Differentially private data release and analytics

By

Haoran Li
Doctor of Philosophy

Computer Science and Informatics

Differentially private data release and analytics

By

Haoran Li
B.S. Hefei University of Technology, China, 2009

Advisor: Li Xiong

An abstract of
A dissertation submitted to the Faculty of the James T. Laney School
of Graduate Studies of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics

2017

ABSTRACT

Differentially private data release and analytics

By Haoran Li

Nowadays data sharing is important for application domains, such as scientific discoveries, business strategies, commercial interests, and social goods, especially when there are not enough local samples to test a hypothesis. However, data in its raw format are sensitive as they essentially contain individual specific information, and publishing such data without proper protection may disclose personal privacy. Netflix canceled their recommendation system contest because the released customers data can identify special individuals with high probability. In order to promote data sharing, it is important to develop privacy-preserving algorithms that respect data confidentiality while present data utility. In this dissertation, we address the privacy concerns in publishing high-dimensional static data and dynamic datasets, and developing mechanisms for personalized differential privacy where data subjects can have various privacy preferences. Our privacy preserving algorithms satisfy differential privacy, a rigorous and de facto standard for privacy protection. Extensive empirical studies demonstrate the effectiveness of our solutions and confirm that our methods have great promise for privacy-preserving data release and analytical tasks in a wide range of application domains.

Differentially private data release and analytics

By

Haoran Li
B.S. Hefei University of Technology, China, 2009

Advisor: Li Xiong

A dissertation submitted to the Faculty of the James T. Laney School
of Graduate Studies of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics

2017

Acknowledgement

First and foremost, I would like to thank my advisor, Prof. Li Xiong, for all of her help and encouragement throughout my Ph.D. When I first began to do my research, little did I know about what it means to do research. Li, with her endless patience and faithful encouragement, guided me through each step to do meaningful works. I can never forget all the moments when Li stays up late to the last minute together with me before conference deadlines, challenges me to push every idea to its limit and iterates over paper and talks numerous times to reach their excellence. In addition, I really appreciate the freedom Li gives me to explore the research domains I am interested in, with which I really enjoyed the past years of research experience. Li's patience about research and commitment to students make her the best academic advisor I could ask for. I learned so much from her, not only about how to do excellent research but also how to be a good mentor with care and patience.

I would like to thank Xiaoqian Jiang for his insights and advice for my research. I would also like to show my sincere thankfulness to other members of my thesis committees Vaidy Sunderam and Joyce Ho for their time and willingness to serve my thesis defense while providing valuable feedback to my research and pushing me to think about it in another level.

Finally I would like to thank my friends and families, who gave me tremendous help for me to complete my Ph.D. I would like to thank my parents, for always encouraging me, believing the best in me and always be there when I need their help. I would also like to thank my boyfriend Lewen Yang for being a constant source of support and encouragement. This thesis would have never been possible without him.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1. Motivation

With the advent of big data era, huge amounts of individual data have been largely collected and analyzed in various application domains, healthcare research, public transit agencies, retailing business, and social networking. Utilizing personal information can undoubtably accelerate scientific discoveries and provide business insights. However, such individual raw data often contains personal identified sensitive information. The risk of disclosing distinguished individual data may be promoted by inappropriate data sharing among different parties and continuous collection of individual data. This can be exemplified by a number of real-life privacy disclosure incidents listed below.

AOL search queries. In the 2006 AOL data publication, 20 million search queries from 650,000 users were released. User identity information (e.g., such as name or SSN) had been anonymized by some pseudonymous user IDs before release [**51**]. But three days later, the release has to be removed because the search logs of a specific user were re-identified by a newspaper journalist.

Netflix prize data. Netflix, the largest on-demand Internet streaming media service provider, released the anonymized movie ratings of 500,000 subscribers for a contest with the purpose of improving the accuracy of its recommendation system. However, Netflix had to cancel the contest because Narayanan and Shmatikov [94] revealed that when combining with the information in the public Internet Movie Database (IMDb), users in the released Netflix dataset could be re-identified with high probability.

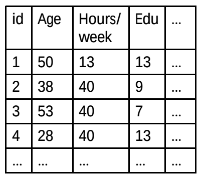
Genome-wide association study. In genome-wide association studies (GWAS) of healthcare research, Lin et al. estimated that 75 - 100 single nucleotide polymorphisms or fewer than 20 micro-satellite markers can unambiguously identify a single individual. Most recently, Gymrek et al. identified personal genomes by surname inference from the "anonymized" 1000 genome data.

These real-world privacy concerns have stimulated strong demands for privacy protection in data sharing. In recent years, privacy preserving data analysis and publishing [1] has received considerable attention as a promising approach for sharing information while preserving data privacy. Differential privacy [2] has recently emerged as one of the strongest privacy guarantees for statistical data release. A statistical aggregation or computation is *DP*[1] if the outcome is formally indistinguishable when run with and without any particular record in the dataset. The level of indistinguishability is quantified by a privacy budget ϵ. A common mechanism to achieve differential privacy is the Laplace mechanism [3] which injects calibrated noise to a statistical measure determined by the privacy budget ϵ, and the sensitivity of the statistical measure influenced by the inclusion and exclusion of a record in the dataset. A lower privacy parameter requires larger noise to be added and provides a higher level of privacy.

Many mechanisms (e.g. [2, 7], etc) have been proposed for achieving differential privacy for a single computation or a given analytical task and programming platforms have been implemented for supporting interactive differentially private queries or data analysis [6]. Due to the composition theorem of differential privacy [6], given an overall privacy budget constraint, it has to be allocated to subroutines in the computation or each query in a query sequence to ensure the overall privacy. After the budget is exhausted, the database can not be used for further queries or computations. This is especially challenging in the scenario where multiple users need to pose a large number of queries for exploratory analysis. Several works started addressing effective

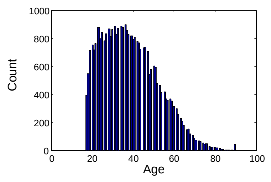
[1]we shorten differentially private as DP

query answering in the interactive setting with differential privacy given a query workload or batch queries by considering the correlations between queries or query history.



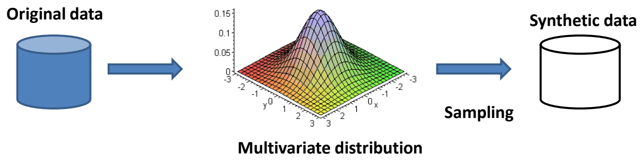
id	Age	Hours/ week	Edu	...
1	50	13	13	...
2	38	40	9	...
3	53	40	7	...
4	28	40	13	...
...	...	...	...	...

(a) Dataset

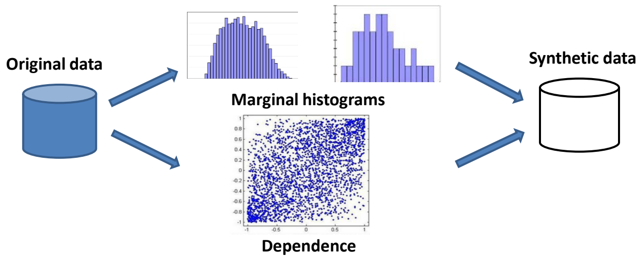


(b) Marginal Histogram for Age

Figure 1.1. Dataset vs. histogram illustration



(a) Generate synthetic data via multivariate distribution



(b) Generate synthetic data via Copula

Figure 1.2. Synthetic data generation

A growing number of works started addressing non-interactive data release and aggregate analysis with differential privacy (e.g. [52, 41, 42, 43, 44, 45]). Given an original dataset, the goal is to publish a DP statistical summary such as marginal or multi-dimensional histograms that can be used to answer predicate queries or to generate DP synthetic data that mimic the original data. For example, Figure 1.1 shows an example dataset and a one-dimensional marginal histogram for the attribute age. The main approaches of existing work can be illustrated by Figure 1.2(a) and classified into two categories: 1) parametric methods that fit the original data to a multivariate distribution and makes inferences about the parameters of the distribution (e.g. [52]). 2) non-parametric methods that learn empirical distributions from the data through histograms (e.g. [42, 43, 44, 45]). Most of these work well for single dimensional or low-order data, but become problematic for data with high dimensions and large attribute domains. This is due to the facts that:

- The underlying distribution of the data may be unknown in many cases or different from the assumed distribution, especially for data with arbitrary margins and high dimensions, leading the synthetic data generated by the parametric methods not useful;

- The high dimensions and large attribute domains result in a large number of histogram bins that may have skewed distributions or extremely low counts, leading to significant perturbation or estimation errors in the non-parametric histogram methods;

- The large domain space $\prod_{i=1}^{m} |A_i|$ [2] (i.e. the number of histogram bins) incurs a high computation complexity both in time and space. For DP histogram methods that use the original histogram as inputs, it is infeasible to read all histogram bins into memory simultaneously due to memory constraints, and external algorithms need to be considered.

To tackle these technical challenges, in this thesis, we propose a novel differentially private data synthesization method, DPCopula [17], for static high dimensional and large domain data using copula functions. It reduces large amount of injected noise for privacy protection by modeling private marginal histogram for each dimension and joint dependence for all dimensions, separately.

Although a series of algorithms aim to release synthetic data under differential privacy, most of them focus on "one-time" release of a static dataset and do not adequately address the increasing need of releasing series of dynamic datasets in

[2]We define $\prod_{i=1}^{m} |A_i|$ as the domain space of all dimensions, where $|A_i|$ is the domain size of the ith attribute and m is the number of attributes

real time. Sharing series of private data evolving overtime under differential privacy enables many important data mining and knowledge discovery applications.

Medical research. A hospital gathers data from individual patients every day. The dynamic datasets, e.g. the daily datasets of individual patients with fevers, coughs, and different demographic attributes can be shared with researchers for cohort discovery, medical research, and seasonal epidemic outbreak monitoring.

Traffic Monitoring. A GPS service provider gathers data from individual users about their locations, speeds, mobility, etc. The dynamic datasets, e.g., the numbers of users at different regions during each time period, can be mined for commercial interest, such as congestion patterns on the roads.

A common scenario of such applications is that a trusted server gathers data from a large number of individual subscribers. The aggregated data can be then continuously shared with other untrusted entities for various purposes. The trusted server, i.e. publisher, therefore must ensure that releasing the data does not compromise the privacy of any individual who contributed data. A straightforward application of existing histogram methods on each snapshot of such dynamic datasets will incur high accumulated error due to the composition theorem [6] and correlations or overlapping users between the snapshots. In this thesis, we address the problem of releasing series of dynamic datasets in real time with differential privacy, using a novel adaptive distance-based sampling approach.

Differential privacy provides only a uniform level of privacy protection for all users (record owners) in a dataset. It assumes every data subject have the same privacy preference for their personal data. However, this "one size fits all" privacy setting ignores the reality that data privacy is a personal and multifaceted concept, and that different individuals may have very different expectations for the privacy of their personal data. In practice, when faced with a dataset including multiple users or data owners with different privacy settings, a data analyst employing differential privacy has limited options. The privacy setting where record owners in a dataset could set their own privacy preferences is considered as “Personalized Differential privacy”, shortened as PDP.

One possibility for achieving PDP is to uses the minimal privacy budget among all records, called minimum mechanism. This is likely to introduce an unacceptable amount of noise into the analysis outputs, resulting in poor utility. Especially when we have a dataset with skewed privacy preferences where only few data owners set their privacy preferences to be the minimum, a large amount of privacy budgets would be wasted while the utility will be significantly reduced. Another possibility is that we can set a privacy budget threshold and select a subset whose privacy budgets are no less than the threshold, then apply differentially private applications using the chosen subset. However, the threshold is difficult to choose because there is a tradeoff between number of records in the subset and the threshold. Higher privacy budget threshold means less number of records, and vice versa. In this dissertation, we develop two novel partitioning mechanisms for achieving personalized differential privacy (PDP): privacy-aware partitioning and utility-based partitioning. The main goal is to design mechanisms that can take full advantage of the different privacy budgets to obtain better utility than could be achieved with the common differential privacy mechanism.

1.2. Research Contributions

In this dissertation, we address the following research problems:

- How to sample high-dimensional and large data from original data while guaranteeing high utility under differential privacy?
- How to release series of dynamic datasets in real time under differential privacy?
- How to handle personalized differential privacy when record owners have different privacy preferences?

1.2.1. Privacy-preserving high-dimensional data release (Chapter 3). In this chapter, we propose DPCopula, a differentially private data synthesization technique using Copula functions for multi-dimensional data. The core of our method is to compute a differentially private copula function from which we can sample synthetic data. Copula functions are used to describe the dependence between multivariate random vectors and allow us to build the multivariate joint distribution using one-dimensional marginal distributions. Most of previous methods work well for single dimensional or low-order data, but become problematic for data with high dimensions and large attribute domains. This is mainly because the high dimensions and large attribute domains result in a large number of histogram bins that may have skewed distributions or extremely low counts, leading to significant perturbation or estimation errors in the non-parametric histogram methods.

Different from previous state-of-the-arts methods, our framework computes a DP copula function and samples synthetic data from the function that effectively captures the dependence implicit in the high-dimensional datasets. With the copula functions, we can separately consider the margins and the joint dependence structure of the original data instead of modeling the joint distribution of all dimensions. Furthermore, DPCopula allows direct sampling for the synthetic data from the margins and the copula function. For most existing techniques, post-processing is required to enforce non-negative histogram counts or consistencies between counts which results in either degraded accuracy or high computation complexity.

We present two methods, DPCopula-MLE (we shorten maximum likelihood estimation as MLE in the paper) and DPCopula-Kendall, for estimating parameters of the Gaussian copula function, a commonly used elliptical class of copula functions modeling the Gaussian dependence. We focus on semi-parametric Gaussian copula as most real-world high-dimensional data has been shown to follow the Gaussian dependence structure. It can be used not only to model data with Gaussian joint distributions, but also data with arbitrary marginal distributions or joint distributions as long as they follow Gaussian dependence. DPCopula-MLE computes correlation among dimensions using DP MLE while DPCopula-Kendall computes DP correlation among dimensions using Kendall's τ correlation which is a general nonlinear rank-based correlation. Extensive experiments using both real datasets and synthetic datasets demonstrate that DPCopula generates highly accurate synthetic multi-dimensional data with significantly better utility than state-of-the-art techniques.

1.2.2. Privacy-preserving dynamic histogram release (Chapter 4). In this chapter, we investigate the problem of releasing histograms for dynamic datasets while guaranteeing user-level differential privacy, i.e., protecting the presence of a user in the entire series of dynamic datasets. In the worst case, a user may be present in all datasets in the series. A straight-forward application of the standard differential privacy mechanism or existing histogram method to each snapshot of the dataset will lead to a very high perturbation error $O(N)$ in the order of the number of datasets or snapshots N in the series. The goal of this chapter is to releases a differentially private histogram only when the current snapshot is sufficiently different from the previous one, thus reducing the amount of injected noise for differential privacy perturbation when releasing dynamic datasets overtime.

Our first method, DSFT, uses a fixed distance threshold and releases a differentially private histogram only when the current snapshot is sufficiently different from the previous one, i.e., with a distance greater than a predefined threshold. Our second method, DSAT, further improves DSFT and uses a dynamic threshold adaptively adjusted by a feedback control mechanism to capture the data dynamics. Extensive experiments on real and synthetic datasets demonstrate that our approach achieves better utility than baseline methods and existing state-of-the-art methods.

1.2.3. Personalized differential privacy (Chapter 5). This chapter focuses on personalized privacy setting, in which record owners in a dataset can have different expectations regarding the acceptable level of privacy for their data. One possibility for achieving PDP is to uses the minimal privacy budget among all records, but this is likely to introduce an unacceptable amount of noise into the analysis outputs, resulting in poor utility. Another possibility is that we can set a privacy budget threshold and select a subset whose privacy budgets are no less than the threshold, then apply differentially private applications using the chosen subset. However, the threshold is difficult to choose because there is a tradeoff between number of records in the subset and the threshold. Higher privacy budget threshold means less number of records, and vice versa. The goal of this chapter is to design mechanisms that can take full advantage of different privacy budgets to obtain better utility than could be achieved with the common differential privacy mechanism.

We develop two novel partitioning mechanisms for achieving personalized differential privacy (PDP): privacy-aware partitioning mechanism and utility-based partitioning mechanism. The privacy-aware partitioning mechanism is to consider all different privacy budgets as a histogram and group histogram bins with similar privacy budgets, such that the amount of wasted budget is minimized. We use sum of squared errors between the *minimum* of each partition and all privacy budgets of this partition to measure the waste of privacy budgets in the current partition. The utility-based partitioning mechanism is to partition the privacy budget histogram in order to maximize the utility. In particular, we find that the utility-based mechanism has superior performance for many important differentially private applications, such as count queries, logistic regression and support vector machine. The reason is that it considers both the minimum privacy budget and the number of records in the current partition, which are significantly related to the utility of goal differentially private mechanisms. We conducted extensive experimental studies of our partitioning mechanisms and compare our methods with the previous sampling mechanism in [25]. We first investigated both of our mechanisms for the important *count* query, then studied the application of our mechanisms to more complex tasks, like logistic regression and support vector machine. Our results demonstrate that the general applicability and superior performance of our methods.

CHAPTER 2

Related Works

This chapter briefly reviews the recent, relevant literatures on differentially private synthetic data generation, and personalized differential privacy.

2.1. Differentially private synthetic data generation

Various approaches have been proposed recently for publishing differentially private histograms (e.g. [40, 49, 42, 43, 44, 45, 46, 47, 48], etc). Among them, the methods of [42] and [43] are designed for single dimensional histograms. The technique of [48] is proposed especially for two dimensional data. Here we focus on discussing the methods for multi-dimensional histograms below.

The method by Dwork et al. [3] publishes a DP histogram by adding independent Laplace random noise to the count of each histogram bin. The Dwork method is more favorable for small-domain attributes and can only guarantee reasonable accuracy for queries about individual entries in the frequency matrix. For queries involving many entries, the Dwork method fails to provide useful results. While the method works well for low-dimensional data, it becomes problematic for high dimensional and large domain data. Barak et al. [40] uses Dwork's method to obtain a DP frequency matrix, then transforms it to the Fourier domain and adds Laplace noise in this domain. With the noisy Fourier coefficients, it employs linear programming to create a non-negative frequency matrix. The problem is that (i) post-processing is not shown to improve accuracy and (ii) it requires solving a linear program where the number of variables equals to the number of entries in the frequency matrix. This can be computationally challenging for practical data sets with large domain space. For instance, for the eight-dimensional data sets each with domain size of 1000, the number of variables is $\prod_{i=1}^{m} |A_i|=10^{24}$.

Xiao et al. [41] propose a Privelet+ method by applying a wavelet transform on the original histogram, then adding polylogarithmic noise to the transformed data. Cormode et al. [45] developed a series of filtering and sampling techniques to obtain a compact summary of DP histograms. The limitation is that if a large number of small-count non-zero entries exists in the histogram, it will give zero entries a higher probability to be in the final summary, leading to less accurate summary results. In addition, it needs carefully choosing appropriate values for several parameters including sample size and filter threshold. The paper did not provide a principled approach to determine them. Both the DPCube [49] and PSD [44] are based on KD-Tree partitioning. DPCube first uses Dwork's method to generate a DP cell histogram and then applies partitioning on the noisy cell histogram to create the final DP histogram. PSD computes KD-tree partitioning using DP medians at each step. It has been shown in [44] that these two methods are comparable. However, for high-dimensional and large attribute domain data, either the level of partitioning will be high which results in high perturbation error or the distribution of each partition will be skewed which results in high estimation error. Acs et al. [47] study two sanitization algorithms for generating DP histograms. The EFPA technique improves the fourier perturbation scheme through tighter utility analysis while P-HP is based on a hierarchical partitioning algorithm. But there are limitations for high dimension data. When the number of bins in original histograms is extremely large, for EFPA, the parameter representing the histogram shape would be selected with high error; for P-HP, the accuracy of each partitioning step would have large perturbation error and the computation complexity would be proportional to the quadratic number of bins in the worst case.

The DiffGen method [46] releases differentially private generalized data especially for classification by adding uncertainty in the generalization procedure. Only predictor attributes are generalized for maximizing the class homogeneity within each partition. It classifies the attributes into class attributes and predictor attributes. The limitation is that it is mainly designed for classification with only the predictor attributes being generalized due to the class attribute. For high-dimensional and large-domain data, the method has similar issues as the KD-partitioning methods. What's more, the number of specializations need to be predefined heuristically.

2.2. Differentially private dynamic data generation

Differentially private stream data release. Several mechanisms (e.g. [4, 34], etc) focus on event-level privacy in releasing counters, i.e. in publishing the number of event occurrences at every time point since the commencement of the system. These mechanisms consider the data stream as a bit string and at each time point they release the number of 1's seen so far. A set of related works have studied the problem of releasing aggregate time series and stream statistics. The works in [4, 34] proposed differentially private continual counters over a binary stream. However, both works adopt an event-level differential privacy, which protects the presence of an individual event, i.e. a user's contribution to the data stream at a single time point, rather than her presence or contribution to the entire series. The works in [35, 36, 37] studied the problem of releasing aggregate time-series with user-level differential privacy. Both works consider temporal correlations of the time-series. The paper [35] uses a Discrete Fourier Transform approach and is not applicable to real-time applications when data needs to be released at each time point. Other works [36, 37] take a model based approach which assumes original data is generated by an underlying process and uses the model based prediction to improve the accuracy of the released data. The limitation is that the model needs to be assumed or learned from public data with similar patterns and the method may not be effective when the real data deviates from the model. [38] releases dynamic transaction data under user-level privacy, and set an upper bound to limit the maximum number of updates to handle infinite updates. But it can only handle insertions updates.

Differentially private dynamic dataset release. The most recent work that is closely related to our work is Kellaris et al. [39] which deals with differentially private release of events or histograms for infinite stream. It proposes a w-event privacy framework by combining user-level and event-level privacy, which protects any event sequence occurring within any window of w timestamps. It is event-privacy with w = 1 and converges to user- level privacy with w = infinity. They also proposed two mechanisms, Budget Distribution (BD) and Budget Absorption (BA), to allocate the budget within one w-timestamp window. The key difference between our work and [39] is that our methods detect the data dynamics and adaptively adjust the distance threshold for sampling such that the privacy budget is not depleted prematurely due to high update and sampling rates or insufficiently utilized due to low update and sampling rates. In [39], privacy budgets may be depleted prematurely, especially when w is very large, or not fully utilized during the w timestamps. In addition, our method is independent of the histogram method used for each time point and can utilize any state-of-the-art histogram methods designed for static data release as a blackbox. In our experiments, we compare our methods with BD and BA in [39], since they represent the state-of-the-art and have been shown to perform better than other existing work.

2.3. Personalized differential privacy

Personalized privacy. Tao and Xiao [16] introduced personalized privacy for k-anonymity, which requires every record in a dataset to be indistinguishable from at least $k-1$ records, based on the identifier attributes. The k-anonymity for personalized privacy in [16] allows each data record owner to specify the minimum k they prefer. A series of related literatures (e.g., [21, 20, 29]) explore advanced methods to guarantee k-anonymity and related privacy notations. However, these privacy frameworks cannot protect sensitive attributes information when attackers have sufficient background knowledge ([15, 27, 28]). A more robust privacy definition, differential privacy [2], has recently emerged and are now preferred as one of the strongest privacy guarantees for statistical data release.

Personalized differential privacy. Alaggan et al. [22] developed a privacy definition, called *heterogeneous differential privacy*, which to our knowledge is the first work to consider various privacy preferences of data record owners under non-uniform differential privacy. Their work proposed a "stretching" mechanism, which is based on Laplace mechanism by rescaling the input values due to corresponding privacy parameters. The weakness is that it cannot be applied to many commonly used functions, such as *median*, *min/max*, and counting queries which count the number of non-zero values in a dataset. Ebadi et al. [23] developed a query system called ProPer based on McSherry's PINQ system [6] to maintain a privacy budget for each individual. It allows better utility than models purely based on the global sensitivity. This method is an interactive model, which spends privacy budgets on the fly (i.e., protecting information through a controlled interface handled by the data owner) and is different from our method that publishes data in an non-interactive manner (i.e., releasing once for all data which we think is of interest to most analysis while still preserving privacy [24]).

Jorgensen et al. [25] proposed two mechanisms for achieving personalized differential privacy (PDP). The first mechanism is a two-step sampling based method that involves a non-uniform sampling step at the individual record level, followed by the invocation of an appropriate differentially private mechanism on the sampled dataset. In the sampling step, the inclusion probabilities for each record are computed due to the individual privacy budget of the corresponding user and the privacy budget threshold. The sampling mechanism is general and can be used to easily convert any existing differentially private algorithm into a PDP algorithm, but an appropriate sampling threshold needs to be selected and predefined. The second mechanism is developed based on the exponential mechanism, where the utility function is designed to satisfy PDP particularly. The mechanism is applicable to only common aggregates such as counts, medians, because the utility function is difficult to develop for complicated applications, like linear regression, support vector machine.

Personalized local privacy. Chen et al. [64] proposed a personalized local differential privacy (PLDP) model and developed an efficient personalized count estimation protocol as a building block for achieving PLDP. In PLDP, each user's data needs to be perturbed before leaving his or her own devices, which means no one is trusted except users themselves. Authors in [64] also designed a framework under PLDP that allows an untrusted server to accurately learn the user distribution over a spatial domain while each user can specify his or her own privacy preference. Wang et al. [26] designed another aggregation scheme that combines locally published data at different differential-privacy levels, which can be kept as a secret to the data owners. Different from personalized differential privacy, these methods [64, 26] are developed under local differential privacy, so we will not make direct comparison with them in this article.

CHAPTER 3

Differentially private synthesization of multi-dimensional data using copula functions

In this chapter, we investigate the problem of high-dimensional synthetic data publication with differential privacy and propose DPCopula, a privacy preserving synthetic data generation technique using Copula functions to release differentially private multi-dimensional data.

3.1. Preliminaries

Consider an original dataset D that contains a data vector $(X_1, X_2, \ldots, X_m)$ with m attributes. Our goal is to release differentially private synthetic data of D. For ease of reference, we summarize all frequently used notations in Table 3.1. Their definitions will be introduced as appropriate in the following (sub)sections.

Table 3.1. Frequently used notations

Notation	Discription
D	original dataset
$\widetilde{D}$	DP synthetic data
n	number of tuples in D
m	number of dimensions in D
$(X_1, \ldots, X_m)$	m-dimensional vector of D
$H(x_1, \ldots, x_m)$	m-dimensional joint distribution
$\hat{F}_j(x_j)$	empirical distribution of jth margin
$\tilde{F}_j(x_j)$	DP empirical distribution of jth margin
$\rho_\tau(X_j, X_k)$	Kendall's τ coefficient
$\hat{\rho}_\tau(X_j, X_k)$	sample estimate of Kendall's τ
$\tilde{\rho}_\tau(X_j, X_k)$	private estimate of Kendall's τ
$\rho(X_j, X_k)$	the general correlation
ϵ_1	privacy budget for margins
ϵ_2	privacy budget for all correlations
Δ	sensitivity of Kendall's τ
k	the ratio of ϵ_1 and ϵ_2
$\widetilde{P}$	DP correlation matrix
$(\widetilde{U}_1, \ldots, \widetilde{U}_m)$	DP pseudo-copula data vector

3.1.1. Differential Privacy. Differential privacy has emerged as one of the strongest privacy definitions for statistical data release. It guarantees that if an adversary knows complete information of all the tuples in D except one, the output of a differentially private randomized algorithm should not give the adversary too much additional information about the remaining tuples. We say datasets D and D' differing in only one tuple if we can obtain D' by removing or adding only one tuple from D. A formal definition of differential privacy is given as follows:

Definition 3.1.1 (ϵ-differential privacy [3])**.** Let $\mathcal{A}$ be a randomized algorithm over two datasets D and D' differing in only one tuple, and let $\mathcal{O}$ be any arbitrary set of possible outputs of $\mathcal{A}$. Algorithm $\mathcal{A}$ satisfies ϵ-differential privacy if and only if the following holds:

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

Intuitively, differential privacy ensures that the released output distribution of $\mathcal{A}$ remains nearly the same whether or not an individual tuple is in the dataset.

The most common mechanism to achieve differential privacy is the Laplace mechanism [3] that adds a small amount of independent noise to the output of a numeric function f to fulfill ϵ-differential privacy of releasing f, where the noise is drawn from *Laplace distribution* with a probability density function $Pr[\eta = x] = \frac{1}{2b} e^{-\frac{|x|}{b}}$. A Laplace noise has a variance $2b^2$ with a magnitude of b. The magnitude b of the noise depends on the concept of *sensitivity* which is defined as follows.

Definition 3.1.2 (Sensitivity [3])**.** Let f denote a numeric function and the sensitivity of f is defined as the maximal L_1-norm distance between the outputs of f over the two datasets D and D' which differs in only one tuple. Formally,

$$\Delta_f = max_{D,D'} ||f(D) - f(D')||_1.$$

With the concept of sensitivity, the noise follows a zero-mean Laplace distribution with the magnitude $b = \frac{\Delta_f}{\epsilon}$. To fulfill ϵ-differential privacy for a numeric function f over D, it is sufficient to publish $f(D) + X$, where X is drawn from $Lap(\frac{\Delta_f}{\epsilon})$.

For a sequence of differentially private mechanisms, the composability theorems guarantee the overall privacy.

Theorem 3.1.1 (Sequential Composition [6]). *For a sequence of n mechanisms $M_1, \ldots, M_n$ and each M_i provides ϵ_i-differential privacy, the sequence of M_i provides $(\sum_{i=1}^{n} \epsilon_i)$-differential privacy.*

Theorem 3.1.2 (Parallel Composition [6]). *If D_i are disjoint subsets of the original database and M_i provides α-differential privacy for each D_i, then the sequence of M_i provides α-differential privacy.*

3.1.2. Synthetic data generation under differential privacy. Given an original dataset, the goal is to publish a DP statistical summary such as marginal or multi-dimensional histograms that can be used to answer predicate queries or to generate DP synthetic data that mimic the original data. For example, Figure 1.1 shows an example dataset and a one-dimensional marginal histogram for the attribute age. The main approaches of existing works can be illustrated by Figure 1.2(a) and classified into two categories

- Parametric methods that fit the original data to a multivariate distribution and makes inferences about the parameters of the distribution (e.g. [52])
- Non-parametric methods that learn empirical distributions from the data through histograms (e.g. [42, 43, 44, 45]).

Most of these work well for single dimensional or low-order data, but become problematic for data with high dimensions and large attribute domains. This is due to the facts that:

- The underlying distribution of the data may be unknown in many cases or different from the assumed distribution, especially for data with arbitrary margins and high dimensions, leading the synthetic data generated by the parametric methods not useful;

- The high dimensions and large attribute domains result in a large number of histogram bins that may have skewed distributions or extremely low counts, leading to significant perturbation or estimation errors in the non-parametric histogram methods;

- The large domain space $\prod_{i=1}^{m} |A_i|$ [1] (i.e. the number of histogram bins) incurs a high computation complexity both in time and space. For DP histogram methods that use the original histogram as inputs, it is infeasible to read all histogram bins into memory simultaneously due to memory constraints, and external algorithms need to be considered.

3.1.3. The Copula function. Consider a random vector $(X_1, \ldots, X_m)$ with the continuous marginal cumulative distribution function (CDF) of each component being $F_i(x) = P(X_i \leq x)$, the random vector $(U_1, \ldots, U_m) = (F_1(X_1), \ldots, F_m(X_m))$ has uniform margins after applying the probability integral transform to each component. Then the copula function can be defined below:

Definition 3.1.3 (Copula and Sklar's theorem [9])**.** The m-dimensional copula $C : [0,1]^m \to [0,1]$ of a random vector $(X_1, \ldots, X_m)$ is defined as the joint cumulative distribution function (CDF) of $(U_1, \ldots, U_m)$ on the unit cube $[0,1]^m$ with uniform margins:

$$C(u_1, \ldots, u_m) = P(U_1 \leq u_1, \ldots, U_m \leq u_m)$$

where each $U_i = F_i(X_i)$. Sklar's theorem states that there exists an m-dimensional copula C on $[0,1]^m$ with $F(x_1, \ldots, x_m) = C(F_1(x_1), \ldots, F_m(x_m))$ for all x in $\bar{\mathbb{R}}^m$. If $F_1, \ldots, F_m$ are all continuous, then C is unique. Conversely, if C is an m-dimensional

[1]We define $\prod_{i=1}^{m} |A_i|$ as the domain space of all dimensions, where $|A_i|$ is the domain size of the ith attribute and m is the number of attributes

copula and $F_1, \ldots, F_m$ are distribution functions, then $C(u_1, \ldots, u_m) = F(F_1^{-1}(u_1), \ldots, F_m^{-1}(u_m))$, where F_i^{-1} is the inverse of marginal CDF F_i.

From definition 3.1.3, copula refers to co-behaviors of uniform random variables only; since any continuous distribution can be transformed to the uniform case via its CDF, this is the appeal of the copula functions: they describe the dependence without any concern of the marginal distributions. Here, *dependence* is a general term for any change in the distribution of one variable conditional on another while *correlation* is a specific measure of linear dependence [10] (e.g. Pearson correlation). Two distributions with the same correlations may have different dependencies. We use the rank correlation in our method and will discuss it later in this section. Although the data should be continuous to guarantee the continuity of margins, discrete data in a large domain can still be considered as approximately continuous as their cumulative density functions do not have jumps, which ensures the continuity of margins. We will discuss later how to handle small-domain attributes.

To study the accuracy of the copula-derived synthetic data, we introduce a convergence analysis on copulas, showing that the copula-derived synthetic data is arbitrarily close to the original data when the data cardinality is sufficiently large. Assume we have an original data D_0 with n records, $\{F_{10}, \ldots, F_{m0}\}$ being the original marginal distributions, C_0 be the original copula function (i.e. the original dependence), H_0 be the original joint distribution of the D_0. We also have t synthetic data $D_1, \ldots, D_t$ with $\{F_{1t}\}, \ldots, \{F_{mt}\}$ be a sequence of m one-dimensional marginal distributions and $\{C_t\}$ be a sequence of copulas. Each $\{F_{1i}\}, \ldots, \{F_{mi}\}$ and C_i correspond to D_i, $i \in \{1, \ldots, t\}$ and are parameterized by the number of records of D_i. We have the following theorem:

Theorem 3.1.3 (Convergence of Copulas). For every t in N^+, a m-dimensional joint distribution function H_t is defined as $H_t(x_1, \ldots, x_m) := C_t(F_{1t}(x_1), \ldots, F_{mt}(x_m))$. Then the sequence $\{H_t\}$ converges to H_0 in distribution, if and only if $\{F_{1t}\}, \ldots, \{F_{mt}\}$

converge to $F_{10}, \ldots, F_{m0}$ respectively in distribution, and if the sequence of copulas $\{C_t\}$ converges to C_0 pointwise in $[0,1]^m$.

The Gaussian copula and Gaussian dependence. Although copula has several families, the elliptical class is the most commonly used, including Gaussian copula and t copula. In this paper, we focus on the semi-parametric Gaussian copula as it has better convergence properties for multi-dimensional data and most real-world high-dimensional data follow the Gaussian dependence structure [9] that can be modeled by the Gaussian copula. We note that Gaussian copula is not to be confused with Gaussian distributions. The Gaussian copula can be used not only to model data with Gaussian joint distributions, but also data with arbitrary marginal distributions or joint distributions as long as they follow Gaussian dependence. For other types of data with special dependence structures, such as tail dependence, we can apply the t copula, the empirical copula and other copulas. Actually we can use many approaches to test the goodness-of-fit, such as Akaike's Information Criterion (AIC) to identify the best copula. We leave designing DP t copula and other copulas and testing the goodness-of-fit for the best copula as our future work. Formally, we give the Gaussian copula and Gaussian dependence definitions as follows:

Definition 3.1.4 (The Gaussian Copula [9])**.** Deducing via Sklar's theorem, a multivariate Gaussian density can be written as the product of two components: the Gaussian dependence and margins, denoted as

$$\Phi_{\mathbf{P}}(\mathbf{x}) = \underbrace{\frac{1}{|\mathbf{P}|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}\phi^{-1}(u)^T(\mathbf{P}^{-1}-\mathbf{I})\phi^{-1}(u)\right\}}_{Gaussian\quad dependence} \underbrace{\prod_{i=1}^{m}\frac{\varphi(\phi^{-1}(u_i))}{\sigma_i}}_{Margins}$$

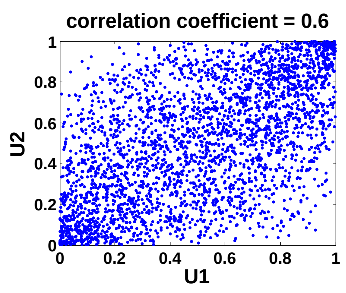
where $\mathbf{P}$ is a correlation matrix[2], $\mathbf{I}$ is the identity matrix, ϕ^{-1} is the inverse CDF of a univariate standard Gaussian distribution, $\phi^{-1}(u) = (\phi^{-1}(u_1), \ldots, \phi^{-1}(u_m))$, $u_i = F_i(x_i)$, $F_i(x_i)$ is Gaussian CDF with the standard deviation σ_i and φ is the standard Gaussian density, $\Phi_{\mathbf{P}}$ denotes the multivariate Gaussian density. If we allow

[2]Here $\mathbf{P}$ must be a semi-positive definite matrix to ensure that $\mathbf{P}^{-1}$ exists

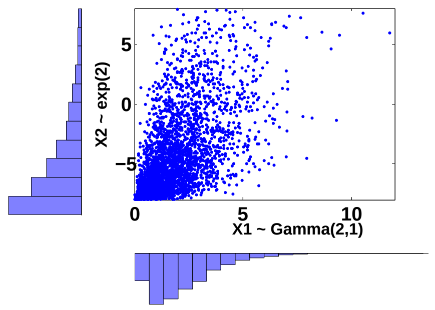
$F_i(x_i)$ to be an arbitrary distribution function, we can obtain the density of Gaussian copula which is the Gaussian dependence part, denoted as $c_{\mathbf{P}}^{Ga}$, with the form

$$c_{\mathbf{P}}^{Ga} = |\mathbf{P}|^{-\frac{1}{2}} exp \left\{ -\frac{1}{2} \phi^{-1}(u)^T (\mathbf{P}^{-1} - \mathbf{I}) \phi^{-1}(u) \right\} \tag{3.1.5}$$

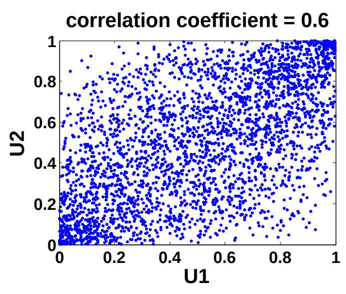
From definition 3.1.4, the density function of Gaussian copula in equation (3.1.5) has no $1/\sqrt{(2\pi)^m}$ compared to that of Gaussian distribution because Gaussian copula allows arbitrary margins. The Gaussian copula does not necessarily have anything to do with Gaussian distributions that require all the margins to be Gaussian distributions. Rather, it represents the Gaussian dependence that arises from a random vector $(U_1, \ldots, U_m)$ with uniform margins. Each component of $(U_1, \ldots, U_m)$ may correspond to an arbitrary distribution $F_i(X_i)$ before the probability integral transform is applied.



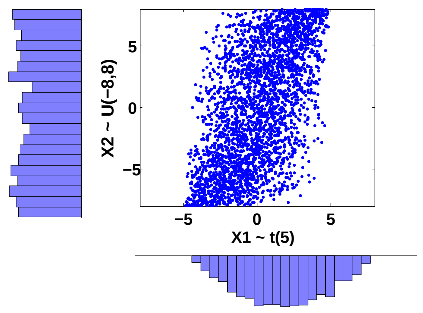
(a) Gaussian copula with Gamma and Exp margins



(b) Bivariate distribution with Gamma and Exp margins



(c) Gaussian copula with Uniform and t margins



(d) Bivariate distribution with Uniform and t margins

FIGURE 3.1. Gaussian copula vs. multivariate distribution

Figure 3.1 illustrates two bivariate Gaussian copula examples (i.e. two uniform random variables on $[0, 1]$ with a Gaussian dependence structure) with the same correlation but different margins and the corresponding joint distribution. The same principle can be extended to more than two random variables. Figure 3.1(a) and (b) illustrates a scatter plot of a bivariate Gaussian copula with the exponential and gamma margins and a corresponding bivariate joint distribution with the attributes on the original domains. The scatter plots in Figure 3.1(c) and (d) is a bivariate Gaussian copula with uniform and t margins and its corresponding joint distribution. We can see that the joint distributions may be different due to different margins but the Gaussian copula scatter plots (i.e. Gaussian dependence) are the same with the same correlation. In other words, the dependence of data can be modeled independently from the margins. While real-world high dimensional data may have different marginal or joint distributions, most data follow the Gaussian dependence which can be modeled by Gaussian copula with different correlations.

Estimation of the Gaussian copula. Since there are unknown parameters which are margins and $\mathbf{P}$ in the copula function, they can be estimated based on input data. The steps of estimation are as follows. First, the data is transformed to *pseudo-copula data* on $[0, 1]^m$ by the non-parametric estimation method to estimate the marginal CDF. Assume $\mathbf{X}_j = (X_{1,j}, \ldots, X_{n,j})^T$is the jth data vector of $(\mathbf{X}_1, \ldots, \mathbf{X}_m)$, the empirical marginal CDF F_j on the jth dimension can be estimated by

$$\hat{F}_j = \frac{1}{n+1} \sum_{i=1}^{n} 1_{\left\{X_{i,j} \geq x\right\}} \tag{3.1.6}$$

where $\hat{F}_j$ is the empirical distribution function of $\mathbf{X}_j$. Here $n+1$ is used for division to keep $\hat{F}_j$ lower than 1. Then, we can generate the jth-dimension pseudo-copula data by

$$\hat{\mathbf{U}}_j = (\hat{F}_j(X_{1,j}), \ldots, \hat{F}_j(X_{n,j}))^T \tag{3.1.7}$$

Once we get the pseudo-copula data, there are two methods to estimate the correlation matrix $\mathbf{P}$. The first method is directly using maximum likelihood estimation with the pseudo-copula data as input [11], named as MLE in our paper. However, maximizing the log likelihood function is specially difficult in multi-dimensions. For this reason, estimation based on dependence measure is of practical interest.

The second method is to estimate the correlation matrix $\mathbf{P}$ based on Kendall's τ correlation coefficients between dimensions. From the original data vectors, we can estimate $\rho_\tau(\mathbf{X}_j, \mathbf{X}_k)$ by calculating the standard sample Kendall's τ coefficient $\hat{\rho_\tau}(\mathbf{X}_j, \mathbf{X}_k)$ (see Section 3.2.3). Due to [11], the estimator of the general correlation coefficient, $\rho(\mathbf{X}_j, \mathbf{X}_k)$, is given by

$$\rho(\mathbf{X}_j, \mathbf{X}_k) = sin(\frac{\pi}{2}\hat{\rho_\tau}(\mathbf{X}_j, \mathbf{X}_k)) \tag{3.1.8}$$

In order to estimate the entire correlation matrix $\mathbf{P}$, we need to obtain all pairwise estimates in an empirical Kendall' τ matrix $\mathbf{R}_\tau(R^\tau_{jk} = \hat{\rho_\tau}(\mathbf{X}_j, \mathbf{X}_k))$, then build the estimator $\hat{\mathbf{P}} = sin(\frac{\pi}{2}R^\tau)$ with all diagonal entries being 1.

Kendall's τ rank correlation. Kendall's τ rank correlation is a well-accepted rank correlation measure of concordance for bivariate random vectors. The definition of Kendall's τ is given as follows:

Definition 3.1.9 (Kendall's τ rank correlation [12])**.** The population version of Kendall's τ rank correlation has the form:

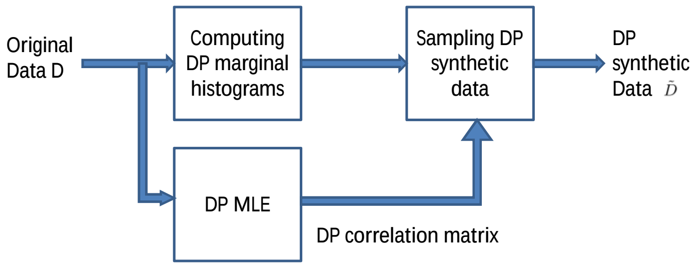
$$\rho_\tau(\mathbf{X}_j, \mathbf{X}_k) = E(sign(x_{i_1,j} - x_{i_2,j})(x_{i_1,k} - x_{i_2,k}))$$

where $(x_{i_1,j}, x_{i_1,k})$ and $(x_{i_2,j}, x_{i_2,k})$ are two different independent pairs with the same distribution. In practice, we can estimate $\rho_\tau(\mathbf{X}_j, \mathbf{X}_k)$ using $\hat{\rho_\tau}(\mathbf{X}_j, \mathbf{X}_k)$ with the form $\binom{n}{2}^{-1} \sum_{1\le i_1<i_2\le n} sign(x_{i_1,j}, x_{i_2,j})(x_{i_1,k}, x_{i_2,k})$.

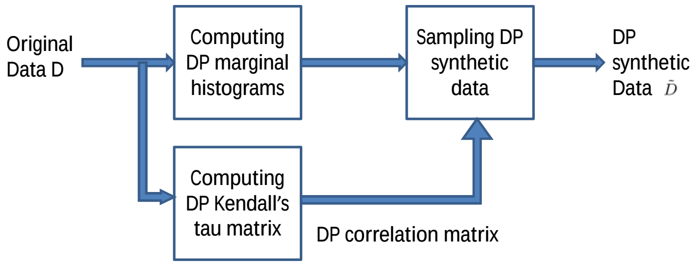
We shorten Kendall's τ rank correlation as Kendall's τ. We choose to use Kendall's τ instead of other correlation metrics such as Pearson or Spearman because it can better describe more general correlations while Pearson can only describe the linear correlation and has better statistical properties than Spearman.

3.2. DPCopula

Under differential privacy, we propose two DPCopula algorithms [17, 18] for estimating Gaussian copula functions based on multi-dimensional data, namely DP-Copula using MLE (DPCopula-MLE) and DPCopula using Kendall's τ (DPCopula-Kendall). The general idea is to estimate marginal distributions and the gaussian copula function based on the original multivariate data, then sample synthetic data from this joint distribution while preserving ϵ-differential privacy. In this section, we first present the methods of DPCopula-MLE and DPCopula-Kendall with privacy proofs and complexity analysis, then analyze their convergence properties. Finally, we present a DPCopula hybrid method to handle small-domain attributes.


Original Data D
Computing DP marginal histograms
Sampling DP synthetic data
DP synthetic Data $\tilde{D}$
DP MLE
DP correlation matrix

(a) DPCopula-MLE


Original Data D
Computing DP marginal histograms
Sampling DP synthetic data
DP synthetic Data $\tilde{D}$
Computing DP Kendall's tau matrix
DP correlation matrix

(b) DPCopula-Kendall

FIGURE 3.2. DPCopula Overview

3.2.1. DPCopula-MLE.

One basic method of DPCopula is to first compute DP marginal histograms, then estimate DP correlation matrix using the DP MLE method proposed by Dwork [5], then sample DP synthetic data. We illustrate this algorithm schematically in Figure 3.2(a). Algorithm 1 presents the steps of DPCopula-MLE. We present the details of each step below.

Algorithm 1 DPCopula-MLE algorithm

Input: Original data vector $D = (\mathbf{X}_1, \ldots, \mathbf{X}_m)$, and privacy budget ϵ.
Output: Differentially private synthetic data $\tilde{D}$

```
1. Create a differentially private marginal histogram with privacy budget $\frac{\epsilon_1}{m}$ for each dimension $X_j$, $j = 1, \ldots, m$, in the original data vector to obtain DP empirical marginal distribution $(\widetilde{U}_1, \ldots, \widetilde{U}_m)$ by equation (3.1.6);
2. Use DP MLE to estimate the DP correlation matrix $\widetilde{\mathbf{P}}$ with privacy budget $\frac{\epsilon_2}{\binom{m}{2}}$ for each correlation coefficient and $\epsilon_2 = \epsilon - \epsilon_1$;
3. Sample DP synthetic dataset $\tilde{D}$ by algorithm 3.
```

Computing DP marginal histograms. As a first step, we compute DP marginal histograms for each attribute. There are several state-of-the-art techniques for obtaining one-dimensional DP histograms effectively and efficiently, such as PSD, Privelet [41], NoiseFirst and StructureFirst [43], EFPA [47]. Here we use EFPA to generate DP marginal histograms which is superior to other methods. We note that an important feature of DPCopula is that it can take advantage of any existing methods to compute DP marginal histograms for each dimension, which can be then used to obtain DP empirical marginal distributions.

DP MLE. In step 2, we fit a Gaussian copula to the pseudo copula data generated from original data using equation 3.1.6 and 3.1.7, then use the DP MLE method to compute DP correlation matrix $\widetilde{\mathbf{P}}$. Our DP MLE method uses the similar idea with [5]. Algorithm 2 presents the general idea of DP MLE. It first divides the D horizontally into l disjoint partitions of $\frac{n}{l}$ records each, computes the MLE coefficient estimator on each partition, and then releases the average of these estimates plus some small additive noise. Here the sensitivity of each coefficient is $\frac{2}{l}$, for the diameter of each coefficient is 2. The value of l should be larger than $\binom{m}{2}/0.025\epsilon_2$ which requires a large data cardinality for high dimensions. Algorithm 2 guarantees ϵ_2 differential privacy due to theorem 3.1.2 because each partition that is disjoint with each other preserves ϵ_2 differential privacy.

Algorithm 2 DP MLE

```
**Input:** Original data vector $D = (\mathbf{X}_1, \dots, \mathbf{X}_m)$, privacy budget $\epsilon_2$, and $k \in N^+$.
**Output:** Differentially private correlation matrix estimator $\tilde{\mathbf{P}}$
  1. Divide $D$ horizontally into $l$ disjoint partitions $D_1, \dots, D_l$ with each partition having $b = \frac{n}{l}$ tuples;
  2. For each partition $D_t$, $t \in 1, \dots, l$:
       $$\tilde{\mathbf{P}^t} = \arg\max_{P_{ij} \in \Theta} \sum_{r=(i-1)b+1}^{rb} log\mathrm{C}^{Ga}_{\mathbf{P}}(x_1^r, \dots, x_m^r)$$
     where $\mathrm{C}^{Ga}_{\mathbf{P}}$ represents the density of Gaussian copula.
  3. For each $P_{ij} \in [-1, 1]$, $i, j \in 1, \dots, m$
       Compute the average value through $\bar{P}_{ij} = \frac{1}{l} \sum_{l}^{t=1} P_{ij}^t$,
       Then inject Laplace noise to $\bar{P}_{ij}$ and obtain DP $\tilde{P}_{ij}$ as
         $$\tilde{P}_{ij} = \bar{P}_{ij} + Lap[\frac{\binom{m}{2}\Lambda}{l\epsilon_2}]$$
       where $\Lambda$ is the diameter of each correlation coefficient space $\Theta$ with a value of 2;
  4. Collect all $\tilde{P}_{ij}$ to constitute the DP correlation matrix estimator $\tilde{\mathbf{P}}$
```

Sampling DP synthetic data. In step 3, we build a joint distribution based on definition 3.1.4 using the DP marginal histograms from step 1, and DP correlation matrix estimator $\tilde{\mathbf{P}}$ from step 2, then sample data points from the joint distribution. The procedure of sampling DP synthetic data is given in Algorithm 3.

Algorithm 3 Sampling DP synthetic data

```
**Input:** DP marginal histograms and DP correlation matrix $\widetilde{\mathbf{P}}$
**Output:** DP synthetic data $\hat{D}$
  1. Generate DP pseudo-copula synthetic data $(\tilde{\mathbf{T}}_1, \dots, \tilde{\mathbf{T}}_m)$:
     a. Generate a multivariate random number vector $(\tilde{\mathbf{X}}_1, \dots, \tilde{\mathbf{X}}_m)$ in an arbitrary domain following the gaussian joint distribution $\Phi(0, \tilde{\mathbf{P}})$, where $\tilde{\mathbf{P}}$ is returned by step 2 of Algorithm 1;
     b. Transform $(\tilde{\mathbf{X}}_1, \dots, \tilde{\mathbf{X}}_m)$ to $(\tilde{\mathbf{T}}_1, \dots, \tilde{\mathbf{T}}_m) \in [0, 1]^{n \times m}$, where $\tilde{\mathbf{T}}_j = \phi(\tilde{\mathbf{X}}_j)$,$j = 1, \dots, m$ and $\phi(\tilde{\mathbf{X}}_j)$ is the standard gaussian distribution;
  2. Compute DP synthetic data $\tilde{D}$ as follows:
       $$\tilde{D} = (\widetilde{F}_1^{-1}(\tilde{\mathbf{T}}_1), \dots, \widetilde{F}_m^{-1}(\tilde{\mathbf{T}}_m))$$
     where $\widetilde{F}_j^{-1}$ is the inverse of DP empirical marginal distribution function generated from the $j$th DP marginal histogram, and in the domain of the original dataset.
```

Privacy Properties. We present the following theorem showing the privacy property of the DPCopula-MLE algorithm.

THEOREM 3.2.1. Algorithm 1 guarantees ϵ - differential privacy.

PROOF. Step 1 guarantees ϵ_1 differential privacy due to theorem 3.1.1. Step 2 guarantees ϵ_2 differential privacy due to [5]. Algorithm 1 guarantees $\epsilon_1 + \epsilon_2 = \epsilon$ differential privacy due to theorem 3.1.1. □

Computation complexity. For the space complexity, the DPCopula-MLE algorithm takes $O(mn)$ (i.e. the size of the original dataset), where m is the number of dimensions, n is the number of records in the original dataset. For the time complexity, computing all DP marginal histograms take $O(\sum_{i=1}^{m}(A_i log A_i + n)) = O(mAlogA + mn)$ due to [47], where $A = max\{A_1, \dots, A_m\}$. DP MLE takes $O(l \times \frac{m^2n^2}{l^2}) = O(\frac{m^2n^2}{l})$. DPCopula-MLE takes $O(mAlogA + m^2n^2/l)$.

3.2.2. DPCopula-Kendall.

In this section, we first present the key steps of DPCopula-Kendall and then provide formal proof for the privacy guarantee. Figure 3.2(b) illustrates the process of DPCopula-Kendall. Algorithm 4 presents the detailed steps of DPCopula-Kendall. From the key steps of algorithm 4, we can see that the differential privacy guarantee relies on step 1 and step 2, which share the privacy budget. As step 1 and step 3 of algorithm 4 are the same with algorithm 1, we only present the details of step 2 below.

Algorithm 4 DPCopula-Kendall's τ algorithm

```
Input: Original data vector $(\mathbf{X}_1, \dots, \mathbf{X}_m)$ containing m attributes, privacy budget $\epsilon$
Output: Differentially private synthetic data $\tilde{D}$
  1. Compute a differentially private marginal histogram with the privacy budget $\frac{\epsilon_1}{m}$
  for each $\mathbf{X}_i$ in $D$ ;
  2. Compute the DP correlation matrix $\widetilde{\mathbf{P}}$ using algorithm 5 with privacy budget
  $\frac{\epsilon_2}{\binom{m}{2}}$ for each correlation coefficient, and $\epsilon_2 = \epsilon - \epsilon_1$;
  3. Sample DP synthetic data $\tilde{D}$ by algorithm 3.
```

Computing differentially private correlation matrix. The differentially private estimator $\tilde{\mathbf{P}}$ of the general correlation matrix is estimated by calculating noisy pairwise Kendall' τ correlation coefficients matrix. From the original data vector $(\mathbf{X}_1, \ldots, \mathbf{X}_m)$, we can compute a noisy Kendall's τ coefficient of any arbitrary two attributes $\mathbf{X}_j$ and $\mathbf{X}_k$ by the standard sample Kendall's τ coefficient $\tilde{\rho}_\tau(\mathbf{X}_j, \mathbf{X}_k)$ using Laplace mechanism that guarantees ϵ_2-differential privacy. We then construct a noisy Kendall' τ matrix $\tilde{\rho}_\tau$ with each element defined by $\tilde{\rho}^{\tau}_{jk} = \tilde{\rho}_\tau(\mathbf{X}_j, \mathbf{X}_k)$. Finally, we construct the noisy correlation matrix estimator as $\tilde{\mathbf{P}} = sin(\frac{\pi}{2}\tilde{\rho}_\tau)$ with all diagonal entries being 1. We note that $\tilde{\mathbf{P}}$ may not be a positive definite matrix (although in most cases, it is positive definite in our experience when ϵ_2 is not too small, $\epsilon_2 \geq 0.001$). In this case, $\tilde{\mathbf{P}}$ can be transformed to be positive definite using postprocessing methods like the eigenvalue procedure proposed by Rousseeuw et al. [14]. Algorithm 5 presents detailed steps of DP correlation coefficient matrix computation.

Algorithm 5 Computing differentially private correlation coefficient matrix

```
Input: Original data vector $(\mathbf{X}_1, \ldots, \mathbf{X}_m)$ containing m attributes, and privacy budget $\epsilon_2$
Output: Differentially private correlation matrix estimator $\tilde{\mathbf{P}}$
  1. Compute DP pairwise noisy Kendall' $\tau$ correlation coefficient $\hat{\rho_\tau}(\mathbf{X}_j, \mathbf{X}_k)$ as follows:
  $\tilde{\rho}_\tau(\mathbf{X}_j, \mathbf{X}_k) = \binom{n}{2}^{-1} \sum_{1 \leq i_1 < i_2 \leq n} sign(X_{i_1 j}, X_{i_2 j})(X_{i_1 k}, X_{i_2 k}) + Lap[\frac{\binom{m}{2}\Delta}{\epsilon_2}]$, where $\Delta$ is the sensitivity of each pairwise Kendall's $\tau$ coefficient with a value of $\frac{4}{n+1}$;
  2. Compute noisy correlation coefficient matrix $\tilde{\mathbf{P}}_1$ using $\tilde{\mathbf{P}}_1 = sin(\frac{\pi}{2}\tilde{\rho}_\tau)$, each element of $\tilde{\rho}_\tau$ is defined by $(\tilde{\rho}^{\tau}_{jk} = \tilde{\rho}_\tau(\mathbf{X}_j, \mathbf{X}_k)')$. If $\tilde{\mathbf{P}}_1$ is NOT positive definite, then go to step 3; else set $\tilde{\mathbf{P}} = \tilde{\mathbf{P}}_1$;
  3. Use the eigenvalue method to transform $\tilde{\mathbf{P}}_1$ to be positive definite matrix $\tilde{\mathbf{P}}_2$:
    a. Compute the eigenvalue decomposition form of $\tilde{\mathbf{P}}_1$ as $\tilde{\mathbf{P}}_1 = \mathbf{RDR}^T$, where $\mathbf{D}$ is a diagonal matrix containing all eigenvalues of $\tilde{\mathbf{P}}_1$ and $\mathbf{R}$ is an orthogonal matrix containing the eigenvectors
    b. Compute $\tilde{\mathbf{D}}$ by replacing all negative eigenvalues in $\mathbf{D}$ by a small value or their absolute values
    c. Compute $\tilde{\mathbf{P}}_2 = \mathbf{R}\tilde{\mathbf{D}}\mathbf{R}^T$ while normalizing $\tilde{\mathbf{P}}_2$ to be the correlation matrix form with diagonal elements to be 1, then set $\tilde{\mathbf{P}} = \tilde{\mathbf{P}}_2$.
```

Privacy Properties. We first present a lemma analyzing the sensitivity of the Kendall's τ coefficient followed by a theorem showing that DPCopula-Kendall satisfies ϵ-differential privacy.

LEMMA 3.2.1. The sensitivity of a pairwise Kendall's τ coefficient is $\Delta = \frac{4}{n+1}$.

PROOF. Assume we have two dataset D and D' differing in only one tuple, and let $\hat{\rho_\tau}(\mathbf{X}_i, \mathbf{X}_j)$ and $\hat{\rho_\tau}(\mathbf{X}'_i, \mathbf{X}'_j)$ be two Kendall's τ coefficients which, respectively comes from D and D', then the sensitivity of a pairwise Kendall's τ coefficient is defined by the domain of $|\hat{\rho_\tau}(\mathbf{X}_i, \mathbf{X}_j) - \hat{\rho_\tau}(\mathbf{X}'_i, \mathbf{X}'_j)|$.
Let $A = |\hat{\rho_\tau}(\mathbf{X}_i, \mathbf{X}_j) - \hat{\rho_\tau}(\mathbf{X}'_i, \mathbf{X}'_j)|$. From Definition 3.6 of Kendall's τ coefficient, we can deduce that

$$A = \frac{(n^2+n)(n_c - n_d) - (n^2-n)(n'_c - n'_d)}{\frac{1}{2}n^2(n+1)(n-1)}$$

where n_c is the number of concordant pairs of $(\mathbf{X}_i, \mathbf{X}_j)$, n_d is the number of disconcordant pairs of $(\mathbf{X}_i, \mathbf{X}_j)$, n'_c is the number of concordant pairs of $(\mathbf{X}'_i, \mathbf{X}'_j)$, and n'_d is the number of disconcordant pairs of $(\mathbf{X}'_i, \mathbf{X}'_j)$. In general, $n_c - n_d = k - [\binom{n}{2} - k]$, and $n'_c - n'_d = k + r - [\binom{n}{2} + n - (k+r)]$, where k is the number of concordance, $k = 0, 1, \ldots, \binom{n}{2}$, r is additive number of concordance after adding one tuple, $r = 0, 1, \ldots, n$. We have $(n^2+n)(n_c - n_d) - (n^2-n)(n'_c - n'_d) = 2n[2k - \binom{n}{2}] + n(n-1)(n-2r)$, where $0 \leq k \leq \binom{n}{2}$, $0 \leq r \leq n$. According to the property of inequality, we have $-2n\binom{n}{2} - n^2(n-1) \leq (n^2+n)(n_c - n_d) - (n^2-n)(n'_c - n'_d) \leq 2n\binom{n}{2} + n^2(n-1)$, followed by $|(n^2+n)(n_c - n_d) - (n^2-n)(n'_c - n'_d)| \leq 2n\binom{n}{2} + n^2(n-1) = 2n^2(n-1)$. Thus
$A = \frac{|(n^2+n)(n_c - n_d) - (n^2-n)(n'_c - n'_d)|}{\frac{1}{2}n^2(n+1)(n-1)} \leq \frac{4}{n+1}$, i.e., the sensitivity of a pairwise Kendall's τ coefficient is $\frac{4}{n+1}$, which completes the proof. □

THEOREM 3.2.2. Algorithm 4 guarantees ϵ - differential privacy and $\epsilon = m\epsilon_1 + \binom{m}{2}\epsilon_2$.

PROOF. In step 1, each margin guarantees $\frac{\epsilon_1}{m}$-differential privacy and there are m margins. Due to theorem 3.1.1, step 1 satisfies ϵ_1-differential privacy. In step 2, each pairwise coefficient guarantees $\epsilon_2/\binom{m}{2}$-differential privacy due to the above Lemma and the Laplace mechanism; and there are $\binom{m}{2}$ pairs. Due to theorem 3.1.1, step 2 satisfies ϵ_2-differential privacy. Due to theorem 3.1.1 again, Algorithm 4 satisfies $\epsilon_1 + \epsilon_2 = \epsilon$-differential privacy. □

Computation complexity. For the space complexity, DPCopula-Kendall is the same with DPCopula-MLE. For the time complexity, the complexity of each Kendall's τ takes $O(nlogn)$ using a fast Kendall's τ computation method. The total time complexity is $O(mAlogA+m^2nlogn)$. When the number of records is large, computing Kendall's τ is very time consuming. A natural technique is to compute Kendall's τ only on $\hat{n}$ sample records of the full data to reduce the computation complexity which requires $O(\frac{4}{\hat{n}+1})$ Laplace noise on each coefficient. This sampling method guarantees differential privacy by enlarging the Laplace noise from $O(\frac{4}{n+1})$ to $O(\frac{4}{\hat{n}+1})$. Here the selection of $\hat{n}$ should guarantee that the Laplace noise $O(\frac{4}{\hat{n}+1})$ be sufficiently small compared to the scale of original correlation coefficients that is $[-1, 1]$. In practice, setting $\hat{n} \geq (50m(m-1)/\epsilon_2) - 1$ is adequate. Thus, no matter how large n is, the time complexity will be fixed to $O(mAlogA + m^2)$.

3.2.3. Convergence properties of DPCopula.

In this subsection, assuming that the original data follows the Gaussian dependence structure, we provide a convergence analysis on DPCopula-Kendall, and show that the distribution of the private synthetic dataset generated by DPCopula-Kendall copula has the same joint distribution as the original dataset when the database cardinality n is sufficiently large. We leave the convergence analysis of DPCopula-MLE as our future work. We first present a few lemmas on the convergence properties of noisy empirical margin and noisy Kendall's τ coefficient, then present the main result in Theorem 3.2.3.

Lemma 3.2.1. (Convergence of private empirical marginal distribution). $\lim_{n\to\infty} \tilde{F}_n(t) = \lim_{n\to\infty} \hat{F}_n(t) = F(t)$ almost surely, where $\tilde{F}_n(t)$ is the empirical CDF based on the private histogram, $\hat{F}_n(t)$ is the empirical CDF based on the original histogram, and $F(t)$ is the population CDF when n tends to be infinity.

PROOF. Due to the analysis in [13], we can deduce that the discrimination of $\hat{F}_n(t)$ and $\tilde{F}_n(t)$ is bounded by $O(\frac{logm}{n})$. Hence, we can achieve that $\lim_{n\to\infty}|\tilde{F}_n(t)-\hat{F}_n(t)| = 0$ leading to $\lim_{n\to\infty}\tilde{F}_n(t) = \lim_{n\to\infty}\hat{F}_n(t)$ and the conclusion can be proved by the strong law of large numbers. □

Lemma 3.2.2. (Convergence of private Kendall's *tau* coefficient). Assume $\tilde{\rho}_\tau$ and ρ_τ are noisy and original Kendall's tau coefficient respectively, then $\lim_{n\to\infty}|\tilde{\rho}_\tau-\rho_\tau| = 0$.

PROOF. Since $\tilde{\rho}_\tau = \rho_\tau + Lap(\frac{4}{(n+1)\epsilon_2})$, then

$$\lim_{n\to\infty}|\tilde{\rho}_\tau-\rho_\tau| = \lim_{n\to\infty}|Lap(\frac{4}{(n+1)\epsilon_2})|$$

When ϵ_2 is a finite real number, it follows that

$$\lim_{n\to\infty}|Lap(\tfrac{4}{(n+1)\epsilon_2})| = 0,$$

leading to $\lim_{n\to\infty}|\tilde{\rho}_\tau-\rho_\tau| = 0$. □

THEOREM 3.2.3. (Convergence of DPCopula) Let $\{\tilde{F}_{1t}\},\ldots,\{\tilde{F}_{mt}\}$ be m sequences of noisy univariate marginal distribution and let $\{\tilde{C}_t\}$ be a sequence of noisy copulas; then, for every t in N^+, an m-dimensional noisy joint distribution function H_t is defined as:

$$\tilde{H}_t(x_1,\ldots,x_m) := \tilde{C}_t(\tilde{F}_{1t}(x_1),\ldots,\tilde{F}_{mt}(x_m))$$

Then the sequence $\tilde{H}_t$ converges to the joint distribution H_0 of original data in distribution, if and only if $\{\tilde{F}_{1t}\},\ldots,\{\tilde{F}_{mt}\}$ converge to $\{F_{10}\},\ldots,\{F_{m0}\}$ respectively in distribution, and if the sequence of copulas $\{\tilde{C}_t\}$ converges to $\{\tilde{C}_0\}$ pointwise in $[0,1]^2$.

PROOF. Since the copula remains invariant under any series of strictly increasing transformation of the random vector **X**, which can be considered as empirical CDF, then the Gaussian copula of Gaussian distribution $G_m(\mu, \sum)$ is identical to that of $G_m(0, P)$ where P is the correlation matrix implied by the dispersion matrix $\sum$ and this Gaussian copula is unique. Due to Lemma 3.2.2, we can deduce that

$$\lim_{n\to\infty} \tilde{\rho}_\tau = \lim_{n\to\infty} \rho_\tau = E(sign(x_j - x'_j)(x_k - x'_k))$$

in probability, where (x_j, x_k) and (x'_j, x'_k) are two distinct independent pair with the same distribution. Then, as the noisy sample correlation matrix converges in probability to the common true correlation matrix of the original data when n tends to be infinity, the noisy gaussian distribution $\tilde{G}_{mt}(0, P_t)$ which is determined only by noisy correlation matrix converges in probability to $G_m(0, P)$ due to the continuous mapping theorem. Therefore, from Theorem 3.1.3 we can imply that the noisy gaussian copula $C^{Ga}_{t,P}$ of $G_{mt}(0, P_t)$ converges pointwise to the gaussian copula C^{Ga}_P of $G_d(0, P)$ as the data cardinality n tends to be infinity.

Then for the noisy joint distribution with noisy Gaussian copula $C_{t,P}$, since the noisy margins converge to the original margins almost surely as n tends to be infinity implied by Lemma 3.2.1, then we can deduce that they converge to the original margins in distribution. Therefore, the noisy joint distribution converges in distribution to the joint distribution of the original data according to Theorem 3.1.3. □

3.2.4. DPCopula Hybrid. Although DPCopula can model continuous attributes and discrete attributes with a large domain (i.e. attributes with the number of values no less than 10), it cannot handle attributes with small domains (i.e. attributes with the number of values less than 10) in the dataset. However, we can first partition the original dataset and compute DP counts for those partitions based on small-domain attributes using other methods, such as Dwork's method, DPCube, PSD and EFPA, then use DPCopula to handle remaining large domain attributes in each partition. We demonstrate the hybrid solution in Algorithm 6. The privacy guarantee is proved in theorem 3.2.4.

THEOREM 3.2.4. Algorithm 6 guarantees ϵ-differential privacy.

Algorithm 6 DPCopula hybrid

```
Input: Original data vector $(\mathbf{X}_1, \ldots, \mathbf{X}_m)$ containing $m_1$ small-domain attributes
    and $m_2$ continuous or large-domain discrete attributes, and privacy budget $\epsilon$
Output: Differentially private synthetic data $\tilde{D}$
    1. Partition the original dataset based on small-domain attributes $A_1, \ldots, A_{m_1}$
    with domain sizes $|A_1|, \ldots, |A_{m_1}|$, and the overall number of partitions will be
    $\prod |A_i| = |A_1| \times \ldots \times |A_{m_1}|$;
    2. Compute the noisy number of tuples $\tilde{n}_i$ of the $i$th partition, $i \in \{1, \ldots, \prod |A_i|\}$
    by $n_i + X$, where $X$ is drawn from $Lap(\frac{1}{\epsilon_1})$ and $n_i$ is the original number of tuples,
    with $\epsilon_1$;
    3. For each partition, generate DP synthetic data using DPCopula and noisy
    number of tuples with $\epsilon - \epsilon_1$, then combine all DP synthetic data in all partitions
    to compose the final DP synthetic data $\tilde{D}$.
```

Proof. In step 1 and 2, each partition guarantees ϵ_1-differential privacy. Since the partitions are disjoint, they preserve ϵ_1-differential privacy overall due to theorem 3.1.2. Likewise, step 3 guarantees $(\epsilon - \epsilon_1)$-differential privacy. Algorithm 6 guarantees ϵ-differential privacy due to theorem 3.1.1. $\square$

3.3. Experiment

In this section, we experimentally evaluate DPCopula and compare it with four state-of-the-art methods. DPCopula methods are implemented in MATLAB R2010b and python, and all experiments were performed on a PC with 2.8GHz CPU and 8G RAM.

3.3.1. Experiment Setup. Datasets. We use two real datasets in our experiments: Brazil Census dataset (https://international.ipums.org) and US census dataset (http://www.ipums.org). The Brazil census dataset has 188,846 records after filtering out records with missing values and eight attributes are used for the experiments: age, gender, disability, nativity, working hours per week, education, number of years residing in the current location, and annual income. We generalized the domain of income to 586. The US Census dataset has a randomly selected 100,000 records from the original 10 million records and all four attributes are used: age, occupation, income and gender. Table 3.3 shows the domain sizes of the datasets. For nominal attributes, we convert them to numeric attributes by imposing a total order on the domain of the attribute as in [41].

(a) US census dataset

Attribute	Domain size
Age	96
Income	1020
Occupation	511
Gender	2

(b) Brazil census dataset

Attribute	Domain size
Age	95
Gender	2
Disability	2
Marital status	2
Number of Years	31
Education	140
Working hours per week	95
Annual income	586

FIGURE 3.3. Domain sizes of the real datasets

In order to study the impact of distribution, dimensionality and scalability, we also generated synthetic datasets with 50000 records. The default attribute domain size is 1000 and each margin follows the Gaussian distribution by default.

Comparison. We evaluate the utility of the synthetic data generated by DPCopula for answering random range-count queries and compare it with the state-of-the-art differentially private histogram methods. We included four methods for comparison (based on our discussions in Section 2): Privelet+ [41], PSD (Private Spatial Decomposition) KD-hybrid methods [44], Filter Priority (FP) with consistency checks [45], and P-HP [47]. Among these methods, we observed that PSD and P-HP consistently outperform others in most settings. Hence, after presenting a complete comparison on US dataset, we only show PSD and P-HP for better readability of the graphs.

For datasets with number of dimensions higher than 2 and domain size of each dimension being 1000 (ie. the number of histogram bin is larger than 10^6), we only show PSD because PSD uses the original dataset as input and hence have a space complexity of $O(mn)$) which is not affected by the domain size. In contrast, P-HP uses the histogram generated from the original data as input and hence have a time and space complexity of $O((\prod_{i=1}^{m} |A_i|)^2)$ in the worst case and $O(\prod_{i=1}^{m} |A_i|)$ respectively. Thus, the computation complexity can be extremely high because the number of bins in the histogram (i.e. $\prod_{i=1}^{m} |A_i|$) is 10^{12}, 10^{18} and 10^{24} respectively in our 4D, 6D and

8D datasets. In fact, for all methods with histograms as inputs, we cannot run their implementations directly due to the extremely high space complexity and memory constraints.

For each method, implementations provided by their respective authors are used and all parameters in the algorithms are set to the optimal values in each experiment. For comparison, we only show the results of DPCopula-Kendall, as the results of DPCopula-MLE are similar to that of DPCopula-Kendall.

Metrics. We generated random range-count queries with random query predicates covering all attributes defined in the following:
Select COUNT(*) from D
Where $A_1 \in I_1$ and $A_2 \in I_2$ and...and $A_m \in I_m$
For each attribute A_i, I_i is a random interval generated from the domain of A_i.

The query accuracy is primarily measured by the relative error defined as follows: For a query q, $A_{act}(q)$ is the true answer to q on the original data. $A_{noisy}(q)$ denotes the answer to q when using DP synthetic data generated from DPCopula or the DP histogram constructed by other methods. Then the relative error is defined as:

$$RE(q) = \frac{|A_{noisy}(q) - A_{act}(q)|}{max\{A_{act}(q), s\}}$$

where s is a sanity bound to mitigate the effects of queries with extremely small query answers (a commonly used evaluation method from existing literatures, e.g. [41]). For most datasets, s is set to 1 by default to avoid division by 0 when $A_{act}(q) = 0$. For the US dataset, s is set to 0.05% of the data cardinality, nearly consistent with [41]. For the brazil dataset, s is set to 10.

While we primarily use relative error, we also use absolute error when it is more appropriate and clear to show the results for extremely sparse data, in which case, the true answers are extremely small. The absolute error is defined as $ABS(q) = |A_{noisy}(q) - A_{act}(q)|$.

Table 3.2. Experiment Parameters

Parameter	Description	Default value
n	number of tuples in D	50000
ϵ	Privacy budget	1.0
m	number of dimensions	8
s	Sanity bound	1
k	ratio of ϵ_1 and ϵ_2	8
A_i	domain size of ith dimension	1000

In each experiment run, 1000 random queries are generated and the average relative error is computed. The final reported error is averaged over 5 runs. Table 3.2 summarizes the parameters in the experiments.

3.3.2. DPCopula Methods. We first evaluate the impact of the parameter k in the DPCopula method and compare the two DPCopula methods: DPCopula-Kendall and DPCopula-MLE.

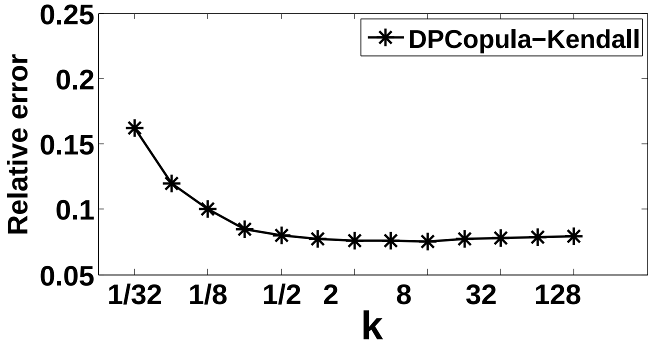
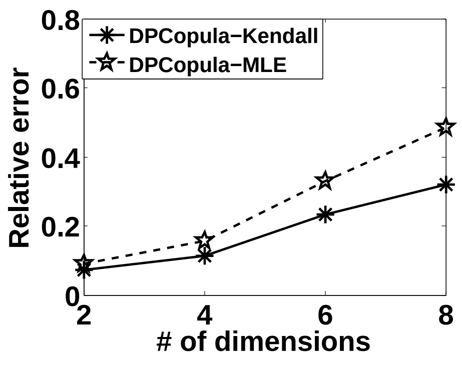
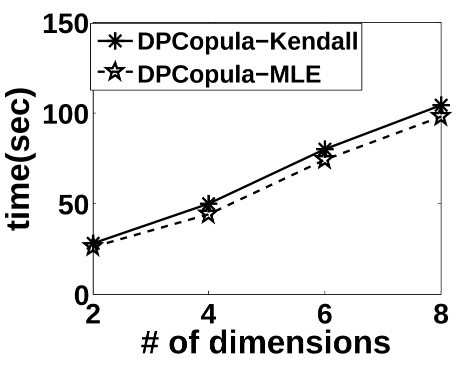


Figure 3.4. Relative error vs. Ratio k



(a) Relative error

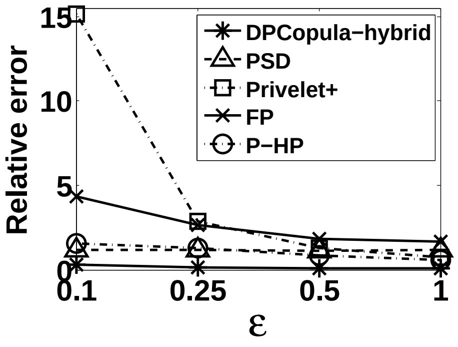


(b) Computation time vs. dimension

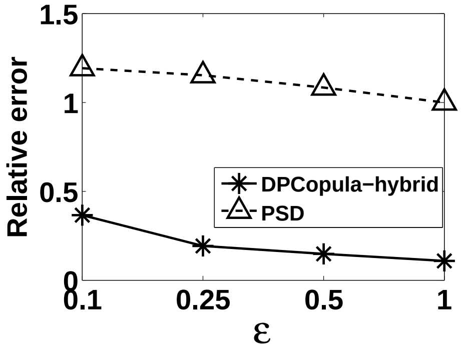
Figure 3.5. DPCopula-Kendall vs. DPCopula-MLE

Impact of Parameter k on DPCopula. Since k is the only algorithmic parameter in the DPCopula method, we first evaluate its impact on the effectiveness of the method. Figure 3.4 shows the relative error of DPCopula-Kendall method for random count queries with respect to varying k for 2D synthetic data. DPCopula-MLE has similar trends and we omit it for the clarity of the graph. We observe that when k is less than 1, the relative error clearly degrades as k increases. When k is greater than 1, the relative error does not change significantly. This shows that having a higher budget allocated for computing differentially private margins than the coefficients ensures better query accuracy. In addition, the method is quite robust and insensitive to the value of k as long as it is greater than 1, which alleviates the burden of parameter selection on the users. For the remaining experiments, we set the value of k to 8.

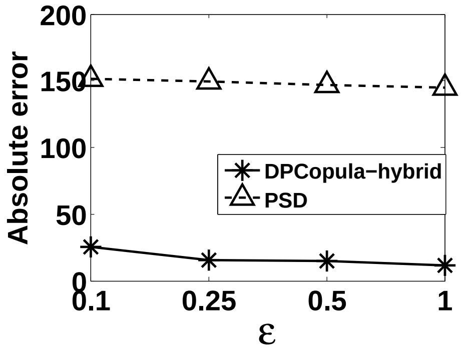
DPCopula-MLE vs. DPCopula-Kendall. Figure 3.5 investigates the trade-off between two DPCopula methods. Figure 3.5(a) compares the relative error for random queries of the two methods on synthetic data with varying number of dimensions and $n = 10^6$ considering the sensitivity of DPCopula-MLE. We observe that DPCopula-Kendall performs better than DPCopula-MLE. This is because the sensitivity of the general coefficient in DPCopula-MLE is higher than DPCopula-Kendall. As a consequence, the correlation matrix estimated by DPCopula-Kendall is more accurate than DPCopula-MLE. Figure 3.5(b) shows the runtime of the two methods. We can see that with higher dimensions, the time to compute the coefficients becomes longer because the time complexity of DPCopula is quadratic with the number of dimensions. We use the sampling method in all experiments to reduce the computation time. DPCopula-Kendall has a slightly higher computation overhead than DPCopula-MLE while the total computation time for both methods are quite efficient. We show that the computation time of DPCopula is acceptable for various data cardinalities and dimensions in later experiments. In the remaining experiments, we only use DPCopula-Kendall to compare with other methods.



(a) US-4D



(b) Brazil-8D

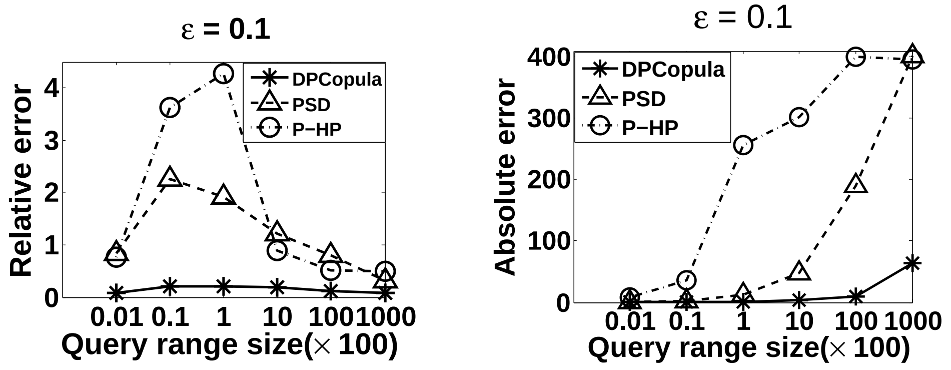


(c) Brazil-8D(absolute error)

FIGURE 3.6. Relative error vs. differential privacy budget

3.3.3. Comparison on real datasets. Query accuracy vs. differential privacy budget. Figure 3.6 compares DPCopula with other methods with respect to varying differential privacy budget. Figure 3.6(a)-(b) shows the relative error for random range count queries on the US census dataset and Brazil dataset, respectively. Note that we use DPCopula-hybrid on top of DPCopula-Kendall for both datasets since they contain binary attributes. From both figures, we observe that DPCopula outperforms all the other methods and their performance gap expands as the privacy budget decreases. The noise incurred by partitioning small domain attributes imposes little impact on the performance of DPCopula. In addition, the accuracy of DPCopula is robust against various epsilon values. This overall good performance is due to the fact that DPCopula method only computes DP margins and DP correlation matrix whose influence on the accuracy is much smaller than the margins. Meanwhile, the other methods require noise being added to histogram cells or partitions and introduce either large perturbation errors or estimation errors.

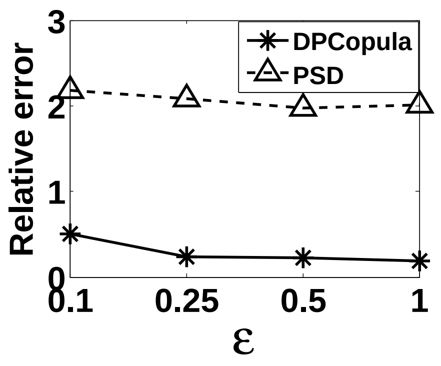
3.3.4. Comparison on synthetic datasets. We use synthetic datasets to evaluate the impact of query range size, distributions of each dimension, and dimensionality on the error, since we can vary these parameters easily in synthetic data.



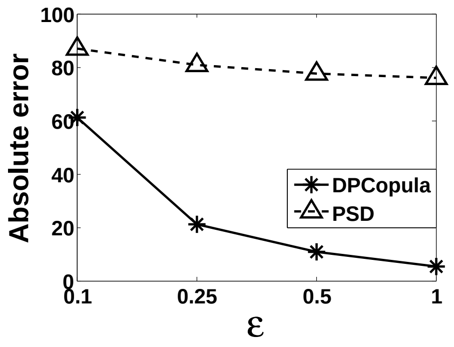
(a) Relative error (b) Absolute error

FIGURE 3.7. Query accuracy vs. query range size

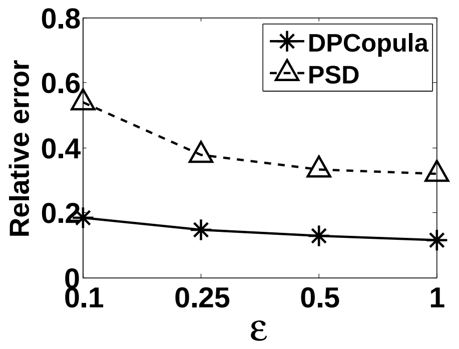
Query accuracy vs. query range size. We study the impact of query range size on the query accuracy for different methods. For each query range size, we randomly generated queries such that the product of the query ranges on each dimension is the same. We use 2D synthetic data in order to include P-HP. We set the privacy budget ϵ to be 0.1 to better present the performance difference of three methods. The trend is similar for PSD and DPCopula in higher dimension data. Figure 3.7 presents the impact of various query range sizes on the query accuracy in terms of relative error and absolute error. DPCopula outperforms PSD and P-HP. For all methods, the relative error gradually degrades as the query range size increases while the absolute error has the contrary trend. The reason is that when the query size is small, the true answer $A_{act}(q)$ is also small which may incur a small absolute error but large relative error. For the cell-based query (i.e. query range size is 1), the average relative error is small because the relative errors of most cell-based queries are zeros, which greatly reduces the average value.



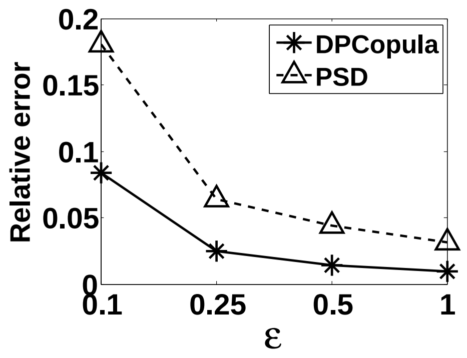
(a) Gaussian



(a) Gaussian(absolute error)



(b) Uniform

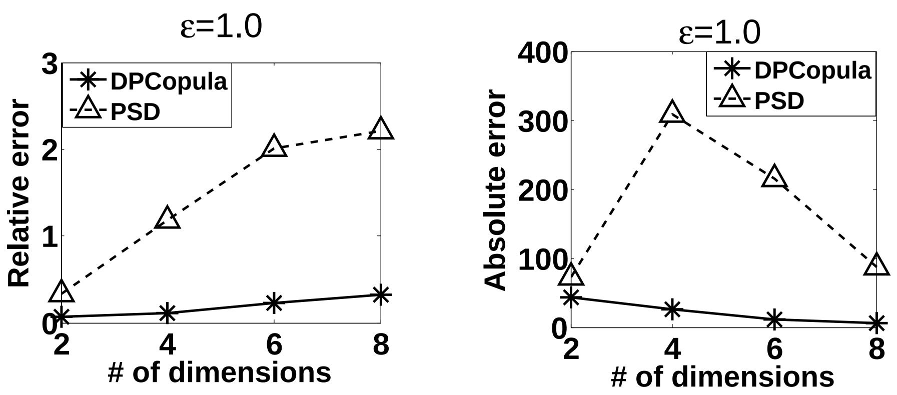


(c) Zipf

FIGURE 3.8. Relative error vs. distribution

Relative error vs. distribution. Figure 3.8 presents the relative error for 8D data with Gaussian dependence and all margins respectively generated from the Gaussian distribution, uniform distribution and zipf distribution, under various ϵ values. Akin to the results in Figure 3.6, DPCopula performs best in all distributions, and significantly outperforms PSD especially when the margin is skewed. Meanwhile, this verifies that DPCopula using Gaussian copula performs well not only for data with Gaussian distributions but also for data with different marginal distributions as long as they follow the Gaussian dependence. An interesting phenomenon is that DPCopula performs better on the uniform and zipf data than Gaussian distribution data. This is because the method used for generating marginal DP histograms in DPCopula, EFPA, performs better on uniform-distributed data than skewed data.

Query accuracy vs. dimensionality. We study the effect of the dataset dimensionality as shown in Figure 3.9. All marginal distributions of synthetic datasets in

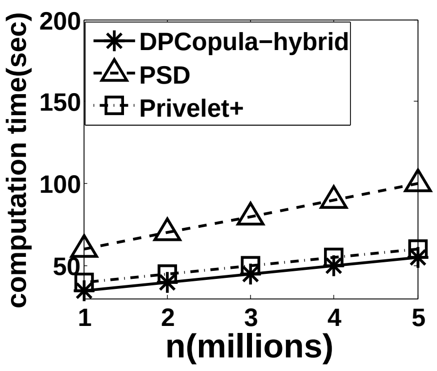


(a) Relative error (b) Absolute error

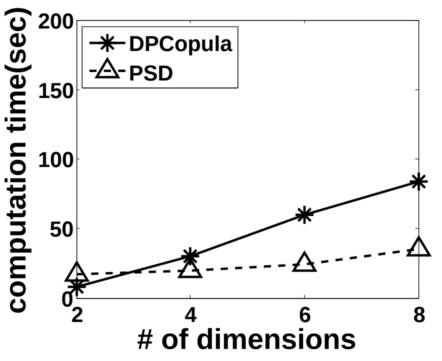
FIGURE 3.9. Query accuracy vs. dimensionality

various dimensions are Gaussian distribution with domain size of 1000. We set the dimensionality ranging from 2D to 8D which corresponds to domain space of 10^6 to 10^{24}. So the dataset is highly sparse with only 50000 records. For all dimensions from 2D to 8D, DPCopula again outperforms PSD. The 2D data has the lowest relative error and absolute error for both methods. The query accuracy of all methods from 4D to 8D gradually drops with the performance gap gradually expanding as the number of dimensions increases. For DPCopula, this is due to the fact that for a fixed overall privacy budget ϵ, higher dimensionality means less privacy budget is allocated to each margin and correlation coefficient incurring larger amount of noise. For PSD, consistent with the analysis in [44], higher dimensionality will increase the size of the domain space $\prod_{i=1}^{m} |A_i|$, resulting in larger relative error. We can also observe that the increasing relative errors for DPCopula are incurred by the small true answers with higher dimensions.

Scalability. Figure 3.10(a) illustrates the computation time with various data cardinality n using the 4D US census dataset. Observe that all three techniques run linear time with respect to n. Computing the correlation matrix is not a bottleneck for DPCopula as we use the sampling technique. PSD incurs a higher computation overhead than DPCopula and Privelet+ since its time complexity $O(m\hat{n}log\hat{n})$ is linearithmic with $\hat{n}$, where $\hat{n} = 0.01 \times n$. Figure 3.10(b) illustrates the computation time



(a) Time vs. data cardinality



(b) Time vs. dimension

FIGURE 3.10. Time efficiency

with various dimensions and data cardinality fixed to 50000. DPCopula has a higher computation overhead than PSD because the time complexity is quadratic with the number of dimensions but the time for 8D is still quite acceptable. In contrast, all the other methods including EPFA that use histograms as input are not shown here due to their high time and space complexity due to the large domain sizes.

CHAPTER 4

Privacy-preserving dynamic histogram release with distance-based sampling

In this chapter, we address the problem of releasing series of dynamic datasets in real time with differential privacy, using a novel adaptive distance-based sampling approach. Our first method, DSFT, uses a fixed distance threshold and releases a differentially private histogram only when the current snapshot is sufficiently different from the previous one, i.e., with a distance greater than a predefined threshold. Our second method, DSAT, further improves DSFT and uses a dynamic threshold adaptively adjusted by a feedback control mechanism to capture the data dynamics. Extensive experiments on real and synthetic datasets demonstrate that our approach achieves better utility than baseline methods and existing state-of-the-art methods.

4.1. Preliminaries

In this section, we formally define the problem of releasing series of real-time dynamic histograms or datasets and introduce definitions on user-level differential privacy and w-event privacy. We summarize all frequently used notations in Table 4.1.

4.1.1. Problem definition. Let N denote the total number of time points. Let $\mathbf{D}$ denote a series of original dynamic datasets and D_i be a dataset snapshot at time stamp t_i. We assume all snapshots have the same domain universe U, the product of domains of all attributes. For every t_i, we are to release a private dataset $\tilde{D}_i$. Over the N time stamps, the series of privately released dynamic datasets $\tilde{\mathbf{D}} = \{\tilde{D}_i : 1 \leq i \leq N\}$ should guarantee user-level ϵ-differential privacy.

TABLE 4.1. Frequently used notations

Notation	Discription
$\mathbf{D}$	A series of original dynamic datasets
$\tilde{\mathbf{D}}$	A set of released DP datasets for $\mathbf{D}$
D_i or $\tilde{D}_i$	Snapshot of $\mathbf{D}$ or $\tilde{\mathbf{D}}$ at time point t_i
$\mathbf{H}$	A series of original dynamic histograms
$\tilde{\mathbf{H}}$	A set of released DP histograms for $\mathbf{H}$
H_i or $\tilde{H}_i$	Snapshot of $\mathbf{H}$ or $\tilde{\mathbf{H}}$ at time point t_i
N	Number of time points
C	Cutoff point (i.e. the upper bound of the number of released DP datasets)
U	Domain universe or number of histogram bins
ϵ	Overall privacy budget
ϵ_1	Privacy budget for the decision step
ϵ_2	Privacy budget for the sampling step
$d(D_i, \tilde{D}_j)$	The distance between D_i and $\tilde{D}_j$
Δ	The sensitivity of L_1 distance

In this chapter, we call $\mathbf{H}$ as a series of original dynamic histograms (corresponding to $\mathbf{D}$) with H_i as a snapshot at t_i, and $\tilde{\mathbf{H}}$ as a series of released private dynamic histograms with $\tilde{H}_i$ as a private snapshot at t_i. Since a dataset can be transformed to a histogram, and a synthetic dataset can be constructed from a histogram, $\mathbf{D}$ and $\mathbf{H}$ are interchangeable in this paper.

4.1.2. Differential Privacy. Intuitively, a randomized mechanism $\mathcal{A}$ is differentially private if its outcome is not significantly affected by the removal or addition of any record. ϵ-differential privacy is formally defined as $Pr[\mathcal{A}(D) \in \mathcal{O}] \leq e^{\epsilon} Pr[\mathcal{A}(D') \in \mathcal{O}]$, where $\mathcal{O}$ is any arbitrary set of possible outputs of $\mathcal{A}$, D and D' are two neighbouring datasets differing in at most one record (i.e. D can be obtained from D' by adding or removing at most one record). In our problem definition, an adversary should learn approximately the same information about any individual user given $\tilde{\mathbf{D}}$, irrespective of its presence or absence in $\mathbf{D}$, and one individual can be present in up to N snapshots in $\mathbf{D}$. Two series of dynamic datasets $\mathbf{D}$ and $\hat{\mathbf{D}}$ are user-level neighbors if one can be obtained by adding or removing one individual (including all its occurrences in the snapshots) from the other. Then user-level ϵ-differential privacy is defined as below.

Definition 4.1.1 (user-level ϵ-differential privacy)**.** Let $\mathcal{A}$ be a randomized mechanism over two user-level neighbors $\mathbf{D}$, and $\hat{\mathbf{D}}$ which differ in one user's presence in the entire series, and let $\mathcal{O}$ be any arbitrary set of possible outputs of $\mathcal{A}$. Algorithm $\mathcal{A}$ satisfies ϵ-differential privacy iff the following holds

$$Pr[\mathcal{A}(\mathbf{D}) \in \mathcal{O}] \leq e^{\epsilon} Pr[\mathcal{A}(\hat{\mathbf{D}}) \in \mathcal{O}]$$

Laplace Mechanism. Dwork et al. [3] show that ϵ-differential privacy can be achieved by adding i.i.d. Laplace noise to query result $q(D)$, where D is a dataset. Formally, $\tilde{q}(D) = q(D) + (\nu_1, \ldots, \nu_M)'$, where $\nu_i \sim Lap(0, \frac{GS(q)}{\epsilon})$, for $i = 1, \ldots, M$, and M is the dimension of $q(D)$. ν_i follows a Laplace distribution with mean zero and scale $\frac{GS(q)}{\epsilon}$, where $GS(q)$ denotes the global sensitivity [3] of the query q. The global sensitivity is the maximum L_1 distance between the results of q from any two neighbouring datasets D and D', formally defined as $GS(q) = max_{D,D'}||q(D) - q(D')||_1$. In our problem setting, the global sensitivity of any two user-level neighbors $\mathbf{D}$ and $\hat{\mathbf{D}}$ is formally defined as

$$GS(q) = max_{\mathbf{D},\hat{\mathbf{D}}}||q(\mathbf{D}) - q(\hat{\mathbf{D}})||_N.$$

For a sequence of DP mechanisms, the sequential composition theorem [6] guarantees its overall privacy as follows:

Theorem 4.1.1 (Sequential Composition [6]). For a sequence of n mechanisms $M_1, \ldots, M_n$ and each M_i provides ϵ_i-differential privacy, the sequence of M_i will provide $(\sum_{i=1}^{n} \epsilon_i)$ differential privacy.

Hence, one way to achieve epsilon-differential privacy for the entire series of $\mathbf{D}$ is to apply Laplace mechanism for each D_i with noise $Lap(\frac{N}{\epsilon})$, which leads to $O(N)$ noise.

(α, σ)**-usefulness.** We use a formal utility metric (α, σ)-usefulness [32] to analyze the utility of each snapshot $\tilde{D}_i$ in $\tilde{\mathbf{D}}$.

Definition 4.1.2 ((α, σ)-usefulness)**.** A randomized mechanism $\mathbf{A}$ is (α, σ)-useful for queries in class $\mathcal{C}$ if with probability $1 - \sigma$, for every query $Q \in \mathcal{C}$ and a dataset D, $\mathbf{A}(D) = \tilde{D}$, $|Q(D) - Q(\tilde{D})| \leq \alpha$.

4.1.3. W-event privacy. W-event privacy [39] is proposed as an extension of differential privacy to address release of infinite streams. It guarantees user-level ϵ-differential privacy for every sub sequence of length w (or over w timestamps) **anywhere** (i.e. it can start from any timestamp) in the original series of dynamic datasets. w-neighboring series of dynamic datasets, $\mathbf{D}_w$, and $\hat{\mathbf{D}}_w$, can be defined as the user-level neighbors under any sub sequence of length w anywhere. w-event privacy can be formally given as below:

Definition 4.1.3 (w-event ϵ-differential privacy)**.** Let $\mathcal{A}$ be a randomized mechanism over two w-neighboring series of dynamic datasets $\mathbf{D}_w$, and $\hat{\mathbf{D}}_w$, and let $\mathcal{O}$ be any arbitrary set of possible outputs of $\mathcal{A}$. Algorithm $\mathcal{A}$ satisfies w-event ϵ-differential privacy (or, w-event privacy) iff the following holds

$$Pr[\mathcal{A}(\mathbf{D}_w) \in \mathcal{O}] \leq e^{\epsilon} Pr[\mathcal{A}(\tilde{\mathbf{D}}_w) \in \mathcal{O}]$$

4.1.4. Baseline and existing state-of-the-art solutions. Given our problem of releasing dynamic datasets under user-level privacy, we review some baseline and existing state-of-the-art methods which will motivate our approach. We will also compare our approach with these methods in the experiment section.

Baseline method. A baseline method is to apply existing "one time" DP histogram release methods to the dataset at every time point. If each released DP histogram preserves $\frac{\epsilon}{N}$-differential privacy, the series of N dynamic datasets guarantee ϵ-differential privacy by sequential composition theorem. This results in an overall noise of $O(N)$ which can be extremely large for large N. In an unbounded setting with N being infinite, this method will not be useful.

Fixed-sampling method. Another potential solution is to release $\frac{N}{I}$ DP histograms periodically given a sampling interval I. Privacy budget $\epsilon/\frac{N}{I}$ is allocated to each dataset at the sampling time point, and the entire private dataset series preserve ϵ-differential privacy. Unfortunately, the pre-defined sampling interval may not accurately capture the update pattern in the original series of dynamic datasets, leading to either high perturbation errors if sampling too frequently or large update errors if sampling not frequently enough or at wrong time points.

Approaches in w-event privacy. [39] proposes a sampling approach which computes the noisy distance between the dataset at the current time point and the original dataset at the latest sampling point, and then compares the noisy distance with the perturbation noise to be added if current dataset is to be released. If the distance is greater than the perturbation noise, a noisy dataset is released at current time stamp. The perturbation noise is determined by their privacy budget allocation schemes, Budget Distribution (BD) and Budget Absorption (BA), that allocate the budget to different timestamps in the w-event window. BD allocates the privacy budget in an exponentially decreasing fashion, in which earlier timestamps obtain exponentially more budget than later ones. BA starts by uniformly distributing the budget to all w timestamps, and accumulates the budget of non-sampling timestamps, which can be allocated later to the sampling timestamps. A main drawback of their approach is that the privacy budget may be exhausted prematurely (sampling too frequently in the beginning) or not fully utilized during all w timestamps (sampling not frequent enough), leading to suboptimal utility of the released data.

4.2. Adaptive Sampling Approach

We propose an adaptive distance-based sampling approach to address the dynamics of evolving datasets under user-level differential privacy. Instead of generating a differentially private histogram at each time stamp, we only compute new histograms when the update is significant, i.e., the distance between the current dataset and the latest released dataset is higher than a threshold. The key observation is that datasets may be subject to small updates at times. Distance-based sampling allows us to release a new histogram only when the datasets have significant updates, hence saving the privacy budget and reducing the overall error of released histograms. In contrast to [39], we use an explicit threshold for distance comparison to determine the sampling points, which provides two advantages: 1) we can predefine a threshold based on the expected update rate of the data if there is prior domain knowledge, 2) we can dynamically adjust the threshold in a principled way based on data dynamics.

In this section, we first present the basic method, DSFT, which uses a predefined fixed threshold. This will allow us to analyze its privacy property which also applies to our adaptive method and facilitate our description of the adaptive method. We then introduce our adaptive method, DSAT, which dynamically adjusts the threshold in a principled way to adapt to the data dynamics.

4.2.1. DSFT. DSFT (Distance-based Sampling with Fixed Threshold) uses a fixed threshold and is divided into two steps at each time point ti: decision and sampling. The decision step computes a noisy distance between the original dataset H_i at current time stamp and the latest released histogram $\tilde{H}_j$ and determines if it is larger than a noisy threshold $\tilde{T}$. If yes, the sampling step generates a new DP histogram $\tilde{H}_i$, otherwise it outputs the previous $\tilde{H}_j$. The overall privacy budget is divided between the decision (ϵ_1) and sampling (ϵ_2) steps which are designed to guarantee differential privacy as we will analyze later.

Algorithm 1 presents DSFT. Line 1-4 initializes the privacy budget for the two steps, computes the noisy threshold, and releases a DP histogram at the first time stamp. Line 5-11 carry out the decision (line 7-8) and sampling (line 8-9) steps for each time point t_i if the number of released histograms is below the cutoff point C, and releases the last histogram with all remaining budget. For the distance $d(H_i, \tilde{H}_j)$, we use the L_1 distance in our implementation and other distance metrics (e.g. KL divergence) can be also used.

Algorithm 7 Distance-based Sampling with Fixed Threshold Algorithm (DSFT)

```
Input:  $\mathbf{D} = \{D_i | 1 \le i \le N, i \in Z\}$, $T$, $C$ and $\epsilon$.
Output:  $\tilde{\mathbf{D}} = \{\tilde{D}_i | 1 \le i \le N, i \in Z\}$
 1: Set $\epsilon_1 = k\epsilon$, $\epsilon_2 = \epsilon - \epsilon_1$, $k$ is computed due to theorem 4.3.4;
 2: Set $\tilde{T} = T + Lap(\frac{2\Delta}{\epsilon_1})$, $\Delta$ is computed due to lemma 4.2.1;
 3: For $D_1$, release a DP dataset $\tilde{D}_1$ with $\frac{\epsilon_2}{C}$ privacy budget;
 4: Set $count = 1$, and $j = 1$;
 5: for each time point $t_i$ with $i \ge 2$ do
 6:     if $count \ge C$, then set $\tilde{D}_i = \tilde{D}_j$ continue;
 7:     Set $\tilde{d}(D_i, \tilde{D}_j) = d(D_i, \tilde{D}_j) + Lap(\frac{2C\Delta}{\epsilon_1})$
 8:     if $\tilde{d}(D_i, \tilde{D}_j) \ge \tilde{T}$, then release $\tilde{D}_i$ at $t_i$ with $\frac{\epsilon_2}{C}$ budget, and set $count =$
        $count + 1$, and $j = i$;
 9:     else use $\tilde{D}_j$ as the release of $D_i$;
10:     if $i == N$ and $count < C$, then release $\tilde{D}_N$ with all remaining privacy budget;
11: end for
```

4.2.2. DSAT. In DSFT, a prior knowledge on $\mathbf{D}$ is needed for the user to determine an appropriate value T. Suppose there exists an optimal value of T which can enable the algorithm to exactly generate C DP histograms. If the threshold T is higher than the optimal value, there will be remaining privacy budgets that are not utilized. On the contrary, if T is smaller than the optimal value, the privacy budget will be exhausted prematurely, resulting in update errors for remaining time points. In this section, we present DSAT, Distance-based Sampling with Adaptive Threshold, that releases a series of DP dynamic histograms while adaptively adjusting the threshold T_i for each time point, based on data dynamics. With DSAT, we do not have to find an optimal value of T, which may be difficult in practice.

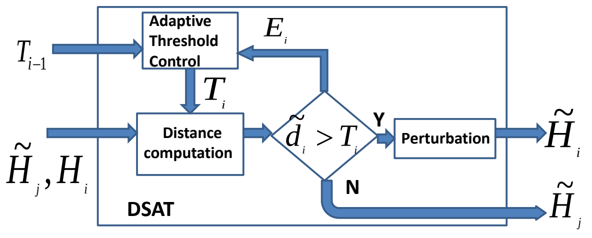


FIGURE 4.1. DSAT Framework

Figure 4.1 illustrates the framework of DSAT. Intuitively, we wish to have C sampling points over N time points, hence our target sampling rate is $\frac{C}{N}$. Suppose we have released C_i histograms at t_i. If $\frac{C_i}{i} < \frac{C}{N}$, we need to decrease the threshold to allow more sampling time points, and vice versa. For each t_i, we adjust the threshold based on the feedback error between the update ratio $\frac{C_i}{i}$ at t_i and the target ratio $\frac{C}{N}$, which is formally defined below.

Definition 4.2.1 (Feedback Error)**.** We define the feedback error E_i at t_i as follows:

$$E_i = |\frac{C_i}{i} - \frac{C}{N}| \tag{4.2.2}$$

where C_i means the number of sampling time points till t_i, C is the cutoff point, and N is the total number of time points.

DSAT adopts a PID (Proportional-Integral-Derivative) [31], a generic control loop feedback mechanism, to dynamically adjust the threshold T over time. Under our problem setting, we redefine the three correcting terms, *Proportional*, *Integral*, and *Derivative*, with the feedback error defined in Equation (4.2.2). These three terms are summed to compute the output u_i of PID controller at t_i. The final PID algorithm is defined as:

$$u_i = \underbrace{\theta_P \times e_i}_{proportional \quad term} + \underbrace{\theta_I \times \sum_{\tau = t_i - w + 1}^{t_i} e_\tau}_{integral \quad term} + \underbrace{\theta_D \times \frac{e_i - e_j}{t_i - t_j}}_{derivative \quad term} \tag{4.2.3}$$

where θ_P, θ_I, θ_D are respectively the proportional gain, the integral gain, and the derivative gain, e_τ is the error at t_τ, t_i is the current time point, t_j is the latest sampling time point.

Proportional term: The first proportional term produces an output value that is proportional to the current error e_i. The proportional term can be amplified by the *proportional gain* θ_P. In our context, the error e_i at the current time point t_i is calculated by

$$e_i = \frac{|E_i - \delta|}{\delta} \tag{4.2.4}$$

where E_i is the feedback error defined in equation (4.2.2), parameter δ is the set point for E_i. We assume δ is 5% in our empirical studies, i.e. the maximum tolerance for the feedback error is 5%. It can be determined by users according to specific applications. The proportional term is defined as $\theta_P \times e_i$.

Integral term: The integral term is to eliminate the cumulated offset through multiplying the sum of the instantaneous error over time by the integral gain. We define the integral term as $\theta_I \times \sum_{\tau=t_i-w+1}^{t_i} e_\tau$, where θ_I is the *integral gain* and w represents the integral time window denoting how many recent errors are taken.

Derivative term: The derivative term determines the slope of error over time and changes the PID output in proportion to this rate of change via the *derivative gain* θ_D. It is defined as $\theta_D \times \frac{e_i - e_j}{t_i - t_j}$. Given the PID error u_i, a new threshold T_i produced at the current time point t_i can be determined as follows:

$$T_i = T_{i-1} + sign(\frac{C_i}{i} - \frac{C}{N}) \times \theta \times u_i \tag{4.2.5}$$

T_{i-1} is the threshold produced at the previous time point t_{i-1}. Parameter θ determines the magnitude of impact of PID error on the T_i. $sign(.)$ is a sign function, indicating that if the update ratio $\frac{C_i}{i}$ is larger than the target ratio $\frac{C}{N}$, we need to increase T_j to generate less DP histograms and reduce the update ratio, and vice versa. Our DSAT uses only the proportional term in equation 4.2.3 in our experiment setting, for simplicity. That means, we set $\theta_P = 1$, $\theta_I = 0$, $\theta_D = 0$, and u_i is the same with e_i as defined in equation (4.2.4).

Algorithm 8 presents DSAT. We use T_i to denote the produced threshold at t_i and other notations are the same as Algorithm 7. In Line 1, T_1 is set to be $T + Lap(\frac{\Delta}{\hat{\epsilon}_1})$. Different from Algorithm 7, $\hat{\epsilon}_1$ is a tiny privacy budget because the initial value T_1 is not significant in DSAT. We only need to bound it between 0 and 2, which is the domain of the L1 distance. Line 2 uses $\tilde{D}_1$ for the first M time points where M is a small integer number to allow a burn-in period and enough discrepancy to be accumulated, avoiding frequent updating of T_i during the beginning time periods. M

Algorithm 8 Distance-based Sampling with Adaptive Threshold Algorithm (DSAT)

```
Input: $\mathbf{D} = \{D_i | 1 \le i \le N, i \in Z\}$, $T$, $C$ and $\epsilon$.
Output: $\tilde{\mathbf{D}} = \{\tilde{D}_i | 1 \le i \le N, i \in Z\}$
 1: Run step 1,2,3,4 in Algorithm 7;
 2: Skip the first $M$ timestamps;
 3: for each time point $t_i$ with $i > M$ do
 4:     if $count \ge C$, then set $\tilde{D}_i = \tilde{D}_j$
 5:     Set $\tilde{d}(D_i, \tilde{D}_j) = d(D_i, \tilde{D}_j) + Lap(\frac{2C\Delta}{\epsilon_1})$
 6:     Set $E_i = |\frac{count}{i} - \frac{C}{N}|$, $e_i = \frac{|E_i - \delta|}{\delta}$, and $u_i = \theta e_i$;
 7:     if $\frac{count}{t} - \frac{C}{N} \le 0$, then set $\tilde{T}_i = max\{0, \tilde{T}_{i-1} - u_i\}$
 8:     else set $\tilde{T}_i = min\{2, \tilde{T}_{i-1} + u_i\}$;
 9:     if $\tilde{d}(D_i, \tilde{D}_j) \ge \tilde{T}_i$ then
10:         release a DP dataset $\tilde{D}_i$ at $t_i$ with $\frac{\epsilon_2}{C}$ budget, and set $count = count + 1$,
            $j = i$;
11:     else
12:         release $\tilde{D}_j$;
13:     end if
14:     if $i == N$ and $count < C$ then
15:         release $\tilde{D}_N$ with all remaining privacy budget;
16:     end if
17: end for
```

can be user-specified and is not a sensitive parameter besides that it is much smaller than N. The algorithm from Line 3 to Line 12 is similar to Algorithm 7 except Line 6 to Line 8 which use the PID control to adaptively adjust and generate a new threshold T_i.

4.2.3. Privacy Analysis. Sensitivity analysis of L_1 Distance. In the sensitivity analysis, we use n_p (n_q) to denote the sum of all histogram bin counts of the histograms H_p (H_q). U is the number of histogram bins. Since the L_1 distance of Algorithm 1 and Algorithm 2 is computed using one private histogram and one original histogram, we only need to protect privacy for the original histogram.

Lemma 4.2.1. The sensitivity of L_1 distance $d(\tilde{H}_p, H_q)$ is $\Delta = \frac{2}{n_q - 1}$, where $\tilde{H}_p$ (H_q) is a DP histogram with the sum of all histogram bin counts as n_p (n_q). (Proof omitted due to space limitation)

Privacy guarantee. Inspired by Hardt et al. [33], we formally provide the proof of privacy guarantee for the decision stage below. The intuition behind theorem 4.2.1 is that, the noises on both sides of $d(D_i, \tilde{D}_j) + Lap(\frac{2C\Delta}{\epsilon_1}) \geq T + Lap(\frac{2\Delta}{\epsilon_1})$ are necessary for the decision stage to be differentially private, even though T is publicly known.

THEOREM 4.2.1. In algorithm 7, the decision stage guarantees ϵ_1-differential privacy.

PROOF. $\mathbf{D}$ is a series of dynamic datasets with $\mathbf{D} = (D_1, \ldots, D_N)$ over N time points. $\hat{\mathbf{D}}$ is the user-level neighbor of $\mathbf{D}$, which is $\hat{\mathbf{D}} = (\hat{D}_1, \ldots, \hat{D}_N)$. We say $\hat{\mathbf{D}}$ is the user-level neighbour of $\mathbf{D}$ if we can obtain $\hat{\mathbf{D}}$ by removing or adding only one individual user from $\mathbf{D}$ by the definition in section 3.2.

Let d_i denote $d(D_i, \tilde{D}_j)$ for every $i, j \in [N]([N] = \{1, \ldots, N\})$ beginning with $i = 2$, which is the true distance between D_i ad $\tilde{D}_j$, where $\tilde{D}_j$ is the private dataset released in the latest sampling time point t_j. Let $\tilde{d}_i$ denote $\tilde{d}(D_i, \tilde{D}_j)$, which is the DP L_1 distance.

For all pairs of user-level neighbours $\mathbf{D}$ and $\hat{\mathbf{D}}$, and the corresponding L_1 distance vectors $\mathbf{d} = (d_1, \ldots, d_N)$, we need to prove:

$$\log(\frac{\Pr_{\mathbf{D}}[d = \tilde{d}]}{\Pr_{\tilde{\mathbf{D}}}[d = \tilde{d}]}) = \sum_{i=1}^{N} \log(\frac{\Pr_{\mathbf{D}}[d_i = \tilde{d}_i | \tilde{d}_{i-1}]}{\Pr_{\tilde{\mathbf{D}}}[d_i = \tilde{d}_i | \tilde{d}_{i-1}]}) \leq \epsilon_1.$$

Because d_i is affected only by d_{i-1} at the previous time point, we have $\Pr_{\mathbf{D}}[d_i = \tilde{d}_i | \tilde{d}_{i-1}] = \Pr_{\tilde{\mathbf{D}}}[d_i = \tilde{d}_i | \tilde{d}_{i-1}, \ldots, \tilde{d}_1]$. Let $S = \{i : \tilde{d}_i \geq \tilde{T}\}$ be the set of indices of $\tilde{d}_i$ at all sampling time points, and $S^C = \{i : \tilde{d}_i \leq \tilde{T}\}$ be the set of indices of $\tilde{d}_i$ at all non-sampling time points, we have $\log(\frac{\Pr_{\mathbf{D}}[d=\tilde{d}]}{\Pr_{\tilde{\mathbf{D}}}[d=\tilde{d}]}) =$
$\sum_{i \in S} \log(\frac{\Pr_{\mathbf{D}}[d_i=\tilde{d}_i | \tilde{d}_{i-1}]}{\Pr_{\tilde{\mathbf{D}}}[d_i=\tilde{d}_i | \tilde{d}_{i-1}]}) + \sum_{i \in S^C} \log(\frac{\Pr_{\mathbf{D}}[d_i=\emptyset | \tilde{d}_{i-1}]}{\Pr_{\tilde{\mathbf{D}}}[d_i=\emptyset | \tilde{d}_{i-1}]})$.

Now we need to bound the two sums respectively. For the first sum, we can see that (1) independent Laplace noise with $Lap(\frac{2C\Delta}{\epsilon_1})$ is added to each distance with $\frac{\epsilon_1}{2C}$ differential privacy, (2) the computation of each L_1 distance needs to access the original histogram once, and (3) $|S| \leq C$ due to the algorithm, so we can obtain the following equation due to sequential composition theorem:

$$\sum_{i\in S}\log(\frac{\Pr_{\mathbf{D}}[d_i=\tilde{d}_i|\tilde{d}_{i-1}]}{\Pr_{\hat{\mathbf{D}}}[d_i=\tilde{d}_i|\tilde{d}_{i-1}]})=\sum_{i\in S}\log(\frac{\Pr_{\mathbf{D}}[\nu_i=\tilde{d}_i-d_i]}{\Pr_{\hat{\mathbf{D}}}[\nu_i=\tilde{d}_i-d_i]})\leq\frac{\epsilon_1}{2}$$

For the second sum, let $A_Z(\mathbf{D})$ be the set of all values of the noise variables $(\nu_1,\dots,\nu_{N-1})$ that cause $\tilde{d}_i\leq\tilde{T}$ for all $i\in S^C$, when the mechanism runs on $\mathbf{D}$, conditioning on $\tilde{T}=Z$ and $d_i=\tilde{d}_i$ for all $i\in S$. Since from $\mathbf{D}$ to $\hat{\mathbf{D}}$, all distances may be increased by at most Δ (i.e. $\Delta=\frac{2}{n-1}$ for the L_1 distance due to lemma 4.2.1), which will cause each distance to remain less than $\tilde{T}$ if we increase $\tilde{T}$ by Δ. But the distances larger than $\tilde{T}$ may become less than $\tilde{T}+\Delta$, so $A_{\tilde{T}+\Delta}(\hat{\mathbf{D}})\subseteq A_{\tilde{T}}(\mathbf{D})\subseteq A_{\tilde{T}+\Delta}(\hat{\mathbf{D}})$. Thus, we have: $\frac{\Pr_{\mathbf{D}}\{\tilde{T}=T+\nu_1\}}{\Pr_{\hat{\mathbf{D}}}\{\tilde{T}=T+\Delta+\nu_2\}}\leq\exp(\frac{\epsilon_1}{2})$. Therefore, let $Z_1=T+\nu_1$ and $Z_2=T+\Delta+\nu_2$, we have:

$$\begin{aligned}
&\prod_{i\in N^C}\Pr_{\mathbf{D}}(d_i=\emptyset|\tilde{d}_{i-1})\\
=&\int_{-\infty}^{\infty}\Pr(\hat{d}=Z_1)\Pr((\nu_1,\dots,\nu_k)\in A_Z(D))dZ\\
\leq&\exp(\frac{\epsilon_1}{2})\int_{-\infty}^{\infty}\Pr(\hat{T}=Z_1)\Pr((\nu_1,\dots,\nu_k)\in A_{Z_1}(D))dZ\\
\leq&\exp(\frac{\epsilon_1}{2})\int_{-\infty}^{\infty}\Pr(\hat{T}=Z_2)\Pr((\nu_1,\dots,\nu_k)\in A_{Z_2}(D'))dZ\\
=&\exp(\frac{\epsilon_1}{2})\int_{-\infty}^{\infty}\Pr(\hat{T}=Z_1)Pr((\nu_1,\dots,\nu_k)\in A_{Z_1}(D'))dZ\\
=&\exp(\frac{\epsilon_1}{2})\prod_{i\in N^C}\Pr_{\hat{\mathbf{D}}}(d_i=\emptyset|\tilde{d}_{i-1})
\end{aligned}$$

Therefore, we have $\frac{\prod_{i\in N^C}\Pr_{\mathbf{D}}(d_i=\emptyset|\tilde{d}_{i-1})}{\prod_{i\in N^C}\Pr_{\hat{\mathbf{D}}}(d_i=\emptyset|\tilde{d}_{i-1})}\leq\frac{\epsilon_1}{2}$ □

THEOREM 4.2.2. Algorithm 7 and 8 preserve ϵ-differential privacy.

PROOF. For Algorithm 7, the decision stage preserves ϵ_1-differential privacy due to theorem 4.2.1. Since releasing at most C DP histograms guarantees ϵ_2-differential privacy, algorithm 7 preserves $\epsilon_1+\epsilon_2=\epsilon$-differential privacy due to theorem 4.1.1. For Algorithm 8, since adaptively adjusting threshold (Line 6 to Line 8) uses no raw data, it does not influence differential privacy guarantee, thus Algorithm 8 guarantees ϵ-differential privacy. $\square$

4.3. Utility Analysis

We analyze the utility of DSFT and DSAT using (α, σ)-usefulness in definition 4.1.2 and show the conclusions in theorem 4.3.1 and 4.3.2. Since we assume LPA as the DP histogram release method, the conclusions can be heuristically used as the upper bound when new methods better than LPA are employed. Here $d(H_i, H_j)$ denotes L_1 distance between H_i and H_j.

Error quantification of DSFT. The utility of released datasets at sampling time points are analyzed based on lemma 4.3.1. The error of datasets at non-sampling time points are obtained via the error bound of the decision stage in lemma 4.3.2.

LEMMA 4.3.1. (Sum of Independent Laplace variables [34]) Suppose that $X_1, \dots, X_n$ are independent Laplace random variables, with each X_i following a $Lap(b_i)$ distribution. Denote $Z = \sum_{i=1}^{n} X_i$ and $b_M = max_i b_i$. Then for all $\gamma \geq \sqrt{\sum_{i=1}^{n} b_i^2}$ and $0 < \lambda < \frac{2\sqrt{2}\gamma^2}{b_M}$, we have $Pr[Z > \lambda] \leq exp(-\frac{\lambda^2}{8\gamma^2})$.

LEMMA 4.3.2. In Algorithm 7, for any $0 < \sigma < 1$, we can obtain

$$Pr\{d(H_i, \tilde{H}_j) \leq T + \frac{4\sqrt{2}\Delta(1 - log\sigma)}{\epsilon_1}\} \geq 1 - \sigma \tag{4.3.1}$$

This means, with probability greater than $1 - \sigma$, we can set t_i as non-sampling time points. (Proof omitted due to space limitation.)

THEOREM 4.3.1. For a range count query covering m histogram bins on $\tilde{H}_k$, and $0 < \sigma < 1$, if k is a sampling time point, we have that $Pr\{|A_k - \tilde{A}_k| \leq -\frac{2\sqrt{2}Clog(\frac{\sigma}{2})}{n\epsilon_2}\} \geq 1 - \sigma$, and if k is a non-sampling time point, we have that $Pr\{|A_k - \tilde{A}_k| \leq T + \frac{4\sqrt{2}\Delta[1-log(1-\sigma)]}{\epsilon_1} - \frac{2\sqrt{2}Clog(\frac{\sigma}{2})}{\epsilon_2}\} \geq (1-\sigma)^2$, where A_k and $\tilde{A}_k$ are the query answers on the original histogram H_k and the DP histogram $\tilde{H}_k$. Therefore, each released histogram $\tilde{H}_k$ of our algorithms maintains (α, σ)-usefulness for range count queries.

Error quantification of DSAT. We analyze the utility of DSAT based on theorem 4.3.2, and give the conclusion as below.

THEOREM 4.3.2. For a range count query covering m histogram bins on $\tilde{H}_k$, and $0 < \sigma < 1$, if k is a sampling time point, the conclusion is the same as theorem 4.3.1 and if k is a non-sampling time point, we have $Pr\{|A_k - \tilde{A}_k| \leq T_k + \sum_{i=1}^{k} I_i u_i - \frac{2\sqrt{2}Clog(\frac{\sigma}{2})}{n\epsilon_2}\} \geq (1-\sigma)^2$, where I_i is a value being 1 or -1, and dependent on the data, and u_i is defined in equation (4.2.3). Therefore, each released histogram $\tilde{H}_k$ of our algorithms maintains (α, σ)-usefulness for range count queries. (The conclusion can be obtained via equation (5) and we omitted the full proof.)

Lower bound of the data cardinality. Since the injected noise in the decision stage is related with data cardinality, we analyze the lower bound of data cardinality to guarantee a relatively small injected noise compared to the true L_1 distance. This lower bound can be used to maintain a high accuracy at the decision stage.

THEOREM 4.3.3. In DSFT, in order to satisfy (α, σ)-usefulness and guarantee the utility of the decision stage, it requires that $n \geq \frac{16\sqrt{2}\alpha(1-log(1-\sigma))}{T\epsilon_1}$, where σ is defined in lemma 4.3.2. (Proof can be deducted from lemma 4.3.2)

Select the value of k in DSFT. Our algorithm requires ϵ to be divided between ϵ_1 and ϵ_2 with $\epsilon_1 = k\epsilon$. We now analyze how to select k. Assume $\mathbf{H} = (H_1, \ldots, H_N)$ corresponds to $\mathbf{D}$. For each i, we analyze the incurred noise variance of L_1 distance between H_i and $\tilde{H}_i$ when i is (1) a sampling time point and (2) a non-sampling time point.

LEMMA 4.3.3. The noise variance of the L_1 distance between H_j and $\tilde{H}_j$, is $\hat{\sigma}_1 = O(\frac{UC^2}{n^2\epsilon_2^2})$ for a sampling time point, and $\hat{\sigma}_2 = O(\frac{8\Delta^2}{\epsilon_1^2} + \frac{32C^2\Delta^2}{\epsilon_1^2} + \frac{UC^2}{n^2\epsilon_2^2})$ for a non-sampling time point.

PROOF. We skip this proof due to space limitation. □

THEOREM 4.3.4. If we use L_1 distance and LPA, the k value can be obtained as $k = min\{\sqrt[3]{\frac{n^2(8\Delta^2+32C^2\Delta^2)}{UC^2}}, 1 - \frac{C}{N}\}$

PROOF. Since k is only used when analyzing the distance at non-sampling time points, we can obtain the upper bound of noise variance at a non-sampling time point due to lemma 4.3.3 with $\epsilon_1 = \frac{k\epsilon}{k+1}$ and $\epsilon_2 = \frac{\epsilon}{k+1}$ by $\hat{\sigma}_2 = \frac{8\Delta^2+32C^2\Delta^2}{\epsilon^2}\frac{(k+1)^2}{k^2} + \frac{UC^2}{n^2\epsilon^2}(k+1)^2$. Let $f(k) = \hat{\sigma}_2$, then the first-order derivative of $f(k)$ is as follows: $\nabla_k f(k) = -\frac{2(k+1)}{k^3}\frac{8\Delta^2+32C^2\Delta^2}{\epsilon^2} + 2(k+1)\frac{UC^2}{n^2\epsilon^2}$. By setting $\nabla_k f(k) = 0$, we can obtain the value of k as: $k = \sqrt[3]{\frac{(8\Delta^2++32C^2\Delta^2)n^2}{UC^2}}$. Since the second-order derivative of $f(k)$ with respect to k is no less than 0, $k = \sqrt[3]{\frac{(8\Delta^2++32C^2\Delta^2)n^2}{UC^2}}$ is the value of k when $f(k)$ arrives at the minimum. Simultaneously, we must require the privacy budget of each sampling time point to be no less than that of each time point in the baseline method, which leads to $\frac{\epsilon_2}{C} \geq \frac{\epsilon}{N}$, and $k \leq 1 - \frac{C}{N}$. Therefore, we can obtain that $k = min\{\sqrt[3]{\frac{n^2(8\Delta^2+32C^2\Delta^2)}{UC^2}}, 1 - \frac{C}{N}\}$. $\square$

]

4.4. Extensions to infinite streams

In this section, we present extensions of DSAT under w-event privacy **DSAT under w-event privacy.** Algorithm 9 presents DSAT under w-event privacy. For the first w time points, we run DSAT normally and record the privacy budget $\epsilon_{2,i}$ for every time point i, i.e. $\epsilon_{2,i} = \frac{\epsilon_2}{C}$ if i is a sampling point and $\epsilon_{2,i} = 0$ otherwise. For time points $w+1$ to N, if the remaining privacy budget ϵ_{rm} for the current w-window is larger than zero, we compare the distance between H_i and $\tilde{H}_j$, modify the threshold and release a private histogram when the private distance is larger than the threshold; if no privacy budget is left, we skip the current time point and go to the next one.

Privacy guarantee. The first w time points guarantees ϵ-differential privacy. The condition in Line 4 of Algorithm 9 guarantees that if there is no remaining privacy budget (i.e. $\epsilon_{rm} \leq 0$) for the current w window from time point t_{i-1} to t_{i-w+1}, no new

Algorithm 9 DSAT under w-event privacy

```
Input:
    $\mathbf{D} = \{D_i | 1 \leq i \leq N, i \in Z\}$, $T$, $C$ and $\epsilon$.
Output:
    $\tilde{\mathbf{D}} = \{\tilde{D}_i | 1 \leq i \leq N, i \in Z\}$
 1: Run DSAT for the first w time points;
 2: for $i = (w+1)$ to $N$ do
 3:     $\epsilon_{rm} = \epsilon_2 - \sum_{m=i-w+1}^{m=i-1} \epsilon_{2,m}$
 4:     if $\epsilon_{rm} \leq 0$ then
 5:         Set $\tilde{D}_i := D_j$, where $j$ is the time point of last release;
 6:     else
 7:         Set $count = \frac{\sum_{m=i-w+1}^{m=i-1} \epsilon_{2,m}}{\epsilon_2/C}$
 8:         Compute $\tilde{d}(D_i, \tilde{D}_j) = d(D_i, \tilde{D}_j) + Lap(\frac{2C\Delta}{\epsilon_1})$;
 9:         Compute $E_i = |\frac{count}{i} - \frac{C}{w}|$, $e_i = \frac{|E_i - \delta|}{\delta}$, and $u_i = \theta \times e_i$;
10:         if $\frac{count}{i} - \frac{C}{w} \leq 0$, then $\tilde{T}_i = max\{0, T_{i-1} - u_i\}$;
11:         else set $\tilde{T}_i = min\{2, T_{i-1} + u_i\}$;
12:         if $\tilde{d}(D_i, \tilde{D}_j) \geq \tilde{T}_i$, then set $\epsilon_{2,i} = \epsilon_2/C$, $\tilde{D}_i := D_i + < Lap(1/\epsilon_{2,i}) >^U$, and
            $j = i$;
13:         else set $\tilde{D}_i := D_j$;
14:     end if
15: end for
```

private datasets will be released. Therefore, for any w-length window beginning with any time point, at most ϵ privacy budget will be used. This leads to the conclusion that Algorithm 9 satisfies w-event privacy.

4.5. Experiment

We implemented our methods on top of two static histogram methods, LPA in Matlab and PSD [44] in Python. All the experiments are performed on a PC with a 2.9GHz CPU and a 8GB memory. Table 4.2 summarizes the parameters and their default values in the experiments.

4.5.1. Experiment Setup.

Datasets. We conducted our experiments with three datasets: the US census (http://ipums.org), the Taxi-Drive trajectory data (http://research.microsoft.com/apps/) and the Oldenburg traffic data [30].

Table 4.2. Experiment Parameters

Parameter	Description	Default value
N	Number of time points	500
d	Number of data dimensions	6
n	Number of tuples in D_i	500K
ϵ	Privacy budget	1.0
C	Cutoff point	$0.01 \times N$
r	Update rate	0.5
δ	Deviation tolerance	0.05
θ	Proportional gain	0.5

The **US census dataset** contains six attributes, *Age*, *Gender*, *Education*, *Health insurance*, *Marital status* and *Income* with 3M tuples and domain sizes of 96, 2, 12, 2, 2, 3. Each tuple represents an individual user. In order to avoid the sparsity of histograms, we convert *Income* into a categorical attribute: values smaller than 0 (mapped to 1), values between 0 and 28K (mapped to 2), and values larger than 28K (mapped to 3). 28K is a median value. Values smaller than 0 means the tuples have ages smaller than 20. The number of histogram bins are the product of the domain sizes of all attributes.

We generate a series of dynamic datasets as follows. D_i is the original dataset at t_i. D_1 has 500K tuples randomly sampled from the original 3M tuples. A public pool is initiated using the remaining tuples. D_i ($i \geq 2$) is obtained by deleting m tuples from D_{i-1} while inserting m tuples randomly selected from the public pool to simulate the user updates. m is sampled from $N(\mu, \sigma^2)$, where μ is $\frac{r \times |D_{i-1}|}{2}$, and σ^2 is set to $100K$. Here, r is the *update rate*, $|D_i|$ is the data cardinality of D_i and datasets at all time points have the same data cardinality. The time points are partitioned into 10 periods with different values of m to simulate varying update patterns. All experiments use US census data by default since we can generate various datasets under different parameter settings.

The **Taxi trajectory dataset** has a one_week trajectories of $10,357$ taxis during the period of Feb. 2 to Feb. 8, 2008 within Beijing. We transfer the time dimension to 168 time points with 24×7. The total number of points in this dataset is about 15 million and the total distance of the trajectories reaches 9 million kilometers. We partition the longitude and latitude into 10×10 grids. We amplify the number of taxis to $110,357$ by sampling dummy points on extremely sparse time points and geographical areas while still keeping the patterns of original data.

We generated **Oldenburg traffic data** with the Brinkhoff generator [30]. The input of the generator is the road map of Oldenburg in Germany, and the output is a set of moving objects on the road network. We created the data set with 1000 discrete timestamps, with 500,000 objects at the beginning. A 2D grid with 1024×1024 cells is used to record the locations of the moving objects.

Comparison. We evaluate the utility of the private DP histograms of dynamic datasets by answering random range count queries. The query accuracy of DSAT is compared with three solutions described in Section 3: the baseline Laplace mechanism, the fixed-sampling method, and the state-of-the-art w-event privacy methods. LPA and PSD [44] are used to generate DP histograms at sampling time points. We note that our proposed sampling framework can utilize any state-of-the-art static histogram method at each sampling point. Here we just use, as an example, the standard LPA method as well as the PSD method [44] which is a state-of-the-art static histogram method that uses spatial partitioning. The goal is to compare our proposed methods and the three solutions. We also include the non-private methods to compare the update errors of DSAT and fixed-sampling.

Metrics. For the US census dataset, we generated random range-count queries with random query predicates on each attribute defined in the SQL format as "Select COUNT(*) from D, Where $A_1 \in I_1$ and $A_2 \in I_2$ and...and $A_m \in I_m$". I_i is a random interval generated from the domain of attribute A_i. For the traffic data, query rectangles with various sizes are randomly generated. In each experiment run, 5000 random queries are generated and the average absolute error over 10 runs is reported, which is defined as $E^a = \frac{1}{M \times N} \sum_{k=1}^{M} \sum_{i=1}^{N} |\tilde{A}_i^k - A_i^k|$, A_i^k is the true answer and $\tilde{A}_i^k$ is the noisy answer. Here we use the range-count query to measure the utility since it composes data histograms, and the range counts can be used for many significant mining tasks, e.g. dynamic stream clustering, outlier detection of time-series data, etc.

4.5.2. Results on user level privacy. In all experiments, we compare our methods with the baseline and fixed-sampling methods, which are denoted by "baseline" and "fixed" in figures. Unless specified, we use LPA by default as the underlying histogram method for the sampling point. We also use "DSAT-true" and "fixed-true" to denote the non-private versions of DSAT and fixed-sampling.

Absolute error vs. k. Figure 4.2 investigates how utility changes with various k values, which specify the budget allocation ratio between ϵ_1 and ϵ_2 for the decision and sampling stages respectively. With the value of C being 10, we compute k to be 0.0532 due to theorem 4.3.4. From Figure 4.2, we can observe that the empirical result matches the theoretical result well and the utility reaches the optimal value with k between 0.01 and 0.1. The error increases as k becomes larger or smaller than 0.1 or or 0.01, respectively. This is reasonable because larger k may lead to more perturbation error while smaller k values result in more update error.

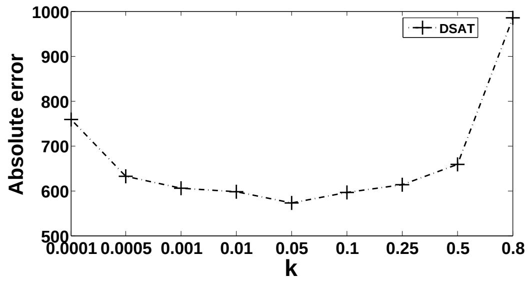


FIGURE 4.2. Absolute error vs. k

DSFT and DSAT. In this experiment, we compare our proposed two methods DSFT and DSAT. From figure 4.3, we can observe that the error of DSFT is very sensitive to the threshold value T. As T initially increases, the error decreases thanks to the decreased perturbation error. As T further increases, the error increases back up due to the increased sampling error which becomes the dominant error. Without prior knowledge, it is difficult to determine the optimal T. However, the average absolute error of DSAT is close to the lowest error of DSFT with the optimal threshold T value being around 0.025. Here the initial value of T for DSAT can be arbitrarily selected. Thus, the DSAT method with the PID control can effectively adjust T to an optimal one. In the remaining experiments, we only use DSAT to compare with other methods.

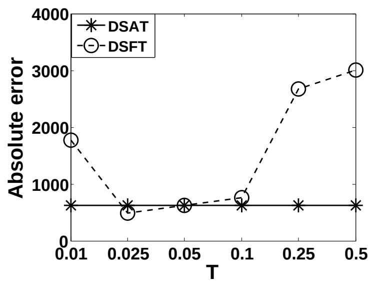
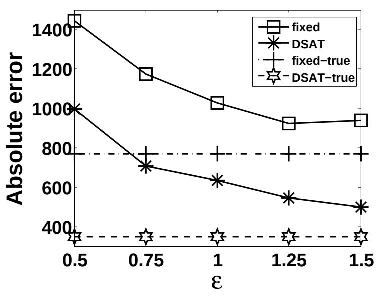


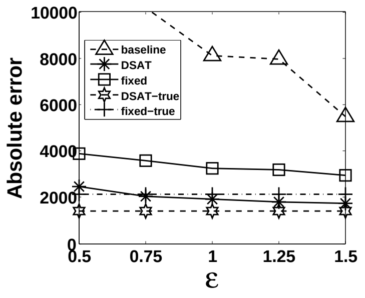
FIGURE 4.3. Absolute error vs. threshold value T

Absolute error vs. differential privacy. Figure 4.4 compares DSAT with other methods under various privacy budgets. The larger the privacy budget is, the closer the query accuracy is to non-private versions. Since the baseline performs one order of magnitude worse than other methods in most experiments, we do not include them for better readability of the graphs. The perturbation errors for fixed-sampling and DSAT are almost similar as the number of released DP histograms are the same. DSAT outperforms fixed-sampling because DSAT has much less update error, which can be seen from the comparison of non-private versions. Figure 4.4(b) uses the taxi trajectory dataset and Figure 4.4(c) uses PSD to release DP histograms with 3D US data. We can see that by using PSD, errors are generally improved compared to the ones using LPA. This further confirms that our methods can take advantage of any state-of-the-art static histogram methods for each sampling point.

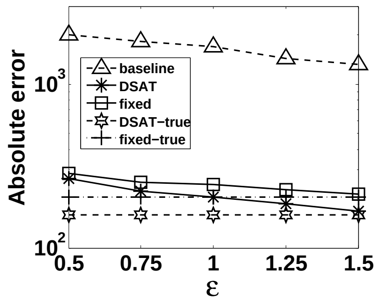
Absolute error vs. update rate. We study the impact of the update rate r (defined in section 4.5.1) on the query accuracy for different methods, as shown in Figure 4.5. All methods remain stable for various update rates. The DSAT performs better than both non-private and private fixed-sampling methods. This is because



(a) US data



(b) Taxi-drive data



(c) US data (log-scale) with PSD

FIGURE 4.4. Accuracy vs. differential privacy budget

the update error of non-private DSAT is much less than non-private fixed-sampling. This further verifies that our DSAT with PID controller succeeds in adaptively adjusting the threshold and the location of the sampling time point, leading to better performance.

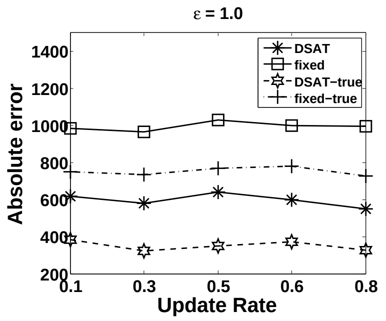


FIGURE 4.5. Absolute error vs. update rate

Absolute error vs. dimensionality. Figure 4.6 examines the absolute error with various numbers of dimensions in the US dataset. DSAT again outperforms both non-private and private fixed-sampling methods with the dimensionality from 3 to 6. One interesting phenomena we observe is that the performances of non-private and private fixed-sampling methods improve sharply after five dimensions. This can be explained by the fact that a higher dimensionality results in a larger number of histogram bins. Given a threshold T, if the L_1 distance between two datasets D_i and D_{i-1} is below T, the previously released histogram will be used which incurs an update error. Given the same L_1 distance between two histograms, a larger number of bins would result in a smaller measured update error since the average difference for each histogram bin is smaller. Hence the fixed sampling methods show a dramatic drop in the error which is dominated by the update error. The DSAT methods are less sensitive to the number of dimensions because they already mitigate the update error by tuning the threshold adaptively. Hence the non-private DSAT shows a slight drop in the update error while the private DSAT shows a slight increase due to the dominating perturbation error.

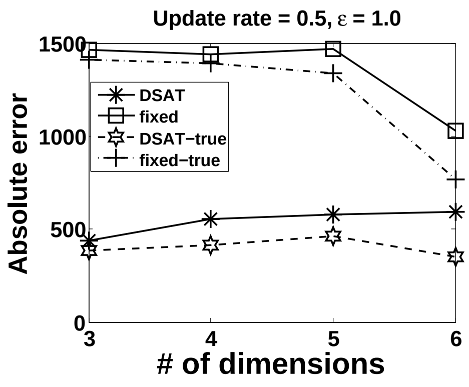
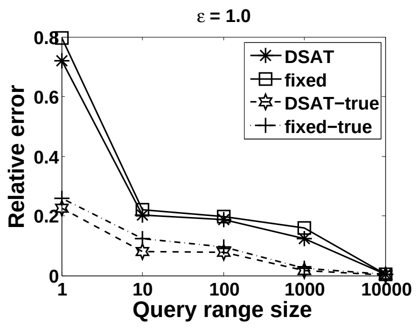
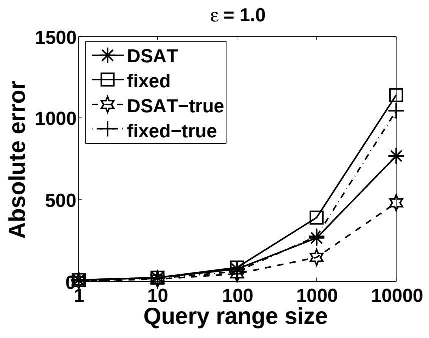


FIGURE 4.6. Absolute error vs. # of dimensions

Query accuracy vs. query range size. We study the impact of the query range size on the query accuracy for different methods. For each query range size, we randomly generated queries such that the product of the query ranges on each dimension equals the given size. Figure 4.7 presents the impact of various query range sizes on query accuracy in terms of relative error and absolute error. The relative error is defined as $E^r = \frac{1}{M \times N} \sum_{k=1}^{M} \sum_{i=1}^{N} \frac{|\tilde{A}_i^k - A_i^k|}{max(s, A_i^k)}$, where s is the sanity bound to mitigate the effect for $A_i^k = 0$. DSAT outperforms the private fixed-sampling method. The difference of relative errors between all methods is not obvious because of the large data cardinality in the US data. For all methods, the relative error gradually degrades as the query range size increases while the absolute error has the opposite trend. The reason is that when the query size is small, the true answer A_i^k is also small which may incur a small absolute error but large relative error. In this experiment, the sanity bound s is set to 1.



(a) Relative error

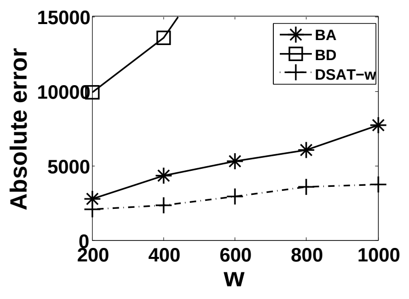


(b) Absolute error

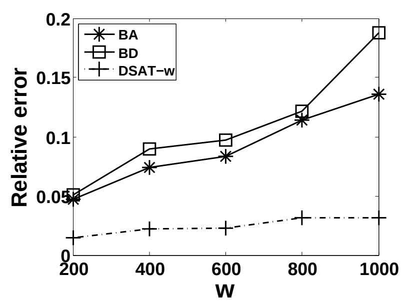
FIGURE 4.7. Query accuracy vs. query range size

4.5.3. Results on w-event privacy. Query accuracy vs. parameter w. We use the Oldenburg traffic data in this experiment, since it contains 1000 timestamps that is sufficient to investigate the impact of w. We compare DSAT with BD and BA in [39] under w-event privacy framework while varying w values. BD and BA are implemented by using column partitioning technique and setting $\epsilon_1 = \frac{\epsilon}{U}$ as recommended in [39]. From figure 4.8, we can see that the gap between DSAT and BD or BA expands greatly as w increases. This is because our technique adaptively adjusts the threshold and allocates the privacy budget more appropriately. In contrast, BA and BD may not fully utilize or in advance exhaust most budget during w timestamps.

Query accuracy vs. differential privacy. In this experiment, we set w to be 800 using Oldenburg traffic data with 1000 timestamps. Figure 4.9 compares DSAT with BA and BD under various privacy budgets. We can see that BA degrades



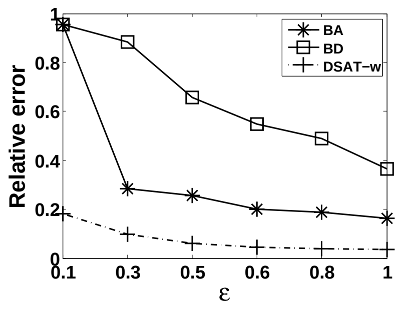
(a) Absolute error



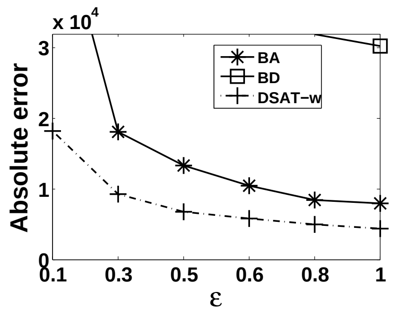
(b) Relative error

FIGURE 4.8. Query accuracy vs. w

dramatically and the gap between BA and DSAT greatly expands as we reduce the privacy budget ϵ. This is because BA starts by uniformly distributing the budget to all w timestamps, and more perturbation error will be incurred when ϵ is small and w is large. Our DSAT performs well since the perturbation error of released datasets depends only on C.



(a) Relative error



(b) Absolute error

FIGURE 4.9. Query accuracy vs. differential privacy

4.6. Conclusions

In this chapter, we have proposed an adaptive distance-based sampling approach to address the challenges of releasing a series of differentially private dynamic datasets in real time. With an upper bound to limit the number of DP data releases, our methods incur much smaller errors. We apply an adaptive control mechanism to dynamically adjust the threshold value. We also provide privacy and utility analysis for our method. Experiments on real and synthetic datasets show that our algorithm outperforms the baseline and existing state-of-the-art techniques. As future work, we would like to study update models and incorporate them into our sampling framework. We are also interested in applying the adaptive sampling framework for releasing other types of dynamic data with differential privacy, e.g. frequent patterns for dynamically changing transactional data and dynamic graph patterns in social networks.

CHAPTER 5

Personalized differential privacy

Previous chapters assume that all records of a dataset have the same privacy preference under differential privacy. A limitation of the privacy model is that the same level of privacy protection is assigned for all individuals. However, it is common that data subjects have different expectations regarding the acceptable level of privacy for their data. Consequently, differential privacy may lead to insufficient privacy protection for some users, while over-protecting others. In this chapter, we propose two partitioning-based mechanisms, privacy-aware partitioning and utility-based partitioning, for personalized differential privacy [60]. The first privacy-aware partitioning mechanism aims to minimize the waste of privacy budgets independent of applications, while utility-based partitioning is to maximize the utility for a given application. Extensive experiments using real datasets demonstrate that our partitioning mechanisms have advantages over existing methods.

5.1. Preliminaries

Personalized differential privacy. Personalized differential privacy is a setting where the privacy budget values over all data records are not uniform. That means, data owners may independently customize privacy budgets of their records and set them based on their subjective judgements. Thus, we assign a privacy budget to each record independently from other records and any of its sensitive attribute values. Notice that the value of the privacy budget should not be correlated with any sensitive information, which would make it also sensitive. In our example (Figure 5.1), a sensitive attribute Salary is not correlated with the privacy budget. One naive baseline mechanism is the Minimum mechanism, which uses the minimal budget among

Name	**Age**	**Zip**	**Salary**	**Budget** α_i
Alice	22	02152	70000	0.01
Emily	32	02112	180000	0.02
John	31	02130	105000	0.05
Olga	27	02114	110000	0.07
Frank	36	02232	90000	0.09
Bob	35	01245	140000	0.11
Mark	33	04323	110000	0.14
Cecilia	39	02121	100000	0.15

FIGURE 5.1. Personalized privacy dataset

records, e.g., 0.01 for Figure 5.1. Although Minimum mechanism preserves PDP, it cannot take advantage of personalized privacy preferences. If we have a dataset where there are relatively few data owners who set their privacy requirements to be the minimum, a large amount of privacy budgets would be wasted while the utility will be significantly reduced.

Baseline mechanisms. The first baseline mechanism *Minimum* is simply using the smallest privacy budget ϵ_{min} in a given privacy vector $V = (\epsilon_1, \dots, \epsilon_n)$, and then invoke a differentially private algorithm using ϵ_{min} as the global privacy parameter and all data records. Although *Minimum* satisfies personalized differential privacy, it gains no benefit from various privacy budget values, and most record owners will receive a much stronger level of privacy guarantee than they prefer. The second baseline mechanism *Threshold* is that we select a privacy threshold due to the distribution of number of users with different privacy levels. However, an optimal threshold would be difficult to choose if we want to make best of all privacy budget while remain a high utility.

Sampling mechanism. Jorgensen et al. [25] considered the personalized differential privacy (PDP) setting in which each user can independently specifies their own privacy budget value for their data record. They proposed two mechanisms for achieving PDP: the sampling mechanism and the personalized exponential mechanism. The main goal is to develop mechanisms that can take advantage of various privacy budgets to attain better utility than could be achieved with basic differential privacy mechanisms. The sampling mechanism is a two-step sampling based method defined in Definition 5.1.1.

Definition 5.1.1 (The Sampling Mechanism [25])**.**
Consider a function $f : D \to R$, a dataset D containing n records corresponding to n individual data owners, and a privacy preference vector $\phi = (\epsilon_1, \ldots, \epsilon_n)$. Let $RS(D, \phi, \epsilon_T)$ denote a procedure that independently and randomly samples each record $x_j \in D$ $(1 \leq j \leq n)$ with probability

$$p_j = \begin{cases} \dfrac{e^{\epsilon_j} - 1}{e^{\epsilon_T} - 1} & if \quad \epsilon_j < \epsilon_T \\ 1 & otherwise \end{cases}$$

where $\epsilon_{min} \leq \epsilon_T \leq \epsilon_{max}$ is a predefined threshold. The sampling mechanism is defined as

$$S(D, \phi, \epsilon_T) = DP^f_{\epsilon_T}(RS(D, \phi, \epsilon_T))$$

where $DP^f_{\epsilon_T}$ is any ϵ_T-differentially private mechanism for the function f.

It involves a non-uniform sampling step at the individual record level, followed by the invocation of an appropriate differentially private mechanism on the sampled dataset. In the sampling step, the inclusion probabilities for each record are computed due to the individual privacy budget of the corresponding user and the privacy budget threshold. The sampling mechanism is general and can be used to easily convert any existing differentially private algorithm into a PDP algorithm, but an appropriate sampling threshold needs to be selected and predefined, which may be difficult in practice. For different number of data records and learning problems, the rule of choosing an optimal threshold value may not be the same. For example, as shown in our experiments, choosing the mean of privacy budgets as the threshold performs the best for count queries. However, for logistic regression, using the maximum privacy budget leads to the best utility. In support vector machine, there is no obvious rule on how to chose an optimal threshold.

Personalized exponential mechanism. The second mechanism in [25] is a personalized exponential mechanism, developed based on the exponential mechanism, where the utility function is designed to satisfy PDP particularly. The mechanism is applicable to only simple aggregates such as counts, medians, because the utility function is difficult to design and develop for complicated applications, like logistic regression and support vector machine.

5.2. Partitioning mechanisms

In this section, we propose two partitioning mechanisms to fully utilize the privacy budget of individuals and maximizing the utility of target DP computations. The general partitioning mechanism includes: (1) partition records of D horizontally into k groups $(D_1, \ldots, D_k)$ due to various privacy budgets; (2) compute noisy output q_i of target aggregate mechtianism M for each D_i with ϵ_i-differential privacy, and (3) ensemble $(q_1, \ldots, q_k)$ to compute q. We define the general partitioning mechanism as below:

Definition 5.2.1 (The General Partitioning Mechanism)**.** For an aggregate function $f : D \to R$, a dataset D with n records of n individual users, and a privacy preference $\phi = (\epsilon_1, \ldots, \epsilon_n)$ $(\epsilon_1 \leq \ldots \leq \epsilon_n)$. Let $Partition(D, \phi, k)$ be a procedure that partitions the original dataset D into k partitions $(D_1, \ldots, D_k)$. The partitioning mechanism is defined as $PM = B(DP^f_{\epsilon_1}(D_1), \ldots, DP^f_{\epsilon_k}(D_k))$ where $DP^f_{\epsilon_i}$ is any target ϵ_i-differentially private aggregate mechanism for f, B is an ensemble algorithm.

The partitioning mechanisms have no privacy risk because it is computed directly from public information, privacy budget of each record. The target aggregate mechanism guarantees ϵ_i-DP for each partition, with ϵ_i as the minimum privacy parameter value of the records in that partition.

5.2.1. Privacy-aware partitioning mechanism.

We develop privacy-aware partitioning mechanism with the goal of grouping records with similar privacy budgets, such that the amount of wasted budget is minimized. Formally, we formulate the privacy budget waste of a partition D_i as $W_i = W(\epsilon_{i,1}, \ldots, \epsilon_{i,n_i}) = \sum_{j=1}^{n_i}(\epsilon_{i,j} -$

$min(\epsilon_{i,j}))^2$, where n_i is number of records in D_i, $\epsilon_{i,j}$ is the privacy budget of jth-record of D_i, and $min(\epsilon_{i,j})$ ensures ϵ_i-DP for D_i. We define privacy-aware partitioning algorithm as follows:

Definition 5.2.2 (Privacy-aware partitioning)**.** In a sorted privacy budget vector $\phi = (\epsilon_1, \ldots, \epsilon_n)$, where $\epsilon_1 \leq \epsilon_2 \leq \ldots \leq \epsilon_n$, we want to split ϕ into k partitions such that $W(\phi) = \sum_{i=1}^{k} W_i$ is minimized, where $W_i = \sum_{j=1}^{n_i}(\epsilon_{i,j} - min(\epsilon_{i,j}))^2$.

With a predefined k, we find the optimal k-partitioning using dynamic programming and present the privacy-aware partitioning algorithm in Algorithm 10.

Algorithm 10 Privacy-aware partitioning mechanism $W*$ of finding the optimal k-partition of $(\epsilon_1, \ldots, \epsilon_n)$ for a given definition of the function W

```
Input: Sorted $\phi = (\epsilon_1, \ldots, \epsilon_n)$ and $k$
Output: $k$ partitions of original dataset
  1. if $k = 0$ then return 0
  2. $minW = \text{inf}$
  3. foreach $j \in \{k-1, \ldots, n\}$ do
       $currentW = W^*((\epsilon_1, \ldots, \epsilon_j), k-1) + W(\epsilon_{j+1}, \ldots, \epsilon_n)$
       if $currentW < minW$ then
           $minW = currentW$
           $partitions[k-1] = (\epsilon_{j+1}, \ldots, \epsilon_n)$
  4. return $minW$ and indexes of k partitions
```

Before running Algorithm 10, we first sort all privacy budgets in ascending order. Sorting records in the descending order of privacy budgets generates the same partition. When we sort privacy budgets, the sequence of corresponding data records follows the order of privacy budgets. Therefore, we know which records are included in which partition. To simplify the algorithm, we do not include representation of data records. In step 3, we use dynamic programming to find the optimal partition for a given definition of the function W. The goal is to minimize the waste of privacy budgets in each partition by computing the distance between individual budget and the minimum budget of the current partition. Note that we represent Algorithm 10 as W^*, and $currentW = W^*((\epsilon_1, \ldots, \epsilon_j), k-1) + W(\epsilon_{j+1}, \ldots, \epsilon_n)$ means that we recursively use Algorithm 10 to compute $k-1$ partitions.

Optimal number of partitions. Algorithm 10 finds an optimal k-partitioning given a predefined k. To choose an optimal k, let us consider two extreme cases: (i) we can have n partitions where each record is its own partition and no privacy budget is wasted, or (ii) all data records can be grouped as one partition to maximize the number of records in the partition. The amount of generated noise could be significant in the previous case, while large amount of privacy budget waste may be incurred in the latter case. We need to consider the tradeoff between n and ϵ to find the optimal k by building the following objective function:

$$\min_{k} \sum_{i=1}^{k} [\frac{1}{n_i} \sum_{j=1}^{n_i} (\epsilon_{i,j} - min(\epsilon_{i,j}))^2] \tag{5.2.3}$$

Equation (5.2.3) implies a tradeoff between the partition size and privacy budget waste. Due to equation (5.2.3), neither extreme case (i) nor (ii) can lead to optimal value of equation (5.2.3). If we set a minimum threshold T of partition size n_i for the target differentially private mechanism, we can search different number of partitions from 1 to $\frac{n}{T}$, and find the optimal partition number. The minimum number of records n_i required in one partition is reasonable because many aggregate mechanisms (e.g. logistic regression, support vector machine) require a minimum training data size to ensure acceptable performance, due to machine learning theory. For example, Shalev-Shwartz et al. [53] show that for a given classifier with expected loss defined on a differentiable loss function, the excess loss of the classifier will be upper bounded if training data size is larger than a threshold.

Complexity. Sorting all privacy budgets takes $O(n \log n)$. Computing optimal k takes $O(n)$, since we need to scan privacy vector at most $m = \frac{n}{T}$ times ($\frac{n}{T}$ is constant here since we control T to make $\frac{n}{T}$ constant for complexity reduction). The privacy-aware partitioning takes $O(mnlogn)$ complexity using dynamic programming with intermediate results saved and optimization tricks. The overall complexity is $O(mnlogn)$.

5.2.2. Utility-based partitioning mechanism. The privacy-aware partitioning mechanism aims to fully utilize the privacy budget of individual users which will indirectly optimize the utility of the target DP computation. In this section, we present a utility-based partitioning mechanism explicitly optimized for target DP computations. The utility-based partitioning is inspired by an observation that many DP machine learning algorithms (e.g. [3, 54, 63, 61, 55], etc.) have their performance related with n, ϵ for a dataset of n records with ϵ-DP. We give definition of utility-based partitioning below.

Definition 5.2.4 (Utility-based partitioning)**.** In a sorted privacy budget vector $\phi = (\epsilon_1, \ldots, \epsilon_n)$, where $\epsilon_1 \leq \epsilon_2 \leq \ldots \leq \epsilon_n$, and let n_i denote the number of records in D_i, we want to split ϕ into k partitions to maximize $\sum_{j=1}^{n_i} U(n_i, min(\epsilon_{i,j}))$, where $U(n_i, min(\epsilon_{i,j}))$ is a utility function of target DP computation, which is related with n_i and $min(\epsilon_{i,j})$.

Algorithm 11 Utility-based partitioning mechanism U^* of finding the optimal k-partition of $(\epsilon_1, \ldots, \epsilon_n)$ for a given definition of utility function U

```
Input: $(\epsilon_1, \ldots, \epsilon_n)$ and $k$
Output: $k$ partitions of original data records
  1. if $k = 0$ then return $U(n, \epsilon_{min})$
  2. $maxUtility = 0$
  3. foreach $j \in \{k-1, \ldots, n\}$ do
       $currentUtility = U^*((\epsilon_1, \ldots, \epsilon_j), k-1) + U(min(\epsilon_{j+1}, \ldots, \epsilon_n), n-j)$
       if $currentUtlity > maxUtility$ then
           $maxUtility = currentUtility$, $partitions[k-1] = (\epsilon_{j+1}, \ldots, \epsilon_n)$
  4. return $maxUtility$
```

Algorithm 11 presents the utility-based partitioning. We observe that $U(n, \epsilon)$ can be considered as a general utility form in a series of existing state-of-the-art DP algorithms (e.g. [58, 50, 56, 57, 8, 17, 65, 64, 62, 66], etc.). (i) *Count query* In the Laplace mechanism, the noisy result of a function f can be represented as $f(D) + \nu$, where ν follows $Lap(\frac{\Delta_f}{\epsilon})$, and Δ_f is the sensitivity related to number of records n. If we normalize $f(D)$ by n, Δ_f would become $\frac{\Delta_f}{n}$. Thus, the variance of Laplace distribution can be considered as the utility function $U(n, \epsilon) = 2(\frac{\Delta_f}{n\epsilon})^2$. Maximizing $n\epsilon$ will lead to best utility with a high probability. (ii) *Empirical risk minimization.* We take for example the DP empirical risk minimization mechanism

(DPERM) proposed by Chaudhuri et al. [58]. The reason is that DPERM can be easily generalized to important machine learning tasks, such as logistic regression and support vector machine, which have a convex loss function as the optimization objective. Our utility function form can be extended to a class of DP machine learning mechanisms.

Assume that n records in a dataset D are drawn i.i.d. from a fixed distribution $F(X,y)$. Given F, the performance of privacy preserving empirical risk minimization algorithms in [58] can be measured by the expected loss $L(f)$ for a classifier f, defined as $L(f) = E_{(X,y)\ F}[l(f^T x, y)]$, where the loss function l is differentiable and continuous, the derivative l' is c-Lipschitz. By [58], the expected loss of the private classifier f_p can be bounded as below

$$L(f_p) \leq L(f_0) + \frac{16||f_0||^4 d^2 \log^2(d/\sigma)(c + e_g/||f_0||^2)}{n^2 e_g^2 \epsilon^2} + O(||f_0||^2 \frac{log(1/\sigma)}{ne_g}) + \frac{e_g}{2} \tag{5.2.5}$$

where $L(f_0)$ is the expected loss of the true classifier f_0, ϵ is the privacy budget, e_g is the generalization error, and d is the number of dimensions of input data. If we consider the second part of equation (5.2.5), we can build a utility function as $U(n,\epsilon) = \frac{16||f_0||^4 d^2 \log^2(d/\sigma)(c+e_g/||f_0||^2)}{n^2 e_g^2 \epsilon^2} + ||f_0||^2 \frac{log(1/\sigma)}{ne_g} + \frac{e_g}{2}$, where only n and ϵ are variables.

Optimal number of partitions. Akin to privacy-aware partitioning mechanism, we need to select an optimal value for k, in order to maximize the sum of utility function value over all partitions.

$$\max_k \sum_{i=1}^{k} U(n_i, \min_{1\leq j\leq n_i}(\epsilon_{i,j})) \tag{5.2.6}$$

Here, a minimum threshold T of each partition size is also required for a differentially private task. Theoretically, we can search different number of partitions from 1 to $\frac{n}{T}$ to find the optimal number of partitions with the maximum value of objective function (5.2.6).

Complexity. Sorting all privacy budgets is $O(n \log n)$. Finding the optimal partitioning takes $O(n)$, due to complexity of Algorithm 10. The utility-based partitioning takes $O(n)$. The overall complexity of Algorithm 11 is $O(n \log n)$.

5.2.3. T-round partitioning.

After the first round of partitioning, we may still have records with remaining budgets. Extra rounds of partitioning can be applied iteratively on the remaining records with leftover privacy budgets. In this part, we prove by iteratively apply our algorithm to the leftover budget from previous iterations, the leftover budget will decrease exponentially, which means all input budgets will be used up soon.

Here we define a T-round partitioning as iteratively grouping n records into k partitions according to the objective function in Definition 3, then consume the smallest budget in each group and update the leftover budget. The leftover budget for the l-th record in the t-th round is denoted as ϵ_l^t.

Theorem: $\sum_{l=1}^{n}(\epsilon_l^T)^2 \leq \left(\frac{n}{n-1+k^2}\right)^T \sum_{l=1}^{n}(\epsilon_l)^2$, which means the leftover privacy budget converges to 0 exponentially.

Proof. Without loss of generality, we assume ϵ_n is the largest among all input privacy budgets, and select the partition that partitions the interval $[0, \epsilon_n]$ into k intervals with equal length ϵ_n/k. In this case, for the leftover budget ϵ_l^{1*} we have $\epsilon_l^{1*} \leq \epsilon_n/k, \epsilon_n^{1*} = \epsilon_n/k$ for all $1 \leq l \leq n$. Thus $\sum_{l=1}^{n}(\epsilon_l^{1*})^2 \leq \sum_{l=1}^{n}(\epsilon_n/k)^2 = n(\epsilon_n/k)^2$. Furthermore, since we have $\epsilon_l \geq \epsilon_l^{1*}$, there is $\sum_{l=1}^{n}(\epsilon_l)^2 - \sum_{l=1}^{n}(\epsilon_l^{1*})^2 \geq \sum_{l=1}^{n}[(\epsilon_l)^2 - (\epsilon_l^{1*})^2] \geq (\epsilon_n)^2 - (\epsilon_n^{1*})^2 = (\epsilon_n)^2\left(1 - \frac{1}{k^2}\right)$. Combining them together, we conclude $\frac{\sum_{l=1}^{n}(\epsilon_l)^2}{\sum_{l=1}^{n}(\epsilon_l^{1*})^2} = \frac{\sum_{l=1}^{n}(\epsilon_l)^2 - \sum_{l=1}^{n}(\epsilon_l^{1*})^2}{\sum_{l=1}^{n}(\epsilon_l^{1*})^2} + 1 \geq \frac{(\epsilon_n)^2\left(1-\frac{1}{k^2}\right)}{n(\epsilon_n/k)^2} + 1 = \frac{k^2-1}{n} + 1 \frac{\sum_{l=1}^{n}(\epsilon_l^{1*})^2}{\sum_{l=1}^{n}(\epsilon_l)^2} \leq \frac{n}{k^2-1+n}$. Since the optimal partition must have smaller $\sum_{l=1}^{n}(\epsilon_l^1)^2$ than this very naive partition, there must be $\frac{\sum_{l=1}^{n}(\epsilon_l^1)^2}{\sum_{l=1}^{n}(\epsilon_l)^2} \leq \frac{n}{k^2-1+n}$. Similarly, if we take ϵ_l^1 as input to the next round, we can get $\frac{\sum_{l=1}^{n}(\epsilon_l^2)^2}{\sum_{l=1}^{n}(\epsilon_l^1)^2} \leq \frac{n}{k^2-1+n}$, etc. When we multiply these inequalities together, we conclude $\sum_{l=1}^{n}(\epsilon_l^T)^2 \leq \left(\frac{n}{n-1+k^2}\right)^T \sum_{l=1}^{n}(\epsilon_l)^2$. □

5.2.4. Ensemble.

Once we have partitions, we run DP mechanism on each partition, and then use ensemble methods to aggregate the result from each partition.

Due to conclusions of [59], our ensemble rule is that the private output of partition with equal number of records but smaller privacy budgets than other partitions would be dropped out. We also consider types of learning problems. For numerical situation, like bagging multiple linear regression or count queries, we aggregate all private predicted values from all partitions. The weights will depend on $O(n_i, \epsilon_i)$. Assume the numerical task is P, the aggregated result would be $\tilde{Y} = \sum_{i=1}^{k} w_i P(D_i)$. For classification tasks, we use majority voting.

5.3. Experiment

In this section, we experimentally evaluate partitioning-based mechanisms and compare it with the sampling mechanism in [25]. Partitioning-based mechanisms are implemented in MATLAB R2010b and Java, and all experiments were performed on a PC with 2.8GHz CPU and 8G RAM.

5.3.1. Experiment Setup.

Datasets. We use two datasets from the Integrated Public Use Microdata Series[1], US and Brazil, which contain 370, 000 and 190,000 census records collected in the US and Brazil, respectively. There are 13 attributes in each datasets, namely, Age, Gender, Martial Status, Education, Disability, Nativity, Working Hours per Week, Number of Years Residing in the Current Location, Ownership of Dwelling, Family Size, Number of Children, Number of Automobiles, and Annual Income. Among these attributes, Marital status is the only categorical attribute whose domain contains more than 2 values, i.e., Single, Married, and Divorced/Widowed. Following common practice in regression analysis, we transform Marital Status into two binary attributes, Is Single and Is Married (an individual divorced or widowed would have false on both of these attributes). With this transformation, both of our datasets become 14 dimensional.

[1]Minnesota Population Center. Integrated public use microdata series-international: Version 5.0. 2009. https://international.ipums.org.

We perform 10-fold cross-validation for logistic regression and support vector machine with 50 times, and report the average results. We select two subsets of the attributes in each dataset for count and support vector machine classification. The first subset contains 4 attributes: Age, Gender, Education, and Annual Income, and 100000 census records. The second subset consists of 8 attributes: the aforementioned five attributes, as well as Nativity, Ownership of Dwelling, and Number of Automobiles, and 30000 census records. For logistic regression, we select all attributes and 50000 census records.

Privacy specification. For personalized differential privacy, we generate the privacy budgets for all records randomly from uniform distribution and normal distribution. We set the range of privacy budget value ϵ from 0.01 to 1.0. $\epsilon = 0.01$ means the most conservative, representing users with high privacy concern while $\epsilon = 1.0$ means liberal, representing users with low privacy concern. We sample uniformly random privacy budgets directly from $U(0.01, 0.1)$, where U is the uniform distribution. We generate normal privacy budgets from $N(0.1, 1)$, where N denotes normal distribution.

Comparison. We evaluate the utility of the partition-based mechanisms for answering random range-count queries, differentially private support vector machine, and logistic regression, and compare it with the sampling mechanism [25] and baseline method *Minimum* which uses the minimum privacy budget and all records. For the other baseline method *threshold*, it is difficult to set optimal threshold values for different differentially private application, so we do not include it in our experiments.

Metrics. For count query evaluation, we generated random range-count queries with random query predicates covering all attributes defined in the following:
Select COUNT(*) from D
Where $A_1 \in I_1$ and $A_2 \in I_2$ and ... and $A_m \in I_m$
For each attribute A_i, I_i is a random interval generated from the domain of A_i.

We measure the count query accuracy by the relative frequency error defined as follows: for a query q, $A_{act}(q)$ is the true answer to q on the original data. $A_{noisy}(q)$ denotes the answer to q when using data generated from partitioning-based mechanisms or the sampling mechanism. Then the relative frequency error is defined as:

$$RFE(q) = \frac{A_{act}(q) - A_{noisy}(q)}{n}$$

where n is the total number of records in the original dataset. The reason we use relative frequency error is that the sampling mechanism generates a partial number of records from original datasets, and we need the query answers to be measured on the same base of record number.

For the support vector machine, we use the area under the curve (AUC), and higher AUC value means better discrimination. For logistic regression, we convert Annual Income into a binary attribute: values higher than a predefined threshold are mapped to 1, and 0 otherwise. Accordingly, when we use a logistic model to classify a tuple t, we predict the Annual Income of t to be 1 if $\frac{exp(x_i^T w)}{1+exp(x_i^T w)} > 0.5$, where w is the model parameter, and x is a vector that contains the values of t on all attributes except Annual Income. We measure the accuracy of a logistic model by its misclassification rate, i.e., the fraction of tuples that are incorrectly classified.

5.3.2. Experimental results.

5.3.2.1. *Partitioning-based mechanisms for count query.* Figure 5.2 to Figure 5.5 investigate the relative frequency error between our two partitioning mechanisms and the sampling mechanism under normal and uniform distribution of privacy preferences. We vary the privacy budget threshold of the sampling mechanism. The errors of the partitioning mechanisms remain at a horizontal line since it does not need to set privacy budget threshold. The accuracy of sampling mechanism reaches optimal when the budget threshold attains the mean of all privacy budget values, which is consistent with the experimental conclusion in [25]. From the results, we can observe that the accuracy of sampling mechanism deteriorates sharply when threshold value is smaller than the mean privacy budget. This is because when the number of records is sufficiently large, the privacy budget dominates the performance. Compared to sampling mechanism, our utility-based mechanism remain stable and perform almost the same with the optimal performance of sampling mechanism. The baseline method *Minimum* performs similarly with the privacy budget threshold being the smallest. This is due to the fact that when the threshold get the smallest value, sampling mechanism is equal to *Minimum*. This conclusion remains the same for the following experiments.

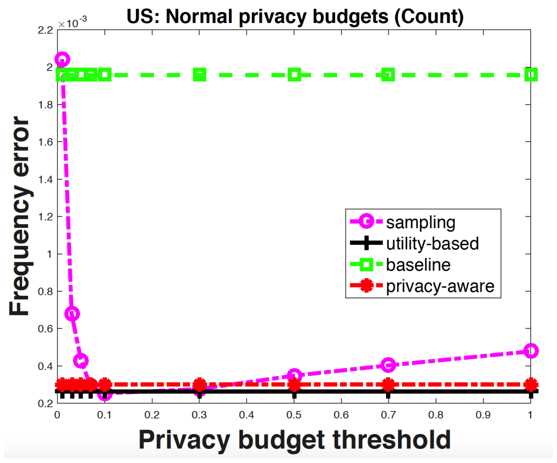


FIGURE 5.2. US: (Count) relative frequency error for the count task under normal privacy preferences, as privacy budget threshold are varied.

5.3.2.2. *Partitioning-based mechanisms for support vector machine.* Figure 5.6 to Figure 5.9 illustrate the performances of partitioning mechanisms, sampling mechanism and baseline *Minimum* for support vector machine classification task. We can observe that there is no obvious pattern for sampling mechanism on which privacy budget threshold has the optimal utility. This means in practice it is very difficult for a data publisher to choose the threshold for an optimal utility. However, our partitioning mechanisms have superior performance than sampling mechanism. The performance of sampling mechanism under uniform privacy budgets fluctuates, because the number of records in the experiment is small for SVM, and as a result, it

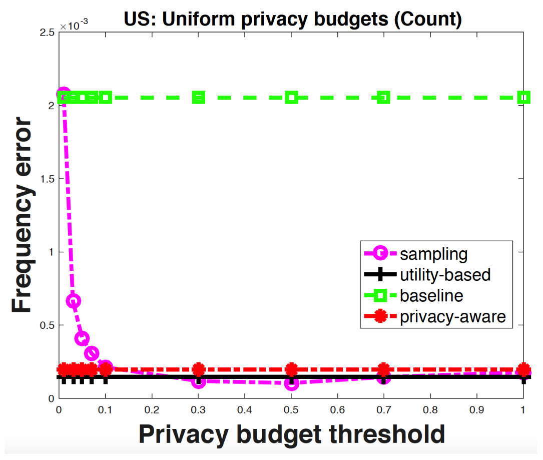


FIGURE 5.3. US: (Count) relative frequency error for the count task under uniform privacy preferences, as privacy budget threshold are varied.

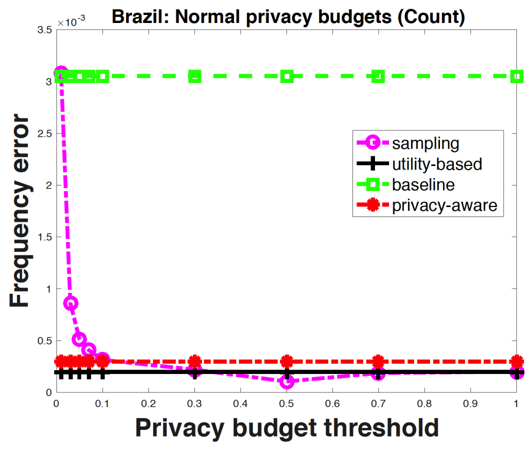


FIGURE 5.4. Brazil: (Count) relative frequency error for the count task under normal privacy preferences, as privacy budget threshold are varied.

it difficult to select an optimal threshold before running private SVM. The performance of sampling mechanism under normal privacy budgets arrives the best when the threshold value is around 0.5, which is approximate to the average of all privacy budgets.

5.3.2.3. *Partitioning-based mechanisms for logistic regression.* Figure 5.10 and Figure 5.11 study the performances of partitioning mechanisms and sampling mechanism for logistic regression classification task. For sampling mechanism, the optimal

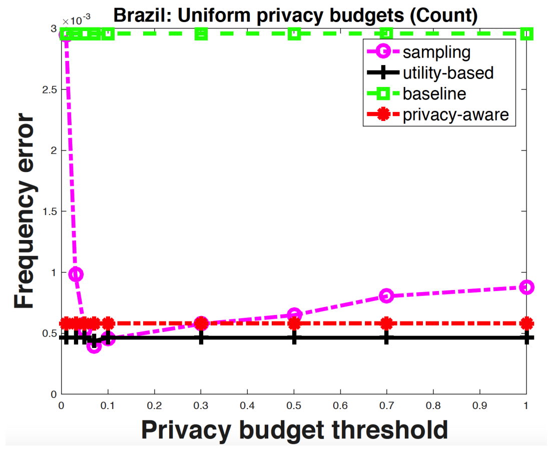


FIGURE 5.5. Brazil: (Count) relative frequency error for the count task under uniform privacy preferences, as privacy budget threshold are varied.

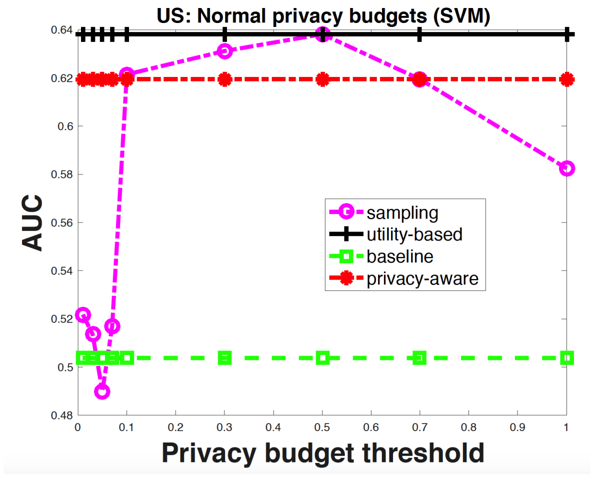


FIGURE 5.6. (SVM) AUC for support vector machine classification under normal privacy preferences, as privacy budget threshold are varied.

utility is obtained when the privacy budget threshold is set to be the maximum budget value. This is because we use training dataset with a sufficiently large data size for this task, and the dataset still contains sufficient information for learning, even if the privacy threshold is maximum corresponding to lowest sampling rate. In this situation, the only factor having impact on the utility is privacy budget, so the maximum threshold leads to the best performance. The partitioning mechanisms achieves comparable performance with the optimal error classification rate of sampling mechanism.

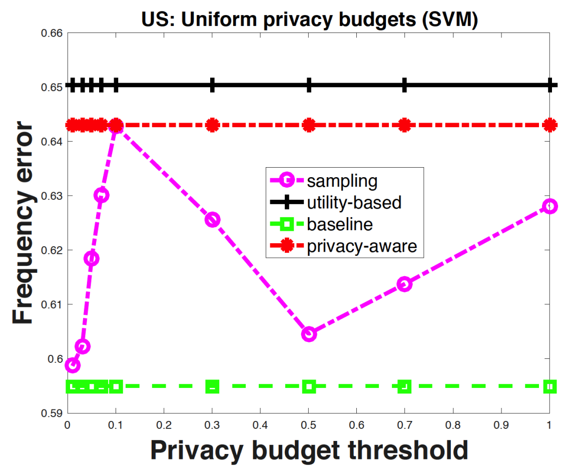


FIGURE 5.7. US: (SVM) AUC for support vector machine classification under uniform privacy preferences, as privacy budget threshold are varied.

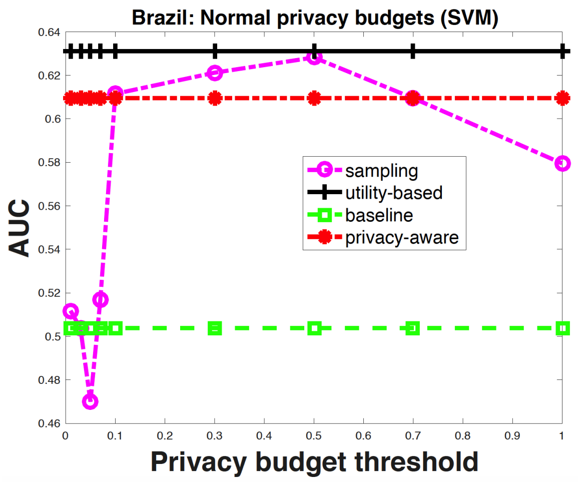


FIGURE 5.8. Brazil: (SVM) AUC for support vector machine classification under normal privacy preferences, as privacy budget threshold are varied.

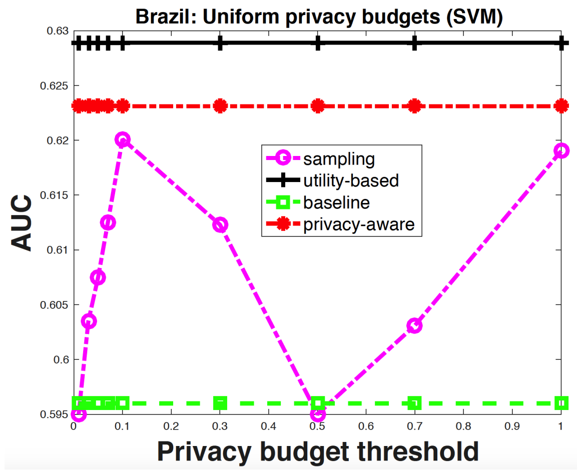


FIGURE 5.9. Brazil: (SVM) AUC for support vector machine classification under uniform privacy preferences, as privacy budget threshold are varied.

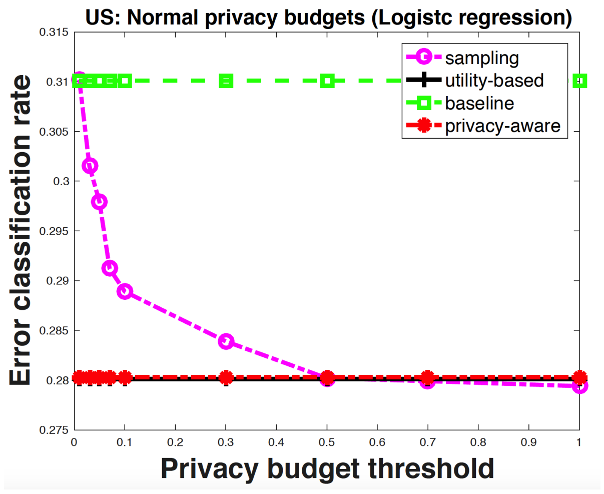


FIGURE 5.10. (Logistic regression) misclassification rate for logistic regression under normal privacy preferences, as privacy budget threshold are varied.

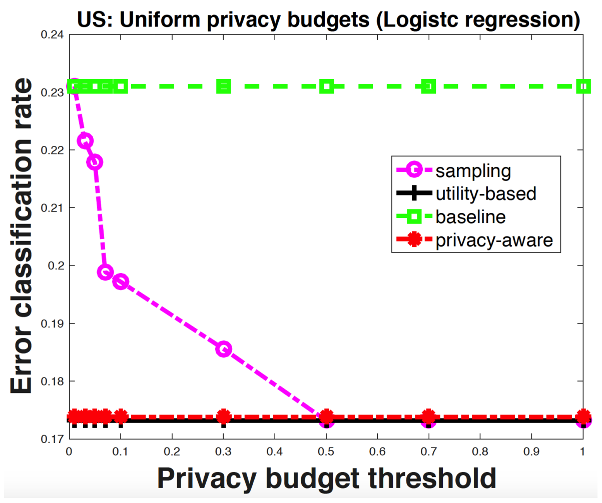


FIGURE 5.11. (Logistic regression) misclassification rate for logistic regression under uniform privacy preferences, as privacy budget threshold are varied.

CHAPTER 6

Conclusions

6.1. Summary of Dissertation

With the general trend of digitalization, information sharing has become part of the routine activity of many individuals, companies, institutes, and government agencies. Privacy-preserving data publishing techniques have been playing an increasingly important role in privacy protection in information sharing. Earlier research of PPDP focuses on protecting private and sensitive information in low-dimensional and static data release under uniform-level differential privacy. However, with the deployment of differential privacy in broader application domains, new privacy concerns have been substantially raised, including sharing high-dimensional and large domain data, series of dynamic datasets evolving in real time under uniform-level differential privacy, and personalized differential privacy which protects different privacy preferences of individuals. In this thesis, we response to these privacy concerns by developing efficient and effective non-interactive data publishing solutions for various utility requirements. We recap the major contributions of this thesis as follows:

- In Chapter 3, we presented DPCopula using copula functions for differentially private multi-dimensional data publication. Different from existing methods, DPCopula captures marginal distribution of each dimension and dependence between separate dimensions via copula functions. Our experimental studies on various types of datasets validated our theoretical results and demonstrated the efficiency and effectiveness of our algorithm, particularly on high-dimensional and large domain datasets. In the future, we are interested in employing other copula families and investigate how to select optimal copula functions for a given dataset. Second, we are interested in developing data synthesization mechanisms for dynamically evolving datasets.

- In Chapter 4, we have proposed an adaptive distance-based sampling approach to address the challenges of releasing a series of differentially private dynamic datasets in real time. With an upper bound to limit the number of DP data releases, our methods incur much smaller errors. We apply an adaptive control mechanism to dynamically adjust the threshold value. We also provide privacy and utility analysis for our method. Experiments on real and synthetic datasets show that our algorithm outperforms the baseline and existing state-of-the-art techniques. As future work, we would like to study update models and incorporate them into our sampling framework. We are also interested in applying the adaptive sampling framework for releasing other types of dynamic data with differential privacy, e.g. frequent patterns for dynamically changing transactional data and dynamic graph patterns in social networks.
- In Chapter 5, we developed two partitioning-based mechanisms for personalized differential privacy framework, which combines the strength of differential privacy with the flexibility of user-specific privacy preferences and guarantees. The first privacy-aware partitioning mechanism partitions original dataset in order to minimize the waste of privacy budgets, while optimizing number of partitions to guarantee an optimal number of records in each partition. The utility-based partitioning groups records into partitions to maximize a utility function, defined for different differentially private algorithms. We also find a common utility function form for a series of state-of-the-arts privacy preserving mechanisms. In addition, the number of partitions of both partitioning-based mechanism can be computed through optimization functions.

In conclusion, as a preliminary effort toward privacy in high-dimensional data publishing, dynamic data release and personalized differential privacy, this thesis has reported encouraging results, which demonstrate great promise for releasing useful high-dimensional or dynamic data while preserving individual privacy, and personalized differential privacy mechanisms.

6.2. Recommendations for Future Work

Although experimental results in this dissertation demonstrated effectiveness of the proposed methods for a variety of data release and learning tasks, they can be extended in the following future directions:

Differentially private synthesization of multi-dimensional data. First, we are interested in employing other copula families, like t-copula to model more types of joint dependence and investigate how to select optimal copula functions for a given dataset. Second, we are interested in developing group-copula functions to build copula function on subsets of attributes correlated within each other. Finally, we can consider to develop data release methods under other privacy frameworks, for example, Pufferfish privacy which is a recent generalization of differential privacy.

Privacy-preserving dynamic histogram release. One potential direction would be to study update models and incorporate them into our sampling framework. We are also interested in applying the adaptive sampling framework for releasing other types of dynamic data with differential privacy, e.g. frequent patterns for dynamically changing transactional data and dynamic graph patterns in social networks. Another future work would focus on releasing correlated data histograms or data series. Exponential mechanism may also be considered to release dynamic data.

Personalized differential privacy. An important future research would be personalized local differential privacy. With the application of cloud computing and mobile devices, these personalized privacy problems will become increasingly significant. A combination of encryption and privacy protection may also need to be investigated. It will also be of interest to extend personalized privacy to social networks, for example, facebook, where individuals are nodes and friendship connections are edges.

Bibliography

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

[2] C. Dwork. Differential privacy: A survey of results. In Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008. Proceedings, pages 1-19, 2008.

[3] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. Theory of Cryptography, pages 1-20.

[4] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In STOC, pages 715-724, 2010.

[5] C. Dwork and A. Smith. Differential privacy for statistics: What we know and what we want to learn. Journal of Privacy and Confidentiality, 1,Number 2:135-154, 2009.

[6] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pages 19-30, 2009.

[7] Frank McSherry, Kunal Talwar: Mechanism Design via Differential Privacy. FOCS 2007: 94-103

[8] A. Friedman and A. Schuster. Data mining with differential privacy. In SIGKDD, 2010.

[9] R. B. Nelsen. An Introduction to Copulas. Springer, 1999.

[10] K. Osband. In Iceberg Risk: An Adventure in Portfolio Theory, 2002.

[11] V. D. A. N. G. R. Bouye, E. and T. Roncalli. Copulas for finance a reading guide and some applications, 2000.

[12] S. Demarta and A. J. McNeil. The t copula and related copulas. International Statistical Review, 73:111-129, 2007.

[13] L. Wasserman and S. Zhou. A statistical framework for differential privacy. JASA, 105(489):375-389, 2009.

[14] G. Rousseeuw, P. Molenberghs. Transformation of non positive semi-de
nite correlations matrices. Communications in statistics: theory and methods, 22:965-984, 1993.

[15] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007 pages 106-115, 2007.

[16] Y. Tao and X. Xiao. Personalized privacy preservation. In Privacy-Preserving Data Mining Models and Algorithms, pages 461- 485. 2008.

[17] H. Li, L. Xiong, and X. Jiang. Differentially private synthesization of multi-dimensional data using copula functions. In Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, pages 475-486, 2014.

[18] H. Li, L. Xiong, L. Zhang, X. Jiang: DPSynthesizer: Differentially Private Data Synthesizer for Privacy Preserving Data Sharing. PVLDB 7(13): 1677-1680 (2014)

[19] R. Chen, H. Li, A. K. Qin, S. Kasiviswanathan, and H. Jin. Private spatial data aggregation in the local setting. In 31st IEEE International Conference on Data Engineering, ICDE, 2016.

[20] P. Wang. Personalized anonymity algorithm using clustering techniques. In JCIS, page 7(3):924-931, 2011.

[21] X. Ye, Y. Zhang, and M. Liu. A personalized (a, k)-anonymity model. In The 9th International Conference on Web-Age Information Management, WAIM 2008, pages 341-348.

[22] M. Alaggan, S. Gambs, and A. Kermarrec. Heterogeneous differential privacy. Workshop on Theory and Practice of Differential Privacy (TPDP'15) alongside ETAPS'15, 2015.

[23] D. S. H. Ebadi and G. Schneider. Differential privacy: Now it's getting personal. Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pages 69-81, 2015.

[24] D. Leoni. Non-interactive differential privacy: A survey. Proceedings of the First International Workshop on Open Data, pages 40-52, 2012.

[25] Z. Jorgensen, T. Yu, and G. Cormode. Conservative or liberal? personalized differential privacy. In 31st IEEE International Conference on Data Engineering, ICDE

[26] M. T. W. Y. H. X. S. Wang, L. Huang and H. Guo. Personalized privacy-preserving data aggregation for histogram estimation. IEEE Global Communications Conference (GLOBECOM), pages 1-6, 2015.

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

[28] M. Yuan, L. Chen, and P. S. Yu. Personalized privacy protection in social networks. PVLDB, 4(2):141-150, 2010.

[29] Y. Shen, H. Shao, and Y. Li. Privacy preserving distributed data mining based on multi-objective optimization and algorithmic game theory. In IEEE FITME, pages 436-439, 2009.

[30] T. Brinkhoff. A framework for generating network-based moving objects. In Geoinformatica, pages 6(2), 153-180, 2002.

[31] K. H. Ang, G. Chong, and Y. Li. Pid control system analysis, design, and technology. IEEE Trans. Contr. Sys. Techn., 13(4):559576, 2005.

[32] K. L. Avrim Blum and A. Roth. A learning theory approach to non-interactive database privacy. In STOC, 2008.

[33] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In FOCS, pages 6170, 2010.

[34] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. ACM Trans. Inf. Syst. Secur., 14(3):26, 2011.

[35] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In SIGMOD, pages 735-746, 2010.

[36] L. Fan, L. Bonomi, L. Xiong, and V. Sunderam. Monitoring web browsing behaviors with differential privacy. In WWW Conference, 2014.

[37] L. Fan and L. Xiong. An adaptive approach to real-time aggregate monitoring with differential privacy. IEEE TKDE, 26(9):2094-2106, 2014.

[38] X. Zhang, X. Meng, and R. Chen. Differentially private set-valued data release against incremental updates. In DASFAA (1), 2013.

[39] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. PVLDB, 7(12):1155-1166, 2014.

[40] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In PODS, 2007.

[41] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In ICDE, pages 225-236, 2010.

[42] M. Hayy, V. Rastogiz, G. Miklauy, and D. Suciu. Boosting the accuracy of differentially-private histograms through consistency. VLDB, 2010.

[43] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu. Differentially private histogram publication. In ICDE, 2012.

[44] G. Cormode, C. M. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In ICDE, 2012.

[45] G. Cormode, C. M. Procopiuc, D. Srivastava, and T. T. L. Tran. Differentially private summaries for sparse data. In ICDT, pages 299-311, 2012.

[46] N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. Differentially private data release for data mining. In SIGKDD, 2011.

[47] G. A'cs, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In ICDM, 2012.

[48] W. H. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In ICDE, 2013.

[49] Yonghui Xiao, Li Xiong, Liyue Fan, Slawomir Goryczka, Haoran Li: DPCube: Differentially Private Histogram Release through Multidimensional Partitioning. Trans. Data Privacy 7(3): 195-222 (2014)

[50] H. Li, L. Xiong, X. Jiang, and J. Liu. Differentially private histogram publication for dynamic datasets: an adaptive sampling approach. In The 24th ACM International Conference on Information and Knowledge Management, 2015.

[51] Michael Barbaro and Tom Zeller. A face is exposed for aol searcher no. 4417749. The New York Times, August 2006.

[52] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map, In International Conference on Data Engineering, 2008.

[53] S. Shalev-Shwartz and N. Srebro. Svm optimization: inverse dependence on training set size. In The 25th International Conference on Machine Learning, 2008.

[54] S. Fletcher and M. Z. Islam. A differentially private random decision forest using reliable signal-to-noise ratios. In AI 2015: Advances in Artificial Intelligence - 28th Australasian Joint Conference, pages 192203, 2015.

[55] Y. Cao. Rigorous and flexible privacy models for utilizing personal spatiotemporal data. In The 42nd International Conference on Very Large Databases, 2016.

[56] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong. Quantifying differential privacy under temporal correlations. In 33rd IEEE International Conference on Data Engineering, 2017.

[57] Y. Cao and M. Yoshikawa. Differentially private real-time data release over infinite trajectory streams. In 16th IEEE International Conference on Mobile Data Management, 2015.

[58] C. M. K. Chaudhuri and A. D. Sarwate. Differentially private empirical risk minimization. Journal of Machine Learning Research, pages 12:1069-1109, 2011.

[59] L. Breiman. Bagging predictors. Machine Learning, 24(2):123140, 1996.

[60] L Haoran, X Li, J Zhanglong, X Jiang: Partitioning-Based Mechanisms Under Personalized Differential Privacy . In Advances in Knowledge Discovery and Data Mining-21st Pacific-Asia Conference (I), PAKDD 2017, 615-627.

[61] Haoran Li, Li Xiong, Lucila Ohno-Machado, and Xiaoqian Jiang. Privacy Preserving RBF Kernel Support Vector Machine. In the Journal of Biomedicine and Biotechnology, 2014.

[62] Haoran Li, Li Xiong, Xiaoqian Jiang: Differentially Private Histogram and Synthetic Data Publication. Medical Data Privacy Handbook 2015: 35-58.

[63] G. Jagannathan, C. Monteleoni, and K. Pillaipakkamnatt. A semi-supervised learning approach to differential privacy. In 13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, pages 841-848, 2013.

[64] Rui Chen, Haoran Li, Hongxia Jin. Private Spatial Data Aggregation in the Local Setting. In International Conference on Data Engineering, 2016.

[65] S. Xu, X. Cheng, S. Su, K. Xiao, and L. Xiong. Differentially private frequent sequence mining. IEEE Trans. Knowl. Data Eng., 28(11):2910-2926, 2016.

[66] Zhang Sicong, Hui Yang Grace, Singh Lisa, Xiong Li: Safelog: Supporting Web Search and Mining by Differentially-Private Query Logs. In AAAI Fall Symposium on Privacy and Language Technologies (FSS'16, PLT'16).