\}$ and c is the average count, it is straightforward to verify that:

Algorithm 3 Mean-StructureFirst (count sequence D with $|D| = n$, the number of bins k, privacy guarantee ϵ, count upper bound F)

1: Partition ϵ into two parts ϵ_1 and ϵ_2 that $\epsilon_1 + \epsilon_2 = \epsilon$.
2: Build the optimal histogram H_k^* on D 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 $r_k = n + 1$
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: set $r_j = q$ $(k - 1 \le q < r_{j+1})$ with a 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 the average count c_j for every bin with interval $[l_j, r_j]$.
10: Add Laplace noise to the average counts as $\hat{c}_j = c_j + Lap\left(\frac{1}{\epsilon_2(r_j - l_j + 1)}\right)$
11: Return histogram $\{(l_1, r_1, \hat{c}_1), \ldots, (l_k, r_k, \hat{c}_k)\}$

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

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

Figure 5 shows an example of StructureFirst. Assume that the total privacy budget ϵ equals to 1 and the optimal assignment for ϵ_1 and ϵ_2 are 0.05 and 0.95, respectively. Four rows $(L1 - L4)$ in the figure illustrate the results of each step in StructureFirst. Given an original count sequence $\{5, 7, 7, 6, 7, 5, 5\}$, StructureFirst first obtains its 3-bin optimal histogram structure (Line 2 of Algorithm 3), which is shown in the row $L1$ with different bins highlighted by different shades of gray. To protect the optimal histogram structure, the algorithm then randomly adjusts all bin boundaries in the optimal histogram structure (Line 7 of Algorithm 3). The adjusted histogram structure is displayed in the row $L2$. The row $L3$ shows that counts in each merged bin are approximated by their mean count. Finally, the algorithm injects $\mathrm{Lap}(\frac{1}{\epsilon_2(r_i - l_i + 1)})$ noise to each mean count and derives the output shown in the row $L4$. Recall that the state-of-the-art method, i.e., the Laplace Mechanism (LM) adds $\mathrm{Lap}(\frac{1}{\epsilon_2})$ noise to each count and obtains a noisy output as $\{3.20, 6.30, 7.43, 6.15, 7.06, 4.77, 2.98\}$. StructureFirst gets a smaller SSE, i.e., $(5 - 5.69)^2 + (7 - 5.69)^2 + \cdots + (5 - 5.48)^2 = 3.36$, which apparently outperforms that of the LM (SSE=8.06). The performance gain of StructureFirst stems from the reduction in noise scale after merging bins.

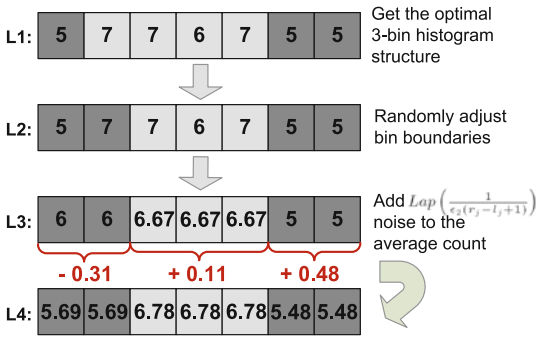


Fig. 5 Example of StructureFirst

Next we analyze the correctness and the expected accuracy of the StructureFirst algorithm and then discuss how to set the values of ϵ_1 and ϵ_2. Unlike the NoiseFirst, the StructureFirst does not decide the value of k itself; instead, users must specify the desired k before running the algorithm. In Sect. 7, we provide some empirical guidelines on choosing generally good value for the parameter k.

4.1 Correctness

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

Lemma 3 *Let F be an upper bound of the maximum count in any of the bins. Given two neighboring databases 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 neighboring 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 the bin (l_z, r_z, c_z) of $H^*(D_i, j)$. Since $H^*(D'_i, j)$ is the optimal histogram on D'_i with j bins, it is straightforward to know that given any histogram H' covering the interval $[1, i]$, we have

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

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 an upper bound on the error of such H' on D'_i.

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

When $x'_m = x_m + 1$, the Eq. 7 can be rewritten as:

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

When $x'_m = x_m - 1$, the Eq. 7 can be estimated as:

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

Combining Eqs. 8 and 9, we have

$$\begin{aligned}
\text{Error}(H^*(D'_i, j), D'_i) &\leq \text{Error}(H', D'_i) \\
&\leq \text{Error}(H^*(D_i, j), D_i) + 2F + 1
\end{aligned}$$

This reaches the conclusion of the lemma. □

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

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

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

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

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

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

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

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

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

Proof In the Algorithm 3, the output histogram relies on the execution of Lines 7 and 10, which are all independent of each other. Line 7 is run $(k-1)$ times and Line 10 is run exactly once. According to the Lemma 4, each execution of Line 7 costs a privacy budget $\frac{\epsilon_1}{k-1}$. Based on the concept of generalized sensitivity used in [28], the privacy cost of Line 10 is ϵ_2. It is because we can define $W_j = (r_j - l_j + 1)$ to meet the requirement of generalized sensitivity. Considering 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 the form of weighted sensitivity.

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

The total privacy budget spent in the algorithm is thus $(k+1)\frac{\epsilon_1}{k+1} + \epsilon_2 = \epsilon$, since $\epsilon_1 + \epsilon_2 = \epsilon$. □

4.2 Accuracy analysis

Next we quantify the expected SSE error of the StructureFirst.

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

Lemma 6 *The SSE of the histogram increases by no more than $\frac{n(k-1)(2F+1)}{\epsilon_1 \alpha}$, when we move the boundary by executing the Line 7 of Algorithm 3. Here, the factor α equals to:*

$$\alpha = \max\left\{n - \frac{\epsilon_1 n^2 F^2}{8(k-1)(2F+1)}, \exp\left\{-\frac{\epsilon_1 n F^2}{8(k-1)(2F+1)}\right\}\right\}$$

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

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

The righthand of Eq. 10 may be negative. To overcome such a negative lower bound, we derive another lower bound for $\sum_z \exp\left\{-\frac{\epsilon_1(E_{\mathrm{SSE}}(z)-E_{\mathrm{SSE}}(q^*))}{2(k-1)(2F+1)}\right\}$ below.

$$\begin{aligned}
&\sum_z \exp\left\{-\frac{\epsilon_1(E_{\mathrm{SSE}}(z)-E_{\mathrm{SSE}}(q^*))}{2(k-1)(2F+1)}\right\}\\
&\quad\geq \exp\left\{-\frac{\epsilon_1(E_{\mathrm{SSE}}(z)-E_{\mathrm{SSE}}(q^*))}{2(k-1)(2F+1)}\right\}\\
&\quad\geq \exp\left\{-\frac{\epsilon_1 nF^2}{8(k-1)(2F+1)}\right\} \qquad (11)
\end{aligned}$$

Therefore, the expectation of the additional error when the algorithm fails to set q to the optimal location q^* is at most:

$$\begin{aligned}
&\sum_q \{(E(q))\Pr(r_j=q)\} - E(q^*, j, r_{j+1})\\
&\quad= \sum_q \{(E(q)-E(q^*))\Pr(r_j=q)\}\\
&\quad= \sum_q\left\{(E(q)-E(q^*))\frac{\exp\left\{-\frac{\epsilon_1(E(q)-E(q^*)}{2(k-1)(2F+1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1(E(z)-E(q^*)}{2(k-1)(2F+1)}\right\}}\right\}\\
&\quad\leq \sum_q\left\{(E(q)-E(q^*))\frac{\exp\left\{-\frac{\epsilon_1(E(q)-E(q^*)}{2(k-1)(2F+1)}\right\}}{\alpha}\right\}\\
&\quad\leq \sum_q \frac{(k-1)(2F+1)}{\epsilon_1}\alpha^{-1}\\
&\quad\leq \frac{n(k-1)(2F+1)}{\epsilon_1\alpha} \qquad (12)
\end{aligned}$$

The first inequality is due to the Eqs. 10 and 11. The second inequality is the result by applying the Lemma 5. The last inequality holds since the number of candidate locations for q is no larger than the size of the database. □

Note that the cost analysis in the 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 shown in the Lemma 6. Combined with the error introduced by the Line 10 of Algorithm 3, the total expected SSE is bounded as being stated by the following theorem.

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

$$Error(H^*(D,k),D) + \frac{n(k-1)^2(2F+1)}{\epsilon_1\alpha} + \frac{2k}{(\epsilon_2)^2}$$

The error analysis in Theorem 3 shows the advantage of the StructureFirst. We can interpret the expected SSE of StructureFirst as $\mathrm{Error}(H^*(D,k),D) + O(\frac{k}{\epsilon^2})$ when ϵ_1 is small enough and ignore all constant items. This is a significant theoretical improvement over all previous methods, including the 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 $\mathrm{Error}(H^*(D,k),D)$ relies on the value of k and the original count sequence D. The other two parts are independent of D, 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:

$$\begin{aligned}
&\text{Minimize} \quad \left(\frac{n(k-1)^2(2F+1)}{\epsilon_1\alpha} + \frac{2k}{(\epsilon_2)^2}\right)\\
&\text{s.t.} \quad \epsilon_1+\epsilon_2=\epsilon
\end{aligned}$$

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

5 Publishing median-histograms

In previous two sections, we have discussed NoiseFirst and StructureFirst for mean-histograms, which report the mean count to approximate the set of counts in a merged bin. We observe that the mean may not be a good choice for our methods. For NoiseFirst, the main problem of the mean is that it is sensitive to outliers; specifically, a single very high or very low noisy count (note that NoiseFirst adds noise to the counts before building the optimal histogram) can significantly distort the mean value of a merged bin. In contract, the *median* is well known for its robustness against outliers, as it is less sensitive to any individual value. Therefore, reporting the median for each merged bin is often more friendly to NoiseFirst. Regarding StructureFirst, outputting the mean value incurs a rather high sensitivity when computing the optimal histogram structure, which depends on the maximum possible unit-count F (Lemma 3). As we show soon, using the median for each bin eliminates this dependence and thus

decreases the sensitivity, leading to a better histogram structure after the boundary adjustment; thus, StructureFirst for median-histograms can often achieve higher overall accuracy than StructureFirst for mean-histograms, even though the former adds a higher amount of noise to each bin.

One difference between mean and median is that the former minimizes the expected SSE 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 a count sequence $D = \{x_1, x_2, \ldots, x_n\}$, the SAE of a histogram $H = \{B_1, B_2, \ldots, B_k\}$ on D is calculated by Eq. 13.

$$\mathrm{SAE}(H, D) = \sum_{j}^{k} \sum_{l_j \le i \le r_j} |m_j - x_i| \tag{13}$$

Computing the optimal median-histogram is more complicated, in the sense that the median cannot be incrementally computed. Algorithm 4 shows the procedure of building an optimal median-histogram, without considering the privacy issue. Let $H^*(D_i, j)$ be the optimal j-bin histogram on prefix sequence D_i, and let table $\mathrm{TSAE}[i, j]$ record the SAE value corresponding to $H^*(D_i, j)$.

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

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

Compared with the optimal mean-histogram construction algorithm, the major bottleneck of Algorithm 4 lies in the calculation of the SAEs. Since median is an order statistic, the calculation of SAEs is naturally more time-consuming than that of SSEs. Fortunately, the median can still be computed in linear time [4].

5.1 Median-NoiseFirst

Algorithm 5 describes the Median-NoiseFirst framework. The major difference between Algorithms 5 and 1 is that Median-NoiseFirst builds the optimal histogram structure based on the median statistic and the SAE metric (see Algorithm 4), rather than the mean statistic and the SSE metric used in Algorithm 1. Next we analyze the expected SAE incurred by Median-NoiseFirst. The key is to make a connection between (i) the SAE of the optimal median-histogram built on the noisy count sequence $\hat{D}$ and (ii) the SAE of the same histogram toward 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 the noisy count sequence $\hat{D}$ and the original count sequence D, respectively. After that, we extend their conclusions to multiple bins in Theorem 4.

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 $\{\hat{x}_{l_j}, \ldots, \hat{x}_{r_j}\}$.

Lemma 7 *Given a 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 upper bound on the expected SAE of the bin with respect to* $\{\hat{x}_l, \hat{x}_{l+1} \ldots, \hat{x}_r\}$ *is*

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

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

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

The above inequality is valid due to the triangle inequality and the property of the median. The last equality holds for each zero-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 $\hat{D}$ when we merge 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 sequence D.

Lemma 8 *Given a 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 Laplace noise and the median in the noisy count sequence $\{X_l, X_{l+1}, \cdots, X_r\}$, respectively.

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

This completes the proof. □

In the following analysis, $\hat{H}_k^*$ denotes the optimal k-bin median-histogram built on the noisy count sequence $\hat{D}$, and H_k represents the median-histogram with the same k-bin structure as $\hat{H}_k^*$ but uses the median for each bin B_j on the original count sequence $\{x_{l_j}, \ldots, x_{r_j}\}$. Thus, we get $\mathrm{Error}(H_k, D) = \sum_{j=1}^{k} \mathrm{SAE}(D, l_j, r_j)$. Extending conclusions in Lemmata 7 and 8 to k bins, we reach Theorem 4.

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

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

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

Note that $\mathrm{Error}(H_k, D)$ is computed using the sensitive data, and thus, it cannot be used to evaluate the value of $\mathbb{E}(\mathrm{Error}(\hat{H}_k^*, D))$. However, the Theorem 4 tells that the $\mathbb{E}(\mathrm{Error}(\hat{H}_k^*, D))$ can be estimated by the $\mathbb{E}(\mathrm{Error}(\hat{H}_k^*, \hat{D}))$. Specifically, based on Theorem 4, we estimate the lower bound of the expected SAE for $\hat{H}_k^*$ on D based on the upper bound of the expected SAE for $\hat{H}_k^*$ on $\hat{D}$ as follows.

$$\mathbb{E}(\mathrm{Error}(\hat{H}_k^*, D)) \geq \mathbb{E}\left(\mathrm{Error}(\hat{H}_k^*, \hat{D}) - \frac{3(n-k)}{\epsilon}\right) \quad (14)$$

To minimize the expected value of $\mathrm{Error}(\hat{H}_k^*, D)$, we minimize its lower bound shown in the Eq. 14. Following the idea of Algorithm 2 in Sect. 3, we can also automatically derive the optimal k. Equation 15 summarizes this optimization problem.

$$\hat{H}^* \approx \arg\min_{\hat{H}_k^*} \mathbb{E}\left(\mathrm{Error}(\hat{H}_k^*, \hat{D}) - \frac{3(n-k)}{\epsilon}\right) \quad (15)$$

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 $[q+1, r_{j+1}]$.

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

1: Partition ϵ into two parts ϵ_1 and ϵ_2 that $\epsilon_1 + \epsilon_2 = \epsilon$.
2: Build the optimal k-bin median-histogram H_k^* based on Algorithm 4 and keep all the intermediate results, i.e., $\mathrm{TSAE}(i, j)$ for all $1 \leq i \leq n$ and $1 \leq j \leq k$.
3: Set $r_k = n$
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 $E_{SAE}(q, j, r_{j+1}) = \mathrm{TSAE}(q, j) + SAE(D, q+1, r_{j+1})$
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 $\{(l_1, r_1, \hat{m}_1), \ldots, (l_k, r_k, \hat{m}_k)\}$

Although the Median-StructureFirst resembles the Mean-StructureFirst described in the 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, the probability of setting the jth boundary to the location q is updated with $\exp\left\{-\frac{\epsilon_1 E_{\mathrm{SAE}}(q, j, r_{j+1})}{2(k-1)}\right\}$, as is shown in Line 7. Finally, the output histogram consists of medians instead of means (Line 10). Each median is then added with a Laplace noise having a scale of $\frac{1}{\epsilon_2}$ rather than the scale of $\frac{1}{\epsilon_2(r_j - l_j + 1)}$ in the Mean-StructureFirst. Although the scale of Laplace noise injected to each bin increases, the Median-StructureFirst still has its advantage compared to the Mean-StructureFirst. This is because the boundary adjustment

procedure in the Median-StructureFirst does not depend on the factor F, which decreases the randomness in the adjustment, leading to a better histogram structure.

5.2.1 Correctness

The worst case sensitivity analysis on the intermediate results during the calculation of the optimal median-histogram is given in Lemma 9. Here, $H^*(D_i, j)$ represents the optimal median-histogram constructed on the subsequence D_i using at most j bins.

Lemma 9 *Given two neighboring databases 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:*

$$|Error(H^*(D'_i, j), D'_i) - Error(H^*(D_i, j), D_i)| \leq 1$$

Proof Given two neighboring 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 $H^*(D_i, j)$, where m_z denotes the median of the bin (l_z, r_z). Since $H^*(D'_i, j)$ is the optimal j-bin histogram on subsequence D'_i, it is straightforward to know that given any histogram H' with j bins covering the interval $[1, i]$, we have

$$\text{Error}(H^*(D'_i, j), D'_i) \leq \text{Error}(H', D'_i) \tag{16}$$

Assume that H' is a special histogram on D'_i that uses 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. Based on the property of the median, we have $|m'_z - m_z| \leq 1$. In the following, we derive an upper bound for the SAE of H' on D'_i.

$$\begin{aligned}&\text{Error}(H', D'_i) - \text{Error}(H^*(D_i, j), D_i)\\ &\quad = m'_z - m_z \leq 1\end{aligned} \tag{17}$$

Based on Eqs. 16 and 17, we have

$$\begin{aligned}&\text{Error}(H^*(D'_i, j), D'_i) \leq \text{Error}(H', D'_i)\\ &\quad \leq \text{Error}(H^*(D_i, j), D_i) + 1\end{aligned} \tag{18}$$

This reaches the conclusion of the lemma. □

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

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

By utilizing Lemma 9, the conclusion of Lemma 10 can be easily reached following the proof line of Lemma 4. Therefore, we skip the proof procedure of Lemma 10 here.

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 a theoretical analysis similar to that for Theorem 2, we prove the correctness of the following theorem.

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

5.2.2 Accuracy analysis

Lemma 11 tells the SAE incurred when we randomly adjust a boundary in the optimal histogram structure.

Lemma 11 *The SAE of the histogram increases by no more than $\frac{n(k-1)}{\epsilon_1 \beta}$, when we move a boundary by executing the Line 7 of Algorithm 6. Here, the factor β equals to:*

$$max\left\{n\left(1 - \frac{\epsilon_1 nF}{2(k-1)}\right), \quad \exp\left\{\frac{-\epsilon_1 nF}{2(k-1)}\right\}\right\}$$

Proof We assume that the optimal $(j+1)$-bin histogram utilizes the optimal location q^* as the boundary between the jth bin and the $(j+1)$th bin to minimize the error of $\text{Error}(H^*(D_q, j), D_q) + \text{SAE}(q+1, r_{j+1})$. 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 $E_{\text{SAE}}(q, j, r_{j+1})$ to represent the SAE of the histogram that consists of the optimal histogram with j bins covering $[1, q]$ and the $(j+1)$th bin covering the domain of $[q+1, r_{j+1}]$. To simplify our symbolic system, we use $E(q)$ as a simplified representation of $E_{\text{SAE}}(q, j, r_{j+1})$ if variables j and r_{j+1} are clear in the context. The following inequalities pave the way to the proof.

$$\begin{aligned}&\sum_z \exp\left\{-\frac{\epsilon_1(E_{\text{SAE}}(z) - E_{\text{SAE}}(q^*))}{2(k-1)}\right\}\\ &\quad \geq \sum_z \left(1 - \frac{\epsilon_1(E_{\text{SAE}}(z) - E_{\text{SAE}}(q^*))}{2(k-1)}\right)\\ &\quad \geq n\left(1 - \frac{\epsilon_1 nF}{2(k-1)}\right)\end{aligned} \tag{19}$$

The above inequality is built based on the fact proposed in Lemma 5. Another easily obtained lower bound for the $\sum_z \exp\left\{-\frac{\epsilon_1(E_{\text{SAE}}(z) - E_{\text{SAE}}(q^*))}{2(k-1)}\right\}$ is given below in case that the lower bound in the Eq. 19 becomes negative.

$$\begin{aligned}&\sum_z \exp\left\{-\frac{\epsilon_1(E_{\text{SAE}}(z) - E_{\text{SAE}}(q^*))}{2(k-1)}\right\}\\ &\quad \geq \exp\left\{-\frac{\epsilon_1(E_{\text{SAE}}(z) - E_{SAE}(q^*))}{2(k-1)}\right\}\\ &\quad \geq \exp\left\{\frac{-\epsilon_1 nF}{2(k-1)}\right\}\end{aligned} \tag{20}$$

Similar to the derivation shown in Eq. 12, the expected increased SAE toward the optimal boundary q^*, is

$$\begin{aligned}&\sum_q \left\{(E(q))\Pr(r_j = q)\right\} - E(q^*, j, r_{j+1}) \\ &= \sum_q \left\{ (E(q) - E(q^*)) \frac{\exp\left\{-\frac{\epsilon_1(E(q)-E(q^*))}{2(k-1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1(E(z)-E(q^*))}{2(k-1)}\right\}} \right\} \\ &\leq \frac{n(k-1)}{\epsilon_1}\beta^{-1} \end{aligned} \tag{21}$$

This completes the proof of Lemma 11. □

Consequently, the total increased SAE after $(k-1)$ times of boundary adjustment (see Line 7 in Algorithm 6) is $(k-1)$ times of the error given in Lemma 11. Considering the SAE incurred by Laplace noise injection (see Line 10 in Algorithm 6) together, the total expected SAE of the published histogram by Algorithm 6 is upper bounded by Theorem 6.

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

$$Error(H^*(D,k), D) + \frac{n(k-1)^2}{\epsilon_1\beta} + \frac{2k}{(\epsilon_2)^2}$$

5.2.3 Budget assignment

Given the parameter k, a good budget assignment for ϵ_1 and ϵ_2 can be achieved by solving the following optimization problem which minimizes the expected SAE shown in Theorem 6. We use the numerical method as mentioned in Sect. 4.3 to obtain a near-optimal assignment for ϵ_1 and ϵ_2.

$$\begin{aligned}\text{Minimize} \quad & \left(\frac{n(k-1)^2}{\epsilon_1\beta} + \frac{2k}{(\epsilon_2)^2}\right) \\ \text{s.t.} \quad & \epsilon_1 + \epsilon_2 = \epsilon\end{aligned}$$

6 Answering arbitrary range counts

So far, we have discussed building differentially private histograms that minimize either the SSE or the *Sum of Absolute Error* (SAE). These two error metrics are corresponding to the error of unit-length queries. When users query for the sum of consecutive counts, the histogram synopsis may not be optimal in terms of query accuracy. To understand the reason behind the ineffectiveness of histogram synopsis on large range queries, let us revisit the example in Fig. 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 for both NoiseFirst and StructureFirst. Two different strategies are designed for NoiseFirst and StructureFirst due to their different properties.

6.1 Range counting using NoiseFirst

In Algorithm 1, after finding the optimal histogram structure, Mean-NoiseFirst uses the average count to replace the original noisy counts. Another option is to keep using the noisy counts. To ensure a better selection, we compare the estimations on the errors of these two options and adopt the one with a smaller expected SSE error. In particular, for each bin B_i covering the interval $[l_i, r_i]$, Mean-NoiseFirst uses the average count for B_i, i.e., c_i here, only when:

$$\text{SSE}(\hat{D}, l, r) < \frac{4(r-l)}{\epsilon^2} \tag{22}$$

Otherwise, the noisy counts $\{\hat{x}_{l_i}, \ldots, \hat{x}_{r_i}\}$ are kept. The Eq. 22 is derived by combining the conclusions of Eqs. 2 and 4 (with $k = 1$ and $n = r_i - l_i + 1$). 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 smoothing the zero-mean noise added in consecutive 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 situation, the noisy counts may perform better.

Regarding Median-NoiseFirst, the condition that we output the median count for bin B_i is when:

$$\text{SAE}(\hat{D}, l, r) < \frac{4(r-l)+1}{\epsilon} \tag{23}$$

Otherwise, the original noisy counts $\{\hat{x}_{l_i}, \ldots, \hat{x}_{r_i}\}$ are outputted for bin B_i. The Eq. 23 is obtained based on the Eqs. 15 and 24. The Eq. 24 denotes the expected SAE of the noisy counts $\{\hat{x}_{l_i}, \cdots, \hat{x}_{r_i}\}$ with respect to the original counts $\{x_{l_i}, \cdots, x_{r_i}\}$.

$$\begin{aligned}&\mathbb{E}\left\{\{(l_i, l_i, \hat{x}_{l_i}), \cdots, (r_i, r_i, \hat{x}_{r_i})\}, \{x_{l_i}, \cdots, x_{r_i}\}\right\} \\ &\quad = \frac{(r_i - l_i + 1)}{\epsilon}\end{aligned} \tag{24}$$

6.2 Range counting using StructureFirst

Unfortunately, the non-uniform strategy used for NoiseFirst is not applicable for StructureFirst, since StructureFirst calculates the histogram on the original counts.

To apply the non-uniform scheme, we mainly borrow the ideas from [15]. Generally speaking, after constructing all bins, StructureFirst runs the boosting algorithm [15] on each

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 range $[l_i, r_i]$, Laplace noise with magnitude $\frac{2}{\epsilon_2}$ is added on every count x_j for $l_i \leq j \leq r_i$, as well as the sum $s_i = \sum_{l_i \leq j \leq r_i} x_j$. Assume $\overline{x}_j$ is the result noisy count and $\overline{s}_i$ is the noisy sum. Our algorithm runs normalization on $\{\overline{x}_{l_i}, \ldots, \overline{x}_{r_i}, \overline{s}_i\}$ and returns a new group of counts $\{x'_{l_i}, \ldots, x'_{r_i}, s'_i\}$ satisfying that $s'_i = \sum_{l_i \leq j \leq r_i} x'_j$. These counts are used to approximate the original counts in the database. This scheme satisfies the ϵ_2-differential privacy [15]. Therefore, the complete modified algorithm of StructureFirst still fulfills the ϵ-different privacy.

There is some connection between such method and [15]. StructureFirst with this non-uniform count assignment scheme can be considered as an adaptive version of the boosting algorithm in [15]. In the original boosting algorithm, the user must specify the fan-out of the tree 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. By using our histogram construction technique, an adaptive partitioning is used instead, which is supposed to better capture the data distribution. This is the underlying reason that StructureFirst outperforms the algorithm in [15]. In the following section, some empirical studies verify the advantages of our proposals.

7 Experiments

This section experimentally compares the effectiveness of the four proposed methods, 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* [9], *Privelet* [28] and *Boost* [15]. The implementations of NoiseFirst and StructureFirst use the non-uniform scheme discussed in Sect. 6 in order to better answer range count queries of arbitrary range lengths. Meanwhile, the parameter F (i.e., the maximal possible count in a count sequence) is fixed to $\frac{10^4 * \text{NumofRecords}}{\text{Domain Size}}$. The implementation of Boost follows the same settings in [15], which uses a binary tree. Similarly, both versions of StructureFirst also utilize a binary tree inside each bin of the histogram, in order to ensure fair competition with Boost. Akin to [15], we evaluate the accuracy of all algorithms for range count queries with varying lengths and locations. In particular, given a query length L, we test all possible range count queries and report the *Average Squared Error*, the *Average Absolute Error* and the *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 quad-core 3.4 GHz CPU and 4GB RAM, running Windows 7. In each experiment, every method is repeated 20 times, and its average accuracy is reported. Our experiments use four real-world datasets:

- *Age* [28] 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.
- *Search Logs* [15] 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 min interval, from January 1, 2004, to August 9, 2009.
- *NetTrace* [15] contains IP-level network trace data 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 counts ranging from 1 to 1,423.
- *SocialNetwork* [15] records the friendship relations among 11K students from an institution derived from an online social network website. Each record contains the number of friends of a student. There are 32,768 students, each of which has up to 1,678 friends.

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

In the following, we use the term *NoiseFirst* to represent both Mean-NoiseFirst and Median-NoiseFirst, and *StructureFirst* to denote both Mean-StructureFirst and Median-StructureFirst. Meanwhile, *mean-based solutions* consists of Mean-NoiseFirst and Mean-StructureFirst, while *median-based solutions* are Median-NoiseFirst and Median-StructureFirst. In StructureFirst, since the parameter k (i.e., the number of bins in the output histogram) cannot be derived by the algorithm itself, we first test the effect of different k's on the accuracy of StructureFirst. These results provide guidelines on the selection of k for StructureFirst. After that, Sect. 7.2 compares the mean-based solutions with their median-based counterparts. Based on such comparisons, we obtain guidelines for selecting between mean-based/median-based methods. The ones chosen according to these guidelines participate in the comparisons in Sect. 7.3 against existing solutions. Finally, we report the execution time of all proposed methods.

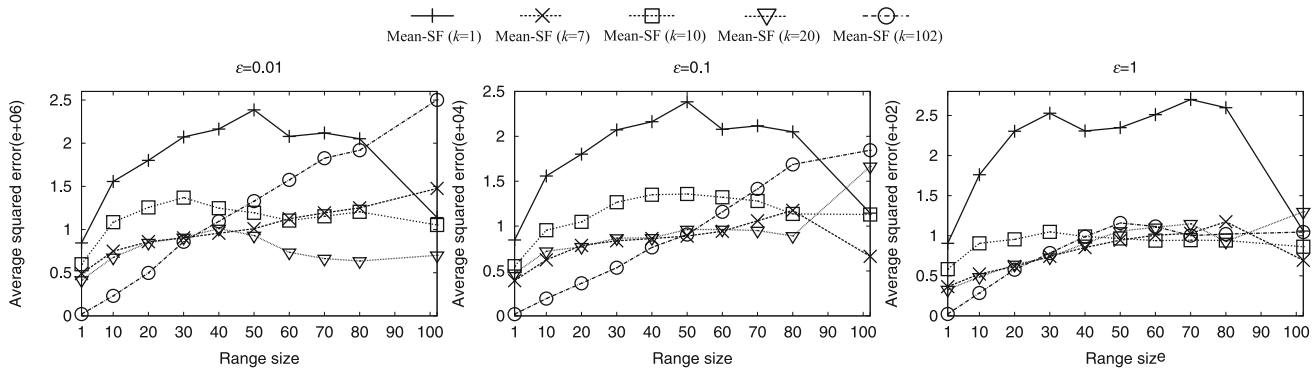


Fig. 6 Varying k on *Age*

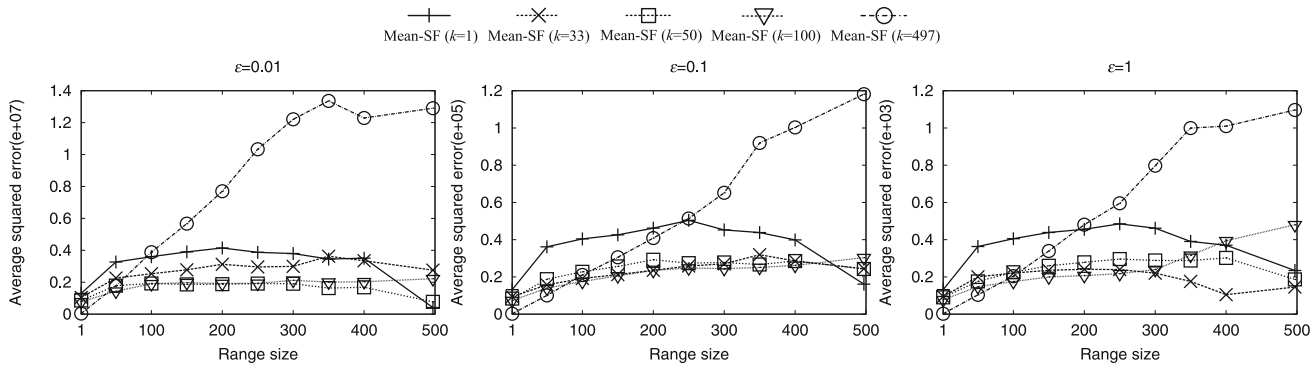


Fig. 7 Varying k on *Search Logs*

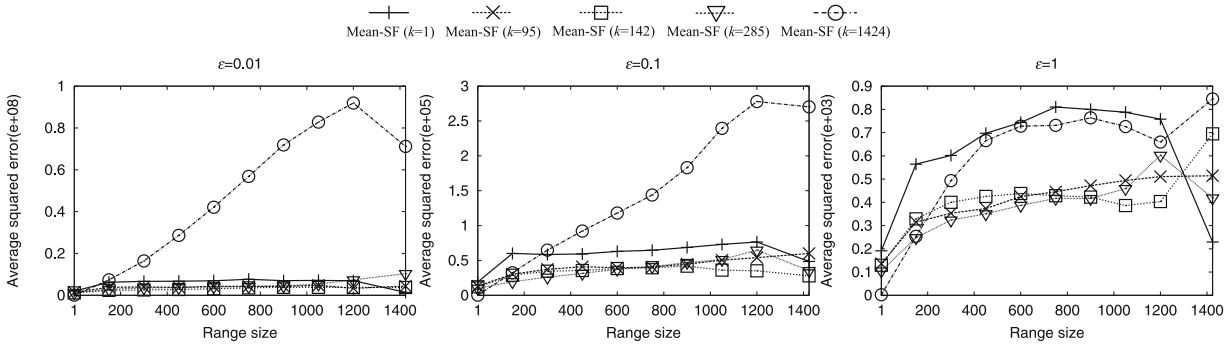


Fig. 8 Varying k on *NetTrace*

7.1 Impact of parameter k on StructureFirst

This subsection evaluates the effectiveness of StructureFirst with different number of bins k in the resulting histogram. Specifically, for each dataset with a domain size n, we test five different ks: 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 6, 7, 8, 9 illustrate the average squared error of Mean-StructureFirst with different ks and different querying range sizes on four datasets. Figures 10, 11, 12, 13 repeat the same experiments for Median-StructureFirst, reporting the absolute error rather than the 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 [9] that adds Laplace noise of magnitude $\frac{1}{\epsilon}$ to every count in the dataset. In this case, the error incurred by StructureFirst increases linearly with the querying 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.

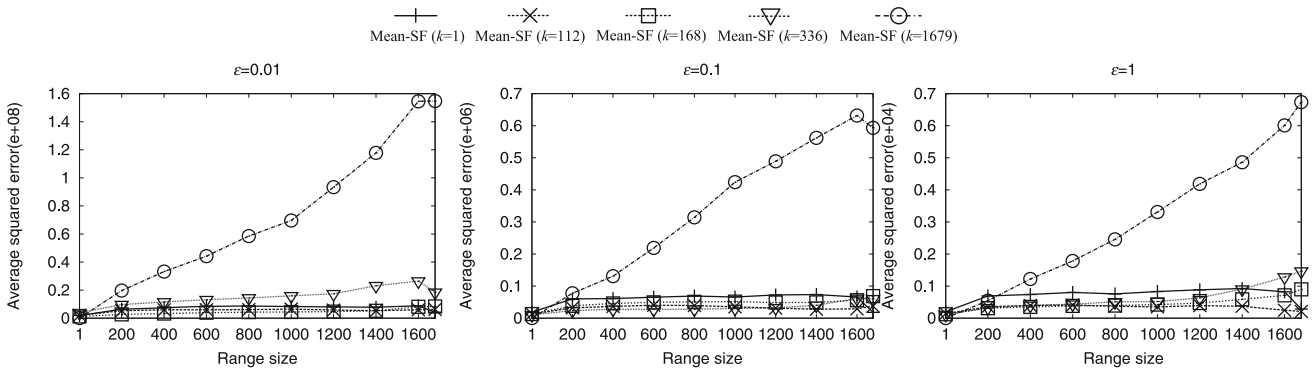


Fig. 9 Varying k on *Social Network*

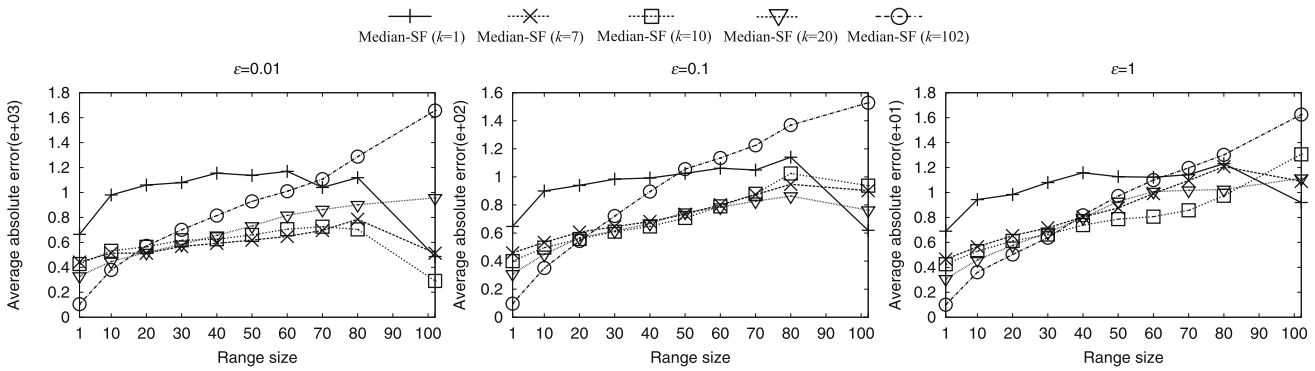


Fig. 10 Varying k on *Age*

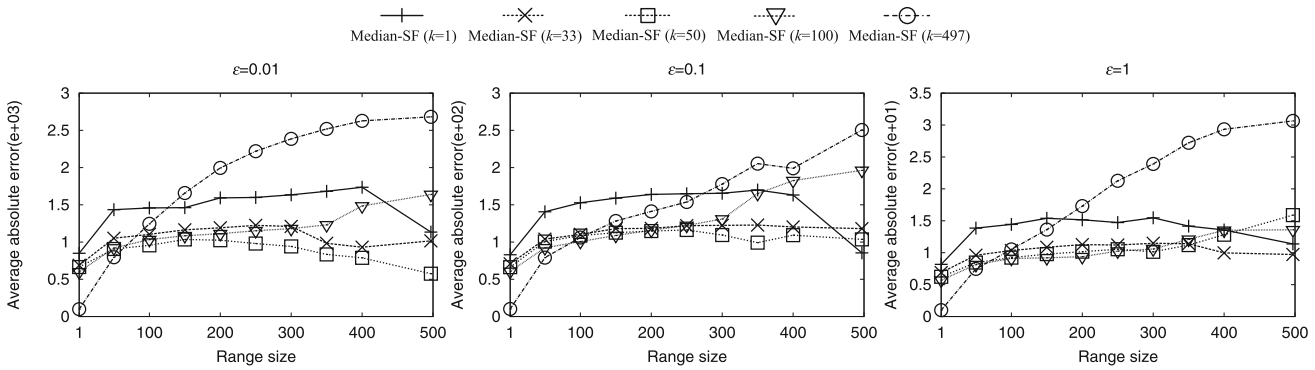


Fig. 11 Varying k on *Search Logs*

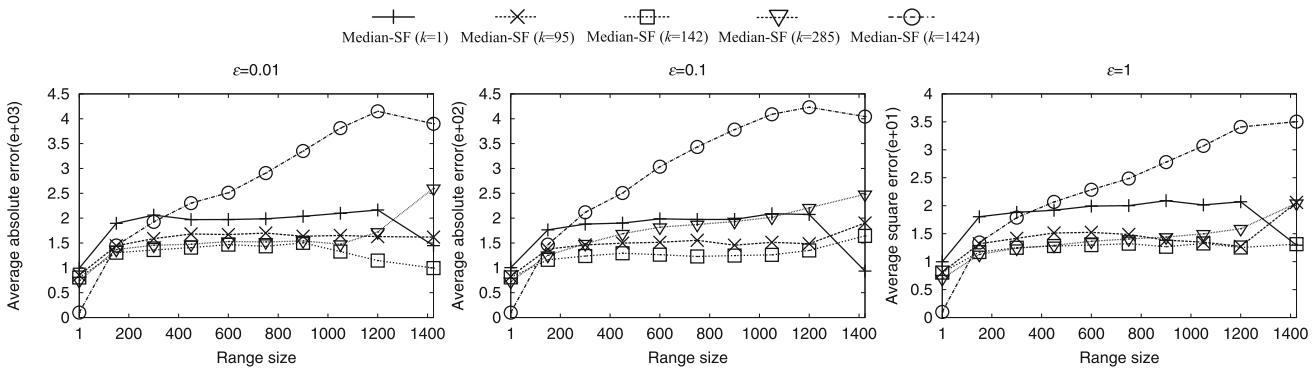


Fig. 12 Varying k on *NetTrace*

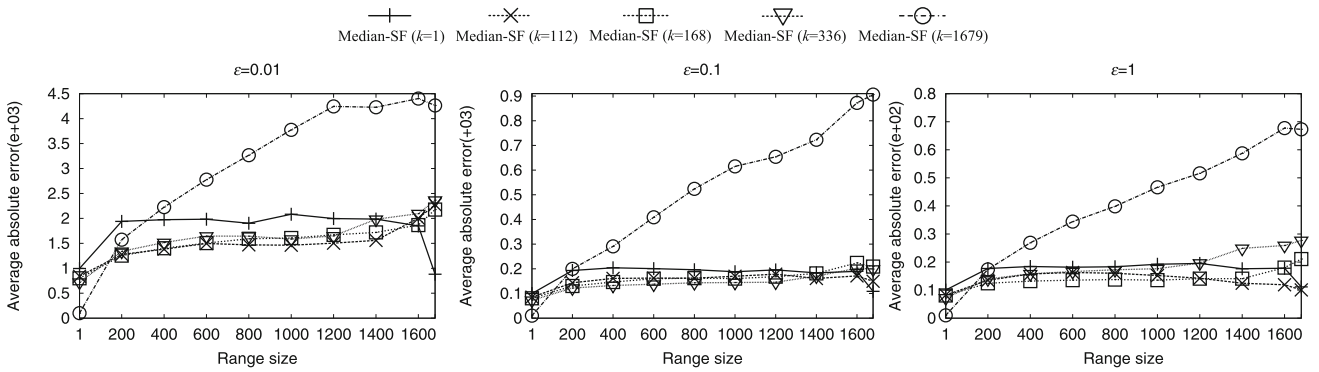


Fig. 13 Varying k on *Social Network*

Second, when $k = 1$, StructureFirst is equivalent to the Boost method [15]. Similar to the case for $k = n$, the histogram structure is fixed, and there is no need to spend privacy budget on it. The performance of StructureFirst follows the properties of Boost, with more accurate results on queries with very small or very large range sizes.

Third, when $1 < k < n$, StructureFirst achieves generally better performance compared to the above extreme cases. 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 [15] 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 make a comparison between the proposed mean-based and median-based solutions. In particular, Mean-NoiseFirst is tested against Median-NoiseFirst (Fig. 14a–d), and Mean-StructureFirst is compared with Median-StructureFirst (Fig. 15a–d). Due to the limited writing space, we uniformly exhibit the tendency changes of SSE for the above-mentioned methods, considering the SSE to some extent reflects the value of SAE. We observe the following from Fig. 14. First, for both Mean-NoiseFirst (Mean-NF) and Median-NoiseFirst (Median-NF), the error increases linearly with the size of the querying range. This is because these two methods spend the whole privacy budget in the noise injection to each bin. Therefore, they incur a noise variance of $O(\frac{m}{\epsilon^2})$, where m is a variable corresponding to the querying 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 larger probability for the algorithm to produce extreme counts in each merged bin. Since the mean statistic is susceptible to those extreme counts, the estimation accuracy of Mean-NoiseFirst is decreased. On the other hand, Median-NoiseFirst performs well because 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 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 too big.

From Fig. 15, we achieve the following conclusion. In most cases, Median-StructureFirst (Median-SF) has higher accuracy than that of Mean-StructureFirst (Mean-SF). 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 $\exp\left\{-\frac{\epsilon_1 E_{\mathrm{SAE}}(q,j,r_{j+1})}{2(k-1)}\right\}$, 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 reduces the randomness of the adjustment and hence produces a more accurate histogram structure. It is the accurate histogram structure that improves the accuracy of Median-SF.

Summarizing the results of this subsection. Mean-NoiseFirst obtains more accurate histograms when ϵ is relatively large (e.g., when $\epsilon = 1$, while Median-NoiseFirst

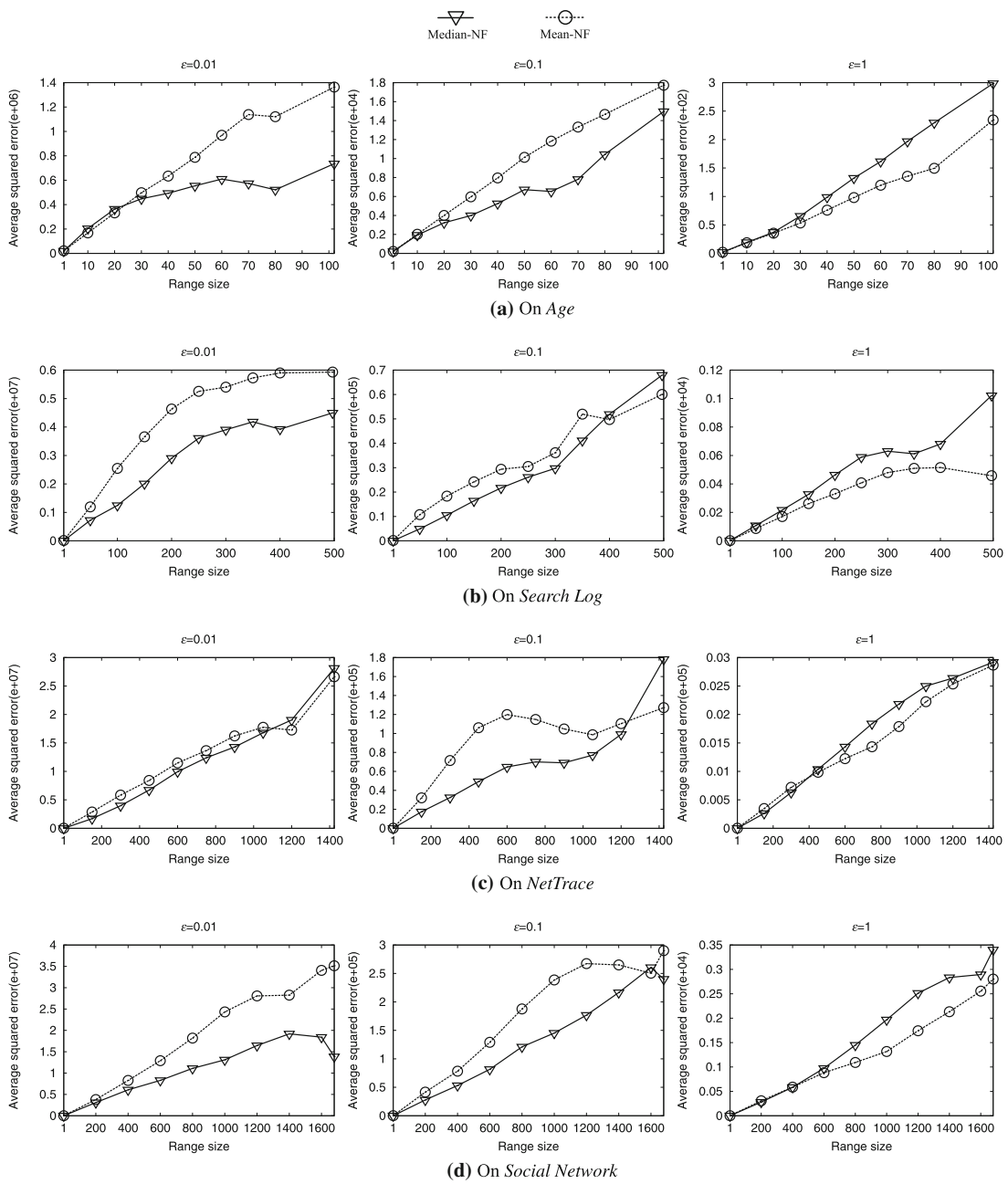


Fig. 14 Average squared error of Mean-NoiseFirst and Median-NoiseFirst

is preferred for smaller values of ϵ. Meanwhile, in most cases, Median-StructureFirst outputs histograms with higher accuracy compared to Mean-StructureFirst, regardless of the value of ϵ. Based on these observations, we suggest choosing Median-NoiseFirst for NoiseFirst when $\epsilon \leq 0.1$, and selecting Mean-NoiseFirst otherwise. Regarding StructureFirst, we suggest using Median-StructureFirst in all settings.

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 each ϵ value are marked

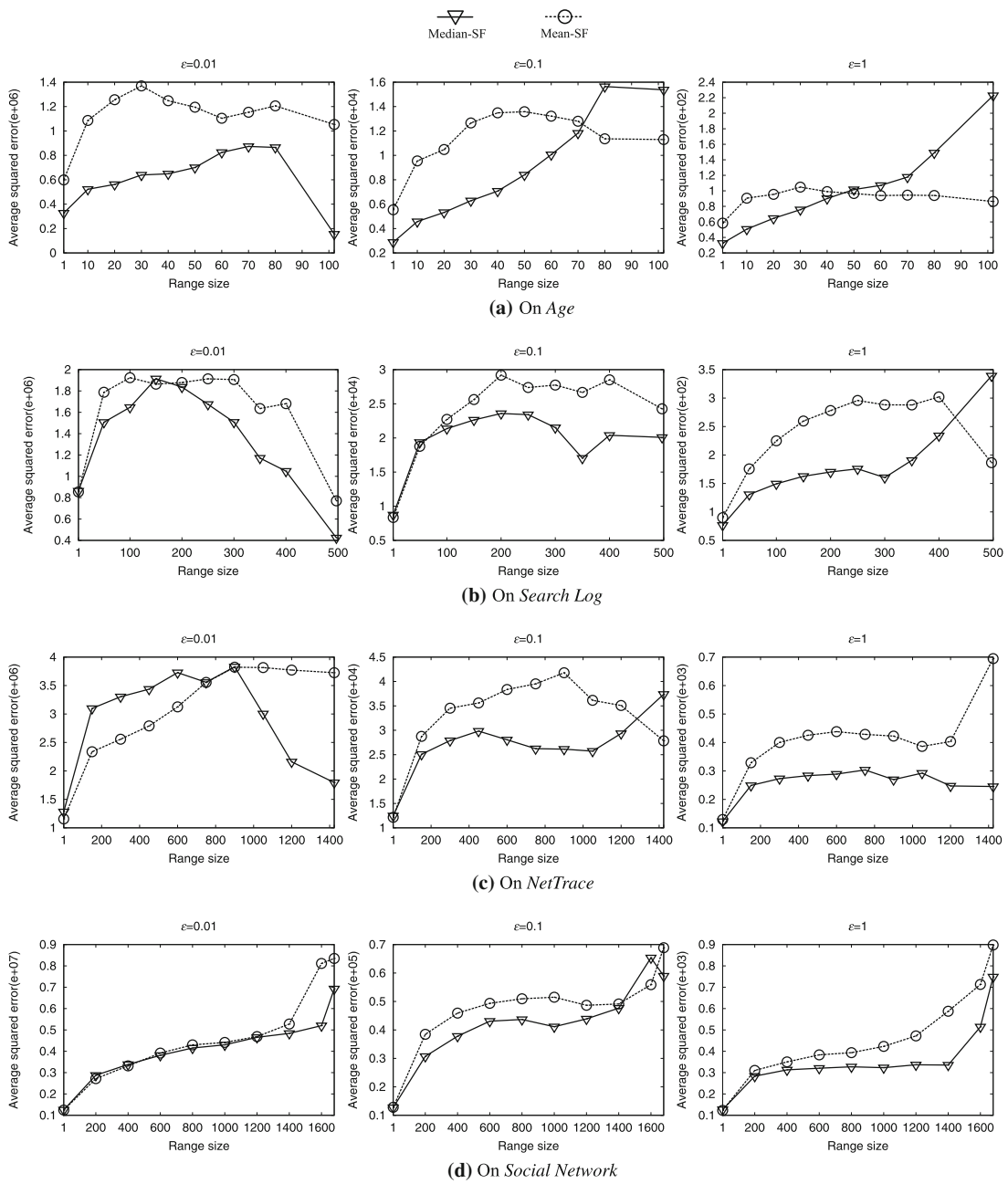


Fig. 15 Average squared error of Median-StructureFirst and Mean-StructureFirst

in bold. We get the following observations from these results.

First, Dwork and NoiseFirst have significant advantages 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, Privelet and StructureFirst) all adopt a hierarchy tree structure to reorganize the original data in order to improve the query accuracy with large range sizes. 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* datasets than Dwork, while the latter sometimes performs better than NoiseFirst on the *Age* dataset. This is because data values in *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 counts. On the other hand, the remaining

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	**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
Privelet	360,774	4,745	42	678,763	6,975	62	978,884	9,987	91	972,103	9,426	92
Median-NF	19,438	203	2.10	**5,081**	**60**	**1.44**	**3,475**	**35**	**0.56**	**4,102**	**50**	**0.76**
Mean-NF	21,178	205	**1.98**	12,231	131	1.48	13,198	95	1.34	12,677	135	1.40
Median-SF	327,525	2, 876	32	863,095	874,2	77	1,283,316	12,465	122	1,252,283	12,725	127
Mean-SF	598,750	5,541	58	854,311	8,353	90	1,156,542	12,154	129	1,245,424	12,819	123

Table 3 Comparison of the average absolute 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
Privelet	453	51	4.90	615	62	5.89	732	74	7.16	730	73	7.21
Median-NF	100	10	0.99	**28**	**3.51**	**0.74**	**21**	**2.06**	**0.31**	**24**	**3.04**	**0.40**
Mean-NF	104	10	**0.96**	62	6.86	0.83	63	6.32	0.67	62	6.50	0.71
Median-SF	427	40	4.25	665	66	6.23	807	81	8.04	800	81	8.08
Mean-SF	535	56	5.59	662	65	6.80	763	79	8.20	798	81	8.08

three datasets have a large number of small unit counts, and NoiseFirst is more effective in merging consecutive small unit counts into bins which eliminates the impact of the random noise in the bins. Especially, on *NetTrace* with $\epsilon = 0.1$, Median-NF is over six 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 which is more robust to those extreme counts. On the other hand, Mean-NF is more easily affected by those extreme values, leading to unstable mean values and consequently has a lower accuracy.

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 and average relative error. Recall that StructureFirst sets $k = \frac{n}{10}$ in all settings, based on the conclusion drawn in Sect. 7.1.

Based on the experimental results shown in Sect. 7.2, we select Median-NF for NoiseFirst (NF) when ϵ equals to 0.1 or 0.01 and choose Mean-NF when ϵ is set as 1. For StructureFirst (SF), Median-SF is used for all settings of ϵ.

For simplicity, regardless whether Mean-NF or Median-NF is used, we denote it as *NF* in the figures. Similarly, Median-SF in the comparison are denoted as *SF*.

In order to clearly display the relative performance of different methods, some points on large range sizes are discarded for Dwork and NoiseFirst in Figs. 16 and 17. This is because these two methods have a linear growth trend with the growth of the querying range sizes. The results shown in Figs. 16 and 17 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 Privelet 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 Privelet is essentially a binary tree). Hence, very short queries favor both methods, as they only need to access a small number of nodes close to the leaf level. Very

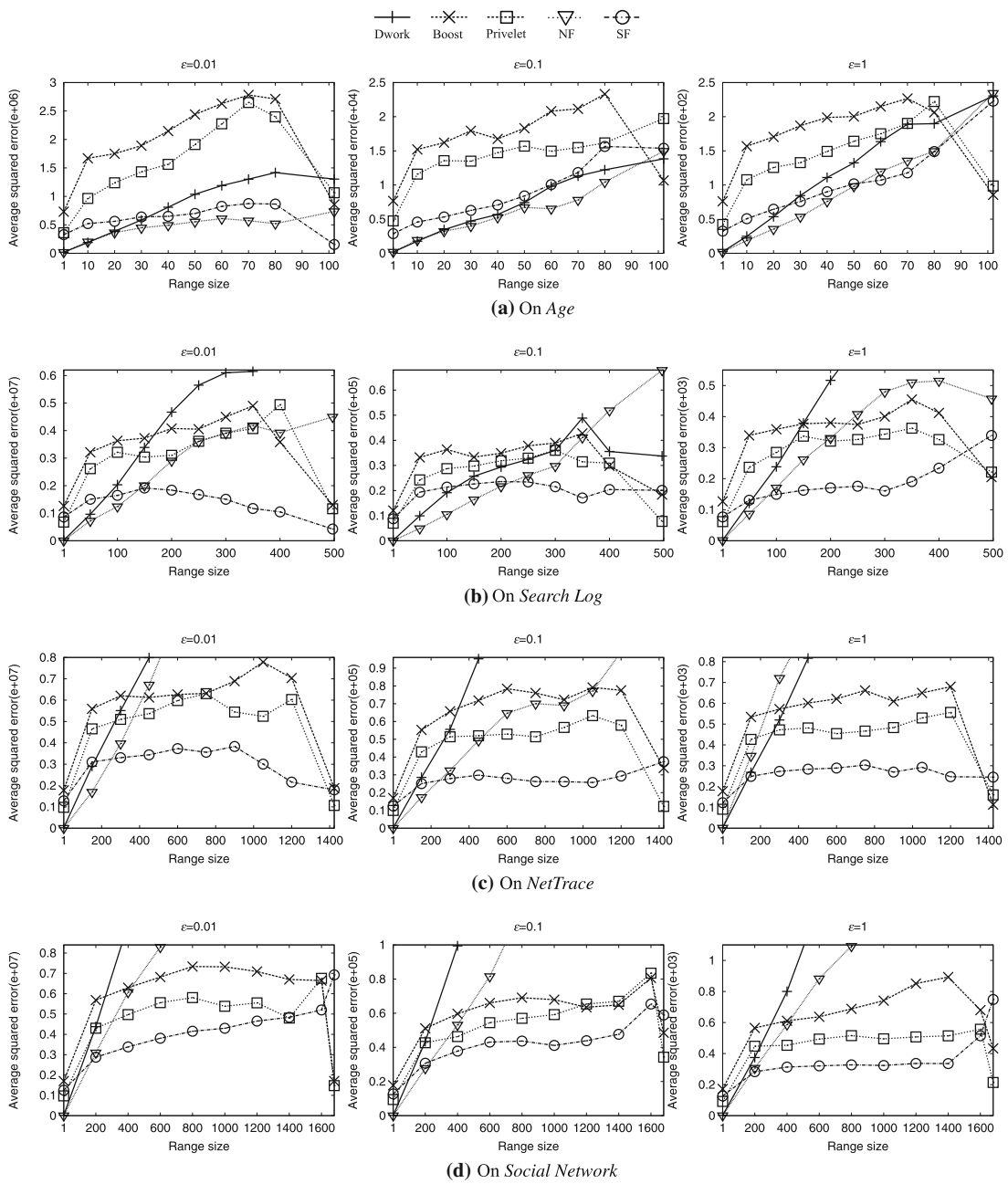


Fig. 16 Average squared error of different methods

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 we propose a query 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.

Third, StructureFirst produces more accurate estimates than both Privelet and Boost on all datasets under most settings of range sizes. Specifically, StructureFirst on average performs twice as well as those two methods. The main reasons are (i) that StructureFirst avoids building a large tree over the whole data domain which reduces the number of nodes required to answer a range query; and (ii) 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 among bins. Another interesting observation concerning StructureFirst is that when the range size is as large as the domain size, Boost and Privelet have better accuracy than StructureFirst. This is because both Boost and Privelet build hierarchical trees to summarize data

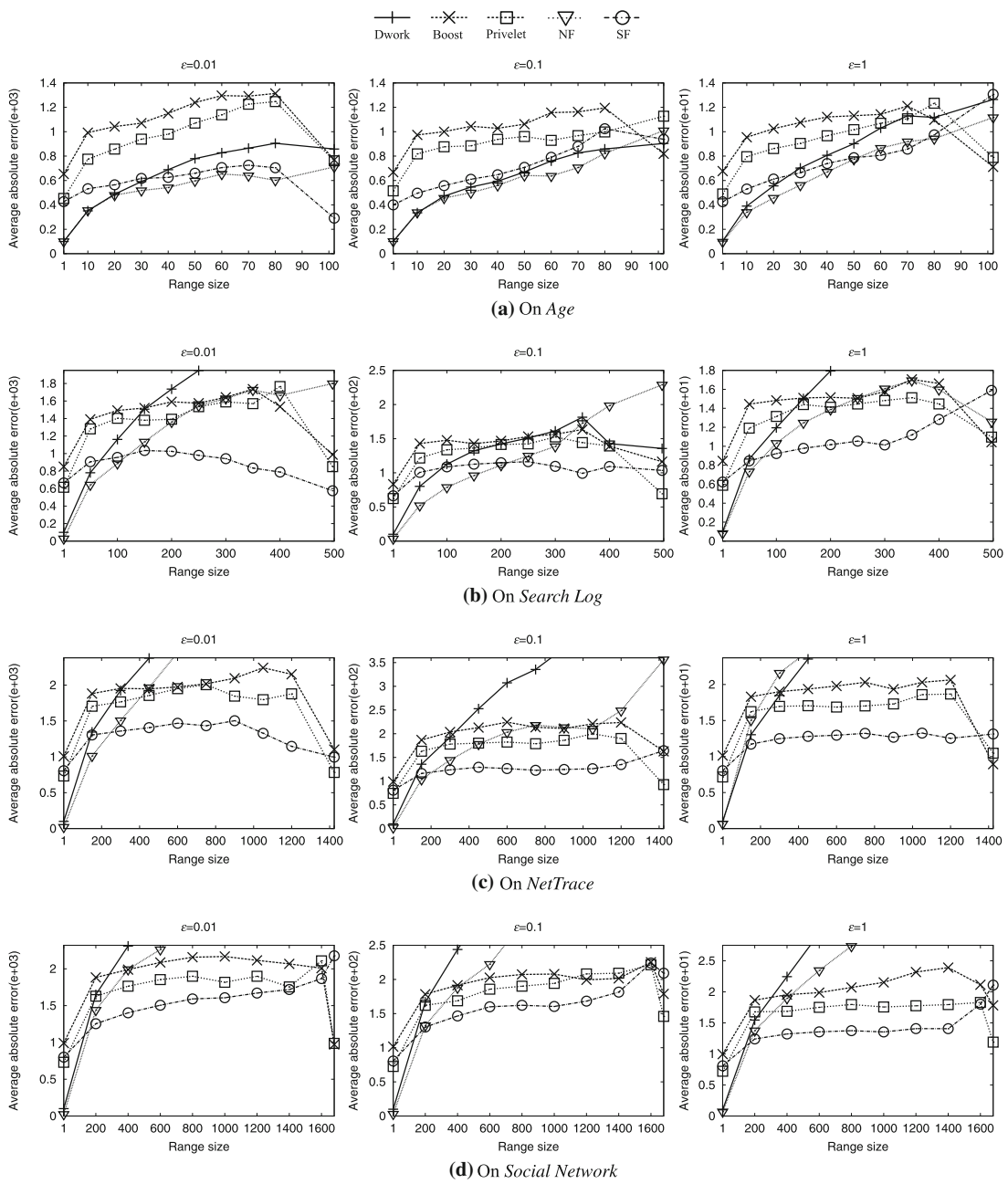


Fig. 17 Average absolute error of different methods

on the whole domain, and hence, they have a better overview on range counts on the complete domain.

Figure 18 exhibits the relative error of all methods on our four datasets. Similar to [28], the relative error of a variable C is computed as $\frac{|C-C_{act}|}{max\{c_{\text{act}},S\}}$, 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, the parameter s is set to 0.1 % of the number of tuples in the dataset. From the figure, we see that all methods tend to have a relatively small error on the *Age* dataset. 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 Fig. 18 is that when the privacy budget ϵ is as small as 0.01, all methods incur relative errors higher than 20 on Search Log, NetTrace and Social Network datasets. This indicates the trade-off between the utility of the query results and the degree of privacy protection.

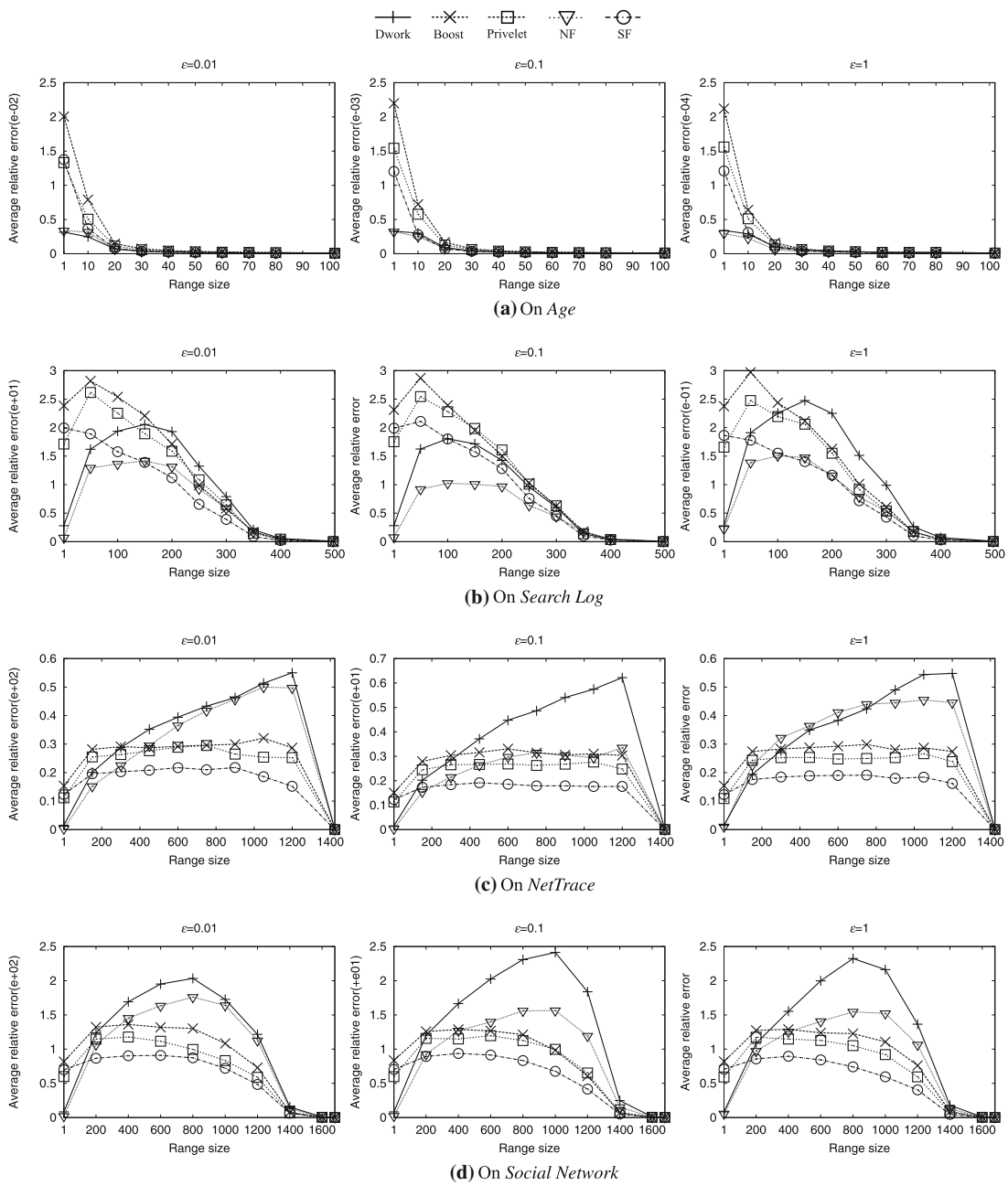


Fig. 18 Average relative error of different methods

7.4 Evaluation on execution time

Both NoiseFirst and StructureFirst have time complexity of $O(n^2k)$, where n is the cardinality of the count sequence, and k represents the number of bins in the constructed histogram. The total running time of these methods is dominated by the cost of the dynamic programming algorithm used for constructing the V-Optimal histogram [18]. To test the scalability of our proposals, we run them on the Social Network dataset, which has the largest n (i.e., 1,678) among all our datasets. Figures 19 and 20 report the execution time of our mean-based and median-based solutions, respectively, with varying values of n, k and ϵ. Note that to ensure fair comparisons, we ignore the recommended value for k computed by NoiseFirst; in particular, in the experiments with varying parameter n and ϵ, we fix k to 100.

Both Figs. 19 and 20 indicate that the execution time of mean-based or median-based solution is not sensitive to the parameter ϵ. The execution time of both NoiseFirst and StructureFirst increases linearly with the number of bins k and grows quadratically with the cardinality of the original count sequence n, which is consistent with the time com-

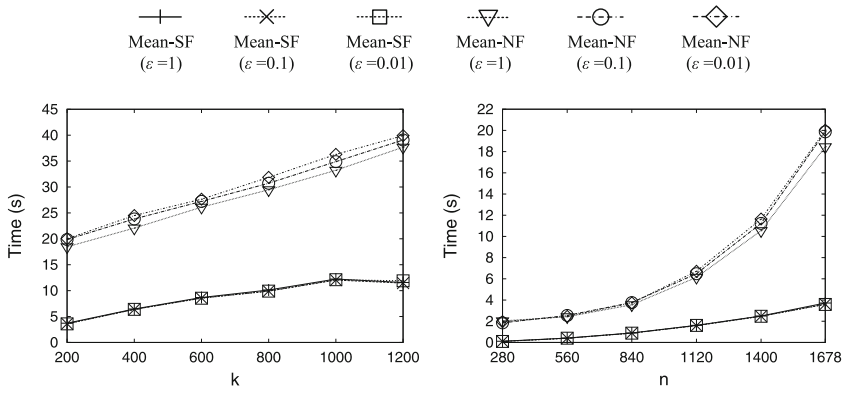


Fig. 19 Execution time for mean-based solutions

(a) Varying k **(b)** Varying n

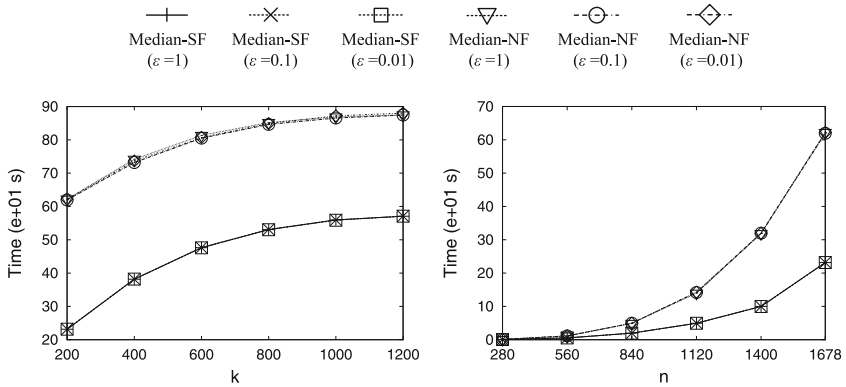


Fig. 20 Execution time for median-based solutions

(a) Varying k **(b)** Varying n

plexity $O(n^2k)$ of computing optimal histograms. StructureFirst runs about 4 times faster than NoiseFirst. This is expected, because NoiseFirst involves an additional module for computing an appropriate value for k, whereas in StructureFirst k is directly provided by users.

7.5 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 in handling queries with larger ranges. Thus, a query executer can provide users more precise results by querying the DP-compliant 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 [8] for a survey). For example, Bhaskar et al. [2] investigate how frequent itemsets from transaction data can be published. Friedman et al. [12] devise methods for constructing decision trees. Korolova et al. [19] and Götz et al. [13] present methods for publishing statistics in search logs, while McSherry and Mahajan [23] develop techniques for network trace analysis. Zhang et al. [31] study the problem of differentially private regression analysis. Finally, Mohan et al. build a general-purpose system [24] for applications where the level of privacy protection changes with time.

Among the existing approaches, the ones most related to ours are by Blum et al. [3], Hay et al. [15], Xiao et al. [28] and Li et al. [21,22]. 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. [15] 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. [28] 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 dataset. 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. [21,22] 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 [30]. 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 datasets.

There are also a lot works handling the multi-dimensional data in a differentially private way [5,6,29]. Cormode et al. [5] 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 when ensure a certain query accuracy, while we target at improving the query accuracy at full stretch. Xiao et al. in [29] present a couple of multi-dimensional 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 solution is seemingly similar to StructureFirst , since we both at first merge consecutive similar counts into different partitions and then consume a certain amount of privacy budget to hide 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 [6], 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. [15] and Xiao et al.'s [28] methods, their problem is different 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. [7] propose methods for releasing *marginals*, i.e., projections of a dataset 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. [27] 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 reduces relative errors. This approach, however, is inapplicable for our problem, since the count queries concerned in a histogram are mutually disjoint.

Rastogi and Nath [25] 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 a database.

By summarizing similar consecutive counts using aggregates (mean/median), our mechanisms dramatically improve the accuracy of range count queries. Experimental results on four real-world datasets 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 state-of-the-art methods on count queries with longer ranges. As for future work, we plan to extend the proposed algorithms to publish multi-dimension histograms under the differential privacy.

Acknowledgments Jia Xu and Ge Yu are supported by the National Basic Research Program of China (973) under Grant 2012CB316201, the National Natural Science Foundation of China (with Nos. 61033007 and 61003058), and the Fundamental Research Funds for the Central Universities (with No. N100704001). Zhenjie Zhang and Yin Yang are supported by SERC Grant No. 102 158 0074 from Singapore's A*STAR. Xiaokui Xiao is supported by Nanyang Technological University under SUG Grant M58020016 and AcRF Tier 1 Grant RG 35/09, and by the Agency for Science, Technology and Research (Singapore) under SERG Grant 1021580074

References

1. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., Talwar, K.: Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In: PODS, pp. 273–282 (2007)
2. Bhaskar, R., Laxman, S., Smith, A., Thakurta, A.: Discovering frequent patterns in sensitive data. In: KDD, pp. 503–512 (2010)
3. Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: STOC, pp. 609–618 (2008)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms, 2nd edn., pp. 185–192. MIT Press and McGraw-Hill, New York (2001)
5. Cormode, G., Procopiuc, C.M., Srivastava, D., Tran, T.T.L.: Differentially private publication of sparse data. In: ICDT (2012)
6. Cormode, G., Procopiuc, M., Shen, E., Srivastava, D., Yu, T.: Differentially private spatial decompositions. In: ICDE (2012)
7. Ding, B., Winslett, M., Han, J., Li, Z.: Differentially private data cubes: optimizing noise sources and consistency. In: SIGMOD, pp. 217–228 (2011)
8. Dwork, C.: Differential privacy: a survey of results. In: TAMC, pp. 1–19 (2008)
9. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: TCC, pp. 265–284 (2006)
10. Dwork, C., McSherry, F., Talwar, K.: The price of privacy and the limits of LP decoding. In: STOC, pp. 85–94 (2007)
11. Dwork, C., Rothblum, G.N., Vadhan, S.P.: Boosting and differential privacy. In: FOCS, pp. 51–60 (2010)
12. Friedman, A., Schuster, A.: Data mining with differential privacy. In: KDD, pp. 493–502 (2010)
13. Götz, M., Machanavajjhala, A., Wang, G., Xiao, X., Gehrke, J.: Publishing search logs—a comparative study of privacy guarantees. IEEE TKDE 24(3): 520–532 (2012)
14. Guha, S., Koudas, N., Shim, K.: Approximation and streaming algorithms for histogram construction problems. ACM TODS **31**(1), 396–438 (2006)
15. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially private histograms through consistency. PVLDB **3**(1), 1021–1032 (2010)
16. Homer N., Szelinger S., Redman M., Duggan D., Tembe W., Muehling J., Pearson J.V., Stephan D.A., Nelson S.F., Craig, D.W.: Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. PLoS Genet. **4**(8), e100167 (2008)
17. Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K.C., Suel, T.: Optimal histograms with quality guarantees. In: VLDB, pp. 275–286 (1998)
18. Jagadish H.V., Koudas N., Muthukrishnan S., Poosala V., Sevcik K.C., Suel T. Optimal histograms with quality guarantees. In: VLDB, pp. 275–286 (1998)
19. Korolova, A., Kenthapadi, K., Mishra, N., Ntoulas, A.: Releasing search queries and clicks privately. In: WWW, pp. 171–180 (2009)
20. Kotz, S., Kozubowski, T., Podgórski, K.: The Laplace Distribution and Generalizations: A Revisit with Applications to Communications, Economics, Engineering, and Finance. Birkhäuser Publication, Boston (2001)
21. Li, C., Hay, M., Rastogi, V., Miklau, G., McGregor, A.: Optimizing linear counting queries under differential privacy. In: PODS, pp. 123–134 (2010)
22. Li, C., Miklau, G.: An adaptive mechanism for accurate query answering under differential privacy. PVLDB **5**(6), 514–525 (2012)
23. McSherry, F., Mahajan R. Differentially-private network trace analysis. In: SIGCOMM, pp. 123–134 (2010)
24. Mohan, P., Thakurta, A., Shi, E., Song, D., Culler, D.E.: Gupt: privacy preserving data analysis made easy. In: SIGMOD, pp. 349–360 (2012)
25. Rastogi V., Nath S.: Differentially private aggregation of distributed time-series with transformation and encryption. In: SIGMOD, pp. 735–746 (2010)
26. Wang, R., Li, Y., Wang, X., Tang, H., Zhou, X.: Learning your identity and disease from research papers: Information leaks in genome wide association study. In: ACM CCS (2009)
27. Xiao, X., Bender, G., Hay, M., Gehrke, J.: ireduct: differential privacy with reduced relative errors. In: SIGMOD, pp. 229–240 (2011)
28. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. In: ICDE, pp. 225–236 (2010)
29. Xiao, Y., Xiong, L., Yuan, C.: Differentially private data release through multidimensional partitioning. In: Secure Data Management, pp. 150–168 (2010)
30. Yuan, G., Zhang, Z., Winslett, M., Xiao, X., Yang, Y., Hao, Z.: Low-rank mechanism: optimizing batch queries under differential privacy. PVLDB **5**(11), 1352–1363 (2012)
31. Zhang, J., Zhang, Z., Xiao, X., Yang, Y., Winslett, M.: Functional mechanism: regression analysis under differential privacy. PVLDB **5**(11), 1364–1375 (2012)