July 2021

Towards Practical Differentially Private Mechanism Design and Deployment

Dan Zhang
University of Massachusetts Amherst

TOWARDS PRACTICAL DIFFERENTIALLY PRIVATE MECHANISM DESIGN AND DEPLOYMENT

A Dissertation Presented

by

DAN ZHANG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2021

College of Information and Computer Sciences

TOWARDS PRACTICAL DIFFERENTIALLY PRIVATE MECHANISM DESIGN AND DEPLOYMENT

A Dissertation Presented

by

DAN ZHANG

Approved as to style and content by:

Gerome Miklau, Chair

Ali Sarvghad, Member

Alexandra Meliou, Member

Çağatay Demiralp, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

ACKNOWLEDGMENTS

First, I want to express thanks to my awesome advisor Professor Gerome Miklau. When I started my PhD at UMASS, I had barely any experience with differential privacy and was just generally interested in the security and privacy management of data. Gerome got me interested in the topic of making formal privacy guarantee more accessible in practical applications and guided me through this journey. A PhD study can get stressful or even depressing at some points. He has been very supportive in the hard times and respected my interests in pursuing different topics. I also want to thank Professor Ali Sarvghad. Many of my projects focus on the application of privacy techniques to data visualization. Ali as an expert from the visualization community provided valuable domain insights and knowledge. He's also a cheerful collaborator and always encourages and pushes (in a good way) me to submit to top-tier visualization conferences when I feel unconfident.

I would like to thank my smart and hardworking collaborators Ryan McKenna, George Bissias, Ios Kotsogiannis. Ryan is the person people enjoy having technical discussions with and always dive deep into questions. George helped a lot in the open-source effort of our differential privacy tools and improved our code quality. Ios is fun to work with and I still remember working on some final changes of submission with him in a hotel lobby while attending another conference. My gratitude extends to Professor Michael Hay, Professor Ashwin Machanavajjhala, and Professor Brendan O'Connor who always provide insightful feedback and detailed advice.

I am proudly a member of the UMASS DREAM(Data systems Research for Exploration, Analytics, and Modeling) lab. Alumni Yue Wang and Xiaolan Wang helped me onboard smoothly in terms of both research and life when I was a junior graduate. I was also fortunate to have spent a lot of time with lovely labmates Anna Fariha, Matteo Brucato,

Chenghao Lyu... (It's a pity that I don't get to know the newer members better due to the pandemic). Thanks to Professor Alexandra Meliou for organizing the paper reading seminars, game and lunch socials and pushing the lab renovations to improve our workspace. Professor Yanlei Diao and Professor Macro Serafini usually ask sharp and insightful questions in our group discussion. Professor Peter Haas is not only greatly knowledgeable in various domains but also actively helps us connect with industry and other research institutions. Alexandra, Yanlei, and Dean Dr. Laura Haas have been convincing examples and great role models of women in computing for us. Thanks to all the faculty members for making DREAM lab a place where everyone could learn and grow.

I have done two internships during my PhD and I was fortunate to have worked with great mentors and teams both times. My mentor at Google Eamonn O'Brien-Strain gave me a lot of freedom to explore an experimental project on user data wipeout from cloud storage which was later open-sourced as part of the Firebase privacy protection effort. I did my second internship at Megagon Labs working on a semantic type project which did not entirely match my PhD research topic. Although I didn't have strong machine learning research background, my mentor Cagatay Demiralp believed in me and provided the right amount of guidance throughout the experience. He provided pointers to learning resources and encouraged me to talk to people and ask questions, while at the same time gave me the right to decide on the path to dive down and approaches to use. I also want to thank Yoshihiko Suhara, the machine learning expert whom I keep bothering. He has always been patient and cheerful. Other team members Jinfeng Li, and Wang-Chiew Tan have also been giving timely help. With all the help, this project lead to a VLDB publication and had acquired attention from both academia and industry.

I also want to thank my great friends Yingying and Chuchu. We started our PhD at UMASS in the same year and know each other's struggles and support each other through the process. They are an irreplaceable part of my memory about Amherst.

I thank my parents for raising me into who I am and for always believe in me. I still remember the happy childhood time I spent with my dad when he explained how seasons change and how machines work. He encouraged me to ask questions and let me know it is ok for girls to prefer toy cars over dolls. He is probably the first person that got me interested in science and engineering. Although I have never told her in person, my mom has always been my role model growing up. She taught me and showed me how to be an honest and diligent person. It would be impossible for me to pass all those stressful times without her unconditional love and support. And I am happy and proud that I have grown enough to also support her back. We still share our up and downs and I am very grateful for the close relationships we have even after I left home.

Last but not least, thanks to my partner Pan for his love and support. He is smart, hardworking, modest, and a loving son and brother. He is one of the most energetic people I know and his passion for research and life impacted me. Without him, I would not even imagine myself camping at 10,000 feet elevation and start hiking before sunrise. He also calms me down from anxiety by sharing his own experiences and being there for me. We've been together for five years and spent three years on different coasts of the US. Long-distance relationships can be difficult but we have learned to think for each other and have grown together. I am beyond grateful to have him in my life.

ABSTRACT

TOWARDS PRACTICAL DIFFERENTIALLY PRIVATE MECHANISM DESIGN AND DEPLOYMENT

MAY 2021

DAN ZHANG

B.Eng., HARBIN INSTITUTE OF TECHNOLOGY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Gerome Miklau

As the collection of personal data has increased, many institutions face an urgent need for reliable protection of sensitive data. Among the emerging privacy protection mechanisms, differential privacy offers a persuasive and provable assurance to individuals and has become the dominant model in the research community. However, despite growing adoption, the complexity of designing differentially private algorithms and effectively deploying them in real-world applications remains high.

In this thesis, we address two main questions: 1) how can we aid programmers in developing private programs with high utility? and 2) how can we deploy differentially private algorithms to visual analytics systems? We first propose a programming framework and system ϵKTELO which can be used to author programs for a variety of statistical tasks that involve answering counting queries. In the framework, programs are described as compositions of reusable modules and automatically satisfy differential privacy. Moving on to the second question, we investigate the challenges of deploying differentially

private algorithms in visualization tasks. Specifically, we conduct a study to better understand the relationship between noise introduced for privacy protection, visual analytics tasks, visualization, and accuracy. We also look at the influence of uncertainty in differentially private visualization and propose an approach to effectively represent uncertainty in two-dimensional location data. Third, we demonstrate how direct deployment of differentially private algorithms causes both efficiency and accuracy issues in an interactive visualization dashboard. To address these challenges, we propose a DashGuard, a private dashboard where a smart middle layer processes front-end queries issued to the back-end ϵKTELO private engine. Through reuse and pre-computation of measurement, the middle layer provides benefits in accuracy, efficiency, and privacy budget consumption.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Motivation

As the world gets more and more data-driven, the collection of personal data has increased. This creates an urgent need for reliable privacy protection mechanisms for institutions that collect and share sensitive personal data. They must balance the need to protect individuals with demands to use the collected data for new applications or modeling their users' behavior.

Privacy technology has emerged as a response to the dire need to protect individuals' sensitive information (e.g., [86, 60, 55]). In particular, Differential Privacy (DP) is becoming the dominant model for data privacy protection [26]. It is a rigorous privacy definition that offers a persuasive assurance to individuals, provable guarantees, and the ability to analyze the impact of combined releases of data. Informally, an algorithm satisfies differential privacy if its output does not change too much when any record in the input database is added or removed. To preserve privacy, a typical DP algorithm adds carefully calibrated noise that blurs information about individuals while preserving overall statistics about the population.

The research community has actively investigated differential privacy and algorithms are known for a variety of tasks ranging from data exploration to query answering to machine learning. However, the adoption of differentially private techniques in real-world applications (e.g., visual exploration systems and machine learning applications that use sensitive data) remains rare until very recent years. In fact, despite the rising popularity, the real-world deployments of differential privacy—like OnTheMap [2, 34] (a U.S. Census

Bureau data product), RAPPOR [29] (a Google Chrome extension), and Apple's private collection of emoji's and HealthKit data—have required teams of privacy experts to ensure that implementations meet the privacy standard and that they deliver acceptable utility.

We believe adoption is impeded by major challenges in both designing and implementing differentially private algorithms and deploying them in real applications. We'll go over detailed problems from both aspects and propose solutions in later sections.

1.1.1 DP Algorithm Design and Implementation

Implementing programs that provably satisfy privacy and ensure sufficient utility for a given task is extremely difficult for non-experts in differential privacy. Here we identify three important challenges.

Challenge 1: Designing utility-optimal algorithms The first and foremost challenge is the difficulty of designing utility-optimal algorithms: i.e., algorithms that can extract the maximal accuracy given a fixed "privacy budget." While there are several general-purpose differentially private algorithms, such as the Laplace Mechanism [26] (which adds noise to queries posed on the private data), they typically offer suboptimal accuracy if applied directly. A carefully designed algorithm can improve on general-purpose methods by an order of magnitude or more—without weakening privacy: accuracy is improved by careful engineering and sophisticated algorithm design.

One might hope for a single dominant algorithm for each task, but a recent empirical study [38] showed that the accuracy of existing algorithms is complex: no single algorithm delivers the best accuracy across the range of settings in which it may be deployed. The choice of the best algorithm may depend on the particular task, the available privacy budget, and properties of the input data such as its size and distribution. Therefore, to achieve state-of-the-art accuracy, a practitioner currently has to make a host of complex algorithm choices, which may include choosing a low-level representation for the input data, translat-

ing their queries into that representation, choosing among available algorithms, and setting parameters. The best choices will vary for different input data and different analysis tasks.

Challenge 2: Diverse tasks The second challenge is that the tasks in which practitioners are interested are diverse and may differ from those considered in the literature. Hence, existing algorithms need to be adapted to new application settings, a non-trivial task. For instance, techniques used by modern privacy algorithms include optimizing error over multiple queries by identifying common sub-expressions, obtaining noisy counts from the data at different resolutions, and using complex inference techniques to reconstruct answers to target queries from noisy, inconsistent and incomplete measurement queries. But different algorithms use different specialized operators for these sub-tasks, and it can be challenging to adapt them to new situations. Thus, designing utility-optimal algorithms requires significant expertise in the complex and rapidly-evolving research literature.

In addition to diverse workloads of range queries, there has been an increasing need in preserving individual privacy for much more complex tasks like visual analysis and machine learning algorithms. We discuss the specific challenges of deploying differentially private algorithms in visual analytics systems in Sec. 1.1.2.

Challenge 3: Privacy proof A third equally important challenge is that correctly implementing differentially private algorithms can be difficult. There are known examples of algorithm pseudocode in research papers not satisfying differential privacy as claimed. For instance, Zhang et al [110] showed that many variants of privacy primitive called the Sparse Vector Technique do not satisfy differential privacy. Differential privacy can also be broken through incorrect implementations of sound algorithms. For example, Mironov [66] showed that standard implementations of basic algorithms like the Laplace Mechanism [26] can violate differential privacy because of their use of floating-point arithmetic. Privacy-oriented programming frameworks such as PINQ [64, 27, 77], Fuzz [31], PrivInfer [9] and LightDP [108] help users implement programs whose privacy can be verified with relatively little human intervention. While they help to ensure the privacy criterion is met, they

may impose their own restrictions and offer little or no support for designing utility-optimal programs.

1.1.2 Deployment in Visual Analytics Systems

Even with well-designed and carefully-implemented differentially private algorithms, deploying them in real-world applications remains nontrivial. Among other applications, we focus on visual analytics systems (both static and interactive) in this thesis. Visual data analysis is widely used across several application domains such as health care and civic decision making for analyzing data about people. Such applications introduce risks of harm to individuals due to possible leakage of sensitive information and calls for reliable privacy protection.

Despite the rising popularity of DP, prior research in the confluence of privacy and visual data analysis has been mainly focused on the use of syntactic privacy models such as k-anonymity and l-diversity (e.g.,[96, 15, 14]). We still know little about the effective use of DP in the context of visual data analysis. More specifically, we look at two typical scenarios

1.1.2.1 Static Private Visualization

To generate static visualization for sensitive data using DP techniques, we need to learn about and manage the effects of noise injection on the `utility` of `private visualizations` created using the output of DP algorithms. Injection of noise by DP algorithm results in data perturbation which in turn causes alteration of visual patterns.

Figure 1.1 shows an example of such effects. Histogram B (at the bottom) is the differentially private counterpart of histogram A (at the top). Data perturbation has resulted in noticeable differences between the histograms. Though privacy-preserving, using histogram B for visual analysis may result in erroneous findings (e.g., finding the group with the minimum value), improper conclusions, and wrong decisions. Hence, noise injection can compromise the utility of private visualizations as well as the reliability of insights

gained through visual analysis. This work is a first step towards gaining a deeper understating of the utility of private visualizations. In particular, we focus on two research questions:

- (RQ1) What is the relationship between noise-injection level, visualization type, visual analysis tasks, and user performance?
- (RQ2) How can we measure the influence of noise injection to downstream perceptual accuracy and choose proper DP algorithms?

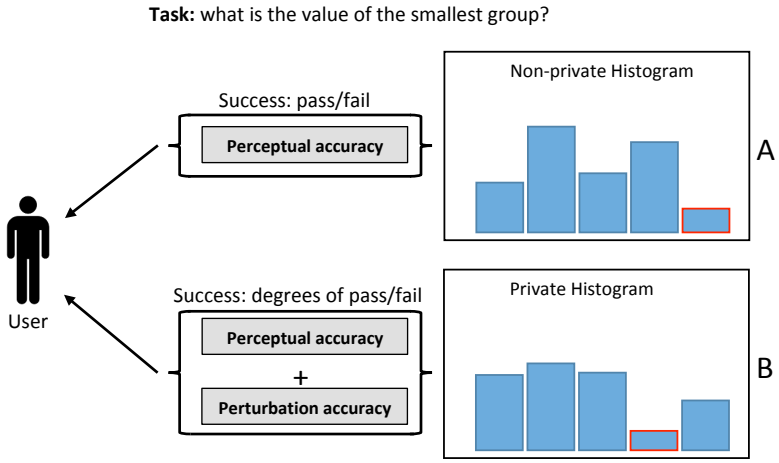


Figure 1.1: On the right-hand side, histogram B (bottom) is the private counterpart of the histogram A (top). Data perturbation caused by the injection of noise has resulted in noticeable alterations in visual patterns and data values. Consequently, the utility of histogram B for supporting several visual analysis tasks is compromised. For instance, for the task, "identify the group with smallest value" using histogram B, even if the user correctly identifies the bar with the red border as the smallest, still, his finding is erroneous based on the non-private histogram, A.

Uncertainty Due to the privacy requirements, differentially private algorithms are inherently randomized. This introduces unavoidable uncertainties into the output. With the standard private visualization approaches, only one random output has been shown to the user and uncertainty is not presented. Appropriate visualization of uncertainty is a key challenge in visualizing differentially private data. This is similar to the challenge of visualizing statistical uncertainty, in which a practitioner is encouraged to not directly trust

data (since there is uncertainty in statistical inference), or forecasts from a computational model like climate simulations (since there's uncertainty in the model's accuracy).

1.1.2.2 Private interactive dashboards

Static visualizations are effective tools to tell specific data stories when included in papers, printed in magazines or posted on web pages. However, users can't go beyond the predefined view to get additional stories or further explore the data. To tackle the limitations, interactive visual dashboards have been widely used in exploratory data analysis and complex data storytelling. Those dashboards visually organize, analyze and display important statistics about the underlying data. For differentially private frameworks like ϵKTELO, supporting interactive exploration also greatly extends their application scenarios.

A typical interactive dashboard usually includes several small panels showing views of the underlying data using visualization techniques like scatterplots, linecharts or geographical distributions over maps. The corresponding views and visualization types usually customizable by the user to meet special data exploration needs. Then the user can interact with the data by zooming in and out interesting regions or drill down by filtering with certain conditions. Besides interacting with a single visualization view, they can also *link* multiple panels such that zooming and filtering interactions on one view will be reflected on all linked panels. This powerful functionality allows users to compare and explore relationships between different data aspects. In this work, we limit the set of interactions to independent browsing, brushing, and linking (linked filtering) of univariate histograms.

In order to perform the exploration in a private-preserving manner, a naive model could privatize the query requested by the user. A user interacts with the front-end visualization dashboard where every interaction request is converted to a query on the underlying data. To protect privacy, every knowledge learned about sensitive data needs to be conducted in a privacy-preserving way. However, this simple model raises certain challenges in practical deployments.

Challenge 1: potential waste of privacy budget A lot of times, queries in the sequence are correlated or even duplicated. Therefore, there are chances that the answer to a certain query could be constructed from earlier results (or even from reusing previous results). Since differential privacy is safe under post-processing, any previous result generated consuming privacy budget can be reused without paying additional budget. For example, any brushing and linking results between attributes A and B can be re-constructed using the histogram on the cross-domain of A and B.

Challenge 2: inconsistent brushing and linking Common differentially private algorithms achieve the guarantee by adding noise or resampling the data. This introduces errors and corresponding inconsistency within interactions which could confuse the users. When a user performs brushing on the source visualization, the linked target should change to the filtered view based on the brushing attribute. Thus, the new target visualization should only contain a subset of data points, leading to strictly lower counts in a histogram. However, with independent noise injected at each step of the interaction sequence, there are chances that the filtered results get higher noise counts than the original data.

Challenge 3: slow response time Computing every query on-the-fly as shown in Fig. 6.1 can be slow and might break the interactivity of the user's workflow, especially when using the more complex and time-consuming mechanisms. One possible alternative is to precompute everything, generating a synthetic version of the full joint distribution over all the attributes. However, this is not practical since the size of the full joint grows exponentially with respect to the domain size and may go beyond the limit of machines' physical memory.

1.2 Thesis Outline

Chapter 2 provides the necessary background knowledge on differential privacy.

To address the aforementioned challenges, in Chapter 3, we propose ϵKTELO, a programming framework and system that aids programmers in developing differentially pri-

vate programs with high utility. Programmers can use ϵKTELO to solve a core class of statistical tasks that involve answering *linear counting queries*. (This class of queries is defined in Chapter 2.) Tasks supported by ϵKTELO include releasing contingency tables, multi-dimensional histograms, answering OLAP and range queries, and implementing private machine learning algorithms.

Next, we start to look at the application of differentially private techniques into visualization applications. Chapter 4, we investigate issues in static private visualization, trying to answer the two proposed research questions. To investigate RQ1, we performed a crowd-sourced user study and examined the effects of three noise-levels (none, low, high) on participants' ability to successfully carry out fundamental visual analysis tasks (eight) on four common four visualizations (bar chart, pie chart, line chart, scatterplot). We measured the task success rate and response time for 204 participants. In order to answer RQ2, we investigate the influence of data distribution on users' ability to accurately perceive visualizations and identify certain that data with certain characteristics are "easier" for visual tasks. Prior research (e.g., [49, 35])in differential privacy has explored different approaches and mechanisms to achieve a certain level of privacy. These variations can result in different distributions of privatized data. Grounded in this phenomenon and inspired by prior work on Scagnostics (e.g., [98, 99, 20], we designed three simple `distribution metrics` to quantify the shape of data distribution in univariate histograms. `Peakedness score`, `Anomaly score`, and `Clusteredness score` respectively measure to what extent there exists a single peak, an anomaly data point or clear cluster boundaries. Based on the combined results of RQ1 and RQ2 investigations, in Chapter 4, we suggest a set of preliminary guidelines for algorithm choice and noise injection configuration to achieve better utility with respect to specific tasks and visualizations. The outcomes of this work will serve as a foundation for static differentially private visualization which can aid practitioners as well as provide insights for researchers who specialize in either privacy or visualization.

Then in Chapter 5, we take a deeper look at the uncertainty issues in private visualizations. We use two-dimensional location data as an example domain, and consider the proper way to present known uncertainty about private output. Then we propose the indistinguishability principle as an attempt to address the challenge. We present a solution to achieve the principle and conduct an online user study on correlation perception, which is a well-studied visual task. Results of the user study show our solution is better than the naive baseline visualizer.

Next, we move on and look at privacy protection in interactive cases. To address the three challenges of the naive model, in Chapter 6, we propose DashGuard, a framework using a smart middle layer that acts as a proxy of any front-end interactions and back-end queries. DashGuard is fully supported by ϵKTELO as the backend.

The sensitive data source is encapsulated in the secure kernel space of ϵKTELO and any access to the data is done privately by ϵKTELO client. The middle layer takes care of optimizations regarding the utility of visual interactions. Motivated by the fact that some queries in an interaction sequence can benefit from reusing partial or full information from previous results, the middle layer automatically caches all previous interaction results and only issues new back-end queries when necessary. Before returning the results, it performs inference to resolve potential inconsistencies. The use of the smart middle layer provides substantial improvements on the accuracy, response time, and privacy budget consumption.

Finally, we conclude the thesis in Chapter 7.

1.3 Outcomes

We proposed and implemented a programming framework ϵKTELO to help programmers write privacy-preserving range query answering algorithms with good utility. The work was published at SIGMOD ([106]) and TODS ([104]). The conference version was selected as a 2018 SIGMOD Research Highlight. The open-source code has been incorporated into PSynDB[43] which is a web-based synthetic table generator.

To better understand the issues in introducing noise into the visual analytics process, we conducted a crowd-sourced study[107] on univariate visualizations published at VIS 2020. [103] summarized challenges with a focus on two-dimensional location data. Moving on to the interactive scenario, we introduced DashGuard to deal with the challenges in privacy-preserving interactions in visual dashboards. The final piece of work has not been published yet by the time this thesis is submitted.

CHAPTER 2

BACKGROUND

The input to the differentially private mechanisms studied in this thesis is a database instance of a single-relation schema $T(A_1, A_2, \ldots, A_\ell)$. Each attribute A_i is assumed to be discrete (or suitably discretized). Let D and D' denote two tables of the same schema, and let $D \oplus D' = (D - D') \cup (D' - D)$ denote the symmetric difference between them. We say that D and D' are neighbors if $|D \oplus D'| = 1$. The formal definition of differential privacy is as follows:

Definition 1 (Differential Privacy [26]) *a randomized algorithm $\mathcal{A}$ satisfies ϵ-differential privacy [26] if for all databases D and D' that differ on one record, and for any subset of outputs $S \subseteq Range(\mathcal{A})$,*

$$Pr(\mathcal{A}(D) \in S) \leq e^{\epsilon} \times Pr(\mathcal{A}(D') \in S)$$

Differentially private algorithms are randomized. We can think of the output of the algorithm, on a given input, as a probability distribution over possible outputs. The definition requires that, if the algorithm runs on any two databases that differ on the details of any individual's record, the output distributions will be "close", where close is formalized by the e^{ϵ} term. As a consequence, seeing the output of the algorithm cannot reveal much about any one contributor's data. The privacy loss parameter ϵ therefore controls the level of privacy protection: smaller ϵ means a stricter limit on privacy loss and, in general, this means the algorithm output will be able to communicate less information about the underlying data.

The data owner must choose an ϵ value to grant to a user of sensitive data. This parameter can be thought of as a privacy loss "budget". Granting a higher ϵ to a user generally means they can receive more accurate results from the private algorithm. In the research literature, a common value for ϵ is 0.1, in which case $e^{\epsilon} \approx 1.1$ and the likelihood of any output cannot differ by more than about 10% on inputs that differ by one record. In practical deployments, higher ϵ values have been used and may still provide reasonable privacy protection.

Laplace Mechanism A standard method for achieving differential privacy is the Laplace Mechanism [26] (although there are many other mechanisms). When the Laplace Mechanism is used to answer a query over a sensitive database (e.g., how many people have the marital status of "divorced") the true query result is computed on the data and then carefully calibrated noise is added to the answer before it is returned to the user. In particular, a sample is drawn from the Laplace distribution with mean zero and a specified scale factor determined by the ϵ parameter and property of the query. The process is ϵ-differentially private and the resulting "noisy" query answer may be shared with the user.

Formally:

Definition 2 (Laplace Mechanism [26]) *Let $f(D)$ denote a function on D that outputs a vector in $\mathbb{R}^d$. The Laplace Mechanism is:*

$$\mathcal{A}_{LM}(D) = f(D) + (Z_1, ..., Z_d)$$

where Z_i are i.i.d random variables from Laplace$(\Delta f/\epsilon)$.

Above, Laplace(b) denotes the Laplace probability distribution centered at 0 with scale b, and Δf is called the *sensitivity of f* and is the maximum difference in f between any two databases D and D' that differ only by a single record: $\Delta f = \max_{D,D'} \|f(D) - f(D')\|_1$.

The most important case for our use of the Laplace Mechanism is when f is a histogram-generating function, which counts the number of records in a set of disjoint ranges or cat-

egories. In this case, adding or removing a single record to the input will only affect the counts in one of the histogram bins by exactly 1. Thus the sensitivity of f is 1 and the Laplace Mechanism adds random noise sampled from Laplace$(1/\epsilon)$ to each histogram bin and releases the noisy histogram.

Definition 3 (Stability) *Let g be a transformation function that takes a data source (table or vector) as input and returns a new data source (of the same type) as output. For any pair of sources, S and S' let $|S \oplus S'|$ denote the distance between sources. If the sources are both tables, then this distance is the size of the symmetric difference; if the sources are both vectors, then this distance is the L_1 norm; if the sources are of mixed type, it's undefined. Then the stability of g is: $\max_{S,S':|S\oplus S'|=1} |g(S) \oplus g(S')|$. When the stability of g is at most c for some constant c, we say that g is c-stable.*

Composition of differential privacy A key property of differentially private algorithms is composition. The sequential execution of multiple differentially private algorithms is still differentially private, but the parameter measuring privacy loss grows. The resulting privacy loss is equal to the sum of all the privacy losses in the subroutines [65]. This property implies that one cannot simply repeatedly run the Laplace Mechanism to get many noisy results: the use of the privacy loss budget will accumulate with repeated runs and eventually it will be exhausted. In other words, for n algorithms $\mathcal{A}_1$ through $\mathcal{A}_n$, each satisfying ϵ_i-differential privacy, the combined algorithm is ϵ_{total}-differential private where $\epsilon_{total} = \sum_{i=1}^{n} \epsilon_i$[65]. Note that, assuming a user is granted a specified ϵ privacy loss budget, this property implies that one cannot simply repeatedly run the Laplace Mechanism to get many noisy results: the use of the privacy loss budget will accumulate with repeated runs and eventually it will be exhausted.

Post-processing of differential privacy The differential guarantee is also invariant under post-processing. This means that the result of a computation carried out in a differentially

private manner remains differentially private, even if we use it in additional computations *that does not involve the input data*.

Next, we define *linear counting queries*, the class of statistical queries supported in this work. A *condition formula*, ϕ, is a Boolean condition that can be evaluated on any tuple of the schema T. We use $\phi(T)$ to denote the number of tuples in T for which ϕ is true. A number of operators in ϵKTELO answer linear queries over the table. A linear query is the linear combination of any finite set of condition counts:

Definition 4 (Linear counting query (declarative)) *A linear query q on T is defined by conditions $\phi_1 \dots \phi_k$ and coefficients $c_1 \dots c_k \in \mathbb{R}$ and returns $q(T) = c_1\phi_1(T) + \cdots + c_k\phi_k(T)$.*

It is common to consider a vector representation of the database, denoted $\mathbf{x} = [x_1 \dots x_n]$, where x_i is equal to the number of tuples of type i for each possible tuple type in the relational domain of T. The size of this vector, n, is the product of the attribute domains. Then it follows that any linear counting query has an equivalent representation as a vector of n coefficients, and can be evaluated by taking a dot product with $\mathbf{x}$. Abusing notation slightly, let $\phi(i) = 1$ if ϕ evaluates to true for the tuple type i and 0 otherwise.

Definition 5 (Linear counting query (vector)) *For a linear query q defined by $\phi_1 \dots \phi_k$ and $c_1 \dots c_k$, its equivalent vector form is $\vec{q} = [q_1 \dots q_n]$ where $q_i = c_1\phi_1(i) + \cdots + c_k\phi_k(i)$. The evaluation of the linear query is $\vec{q} \cdot \mathbf{x}$, where $\mathbf{x}$ is vector representation of T.*

We will use vectorized representations of the data frequently. We refer to the *domain* as the size of $\mathbf{x}$, the vectorized table. This vector is sometimes large and a number of methods for avoiding its materialization are discussed later.

CHAPTER 3

ϵKTELO: A PROGRAMMING FRAMEWORK FOR EASIER DP ALGORITHM DESIGN AND IMPLEMENTATION

To address the challenges for practitioners to design accurate and private algorithm, we propose ϵKTELO, a programming framework and system that programmers can use to solve statistical tasks that involve answering *linear counting queries*.

In ϵKTELO, differentially private programs are described as *plans* over a high level library of *operators*. Each operator is an abstraction of a key subroutine from a state-of-the-art algorithm. Within ϵKTELO, these operators are organized based on their functionality into a small set of classes: transformation, querying, inference, query selection, and partition selection.

3.1 Overview and Design Principles

In this section we provide an overview of ϵKTELO by presenting an example algorithm written in the framework. Then we discuss the principles guiding the design of ϵKTELO.

3.1.1 An example plan: CDF estimation

In ϵKTELO, differentially private algorithms are described using *plans* composed over a rich library of *operators*. Most of the plans we consider are linear sequences of operators, but ϵKTELO also supports plans with iteration, recursion, and branching. Operators supported by ϵKTELO perform a well defined task and typically capture a key algorithm design idea from the state-of-the-art. Each operator belongs to one of five *operator classes* based on its input-output specification. These are: (a) transformation, (b) query, (c) inference, (d)

Algorithm 1 ϵKTELO CDF Estimator

1: $D \leftarrow$ PROTECTED(source_uri) ▷ Init
2: $D \leftarrow$ WHERE(D, sex == 'M' AND age $\in [30, 39]$) ▷ Transform
3: $D \leftarrow$ SELECT(salary) ▷ Transform
4: $\mathbf{x} \leftarrow$ T-VECTORIZE(D) ▷ Transform
5: $\mathbf{P} \leftarrow$ AHPPARTITION ($\mathbf{x}$, $\epsilon/2$) ▷ Partition Select
6: $\bar{\mathbf{x}} \leftarrow$ V-REDUCEBYPARTITION ($\mathbf{x}$, $\mathbf{P}$) ▷ Transform
7: $\mathbf{M} \leftarrow$ IDENTITY($|\bar{\mathbf{x}}|$) ▷ Query Select
8: $\mathbf{y} \leftarrow$ VECLAPLACE($\bar{\mathbf{x}}$, $\mathbf{M}$, $\epsilon/2$) ▷ Query
9: $\hat{\mathbf{x}} \leftarrow$ NNLS($\mathbf{P}$, $\mathbf{y}$) ▷ Inference
10: $\mathbf{W_{pre}} \leftarrow$ PREFIX($|\mathbf{x}|$) ▷ Query Select
11: **return** $\mathbf{W_{pre}} \cdot \hat{\mathbf{x}}$ ▷ Output

query selection, and (e) partition selection. Operators are fully described in Sec. 3.3 and listed in Fig. 3.1.

First, we describe an example ϵKTELO plan and use it to introduce the different operator classes. Algorithm 1 shows the pseudocode for a plan authored in ϵKTELO, which takes as input a table D with schema [Age, Gender, Salary] and returns the differentially private estimate of the empirical cumulative distribution function (CDF) of the Salary attribute, for males in their 30's. The plan is fairly sophisticated and works in multiple steps. First the plan uses *transformation* operators on the input table D to filter out records that do not correspond to males in their 30's (Line 2), selecting only the salary attribute (Line 3). Then it uses another transformation operator to construct a vector of counts $\mathbf{x}$ that contains one entry for each value of salary. x_i represents the number of rows in the input (in this case males in their 30's) with salary equal to i.

Before adding noise to this histogram, the plan uses a *partition selection* operator, AHPPARTITION (Line 5). Operators in this class choose a partition of the data vector which is later used in a transformation. AHPPARTITION uses the sensitive data to identify a partition $\mathbf{P}$ of the counts in $\mathbf{x}$ such that counts within a partition group are close. Since AHPPARTITION uses the input data, it expends part of the privacy budget (in this case $\epsilon/2$). AHPPARTITION is a key subroutine in AHP [111], which was shown to have state-of-the-art performance for histogram estimation [38].

Next the plan uses V-REDUCEBYPARTITION (Line 6), another transformation operator on $\mathbf{x}$, to apply the partition $\mathbf{P}$ computed by AHPPARTITION. This results in a new reduced vector $\bar{\mathbf{x}}$ that contains one entry for each partition group in $\mathbf{P}$ and the entry is computed by adding up counts within each group.

The plan now specifies a set of measurement queries $\mathbf{M}$ on $\bar{\mathbf{x}}$ using the IDENTITY *query selection* operator (Line 7). The identity matrix corresponds to querying all the entries in $\bar{\mathbf{x}}$ (since $\mathbf{M}\bar{\mathbf{x}} = \bar{\mathbf{x}}$). Query selection operators do not answer any query, but rather specify which queries should be estimated. (This is analogous to how partition selection operators only select a partition but do not apply it.) Next, VECTOR LAPLACE returns differentially private answers to all the queries in $\mathbf{M}$. It does so by automatically calculating the sensitivity of the vectorized queries – which depends on all upstream data transformations – and then using the standard Laplace mechanism (Line 8) to add noise. This operator consumes the remainder of the privacy budget (again $\epsilon/2$).

So far the plan has computed an estimated histogram of partition group counts $\mathbf{y}$, while our goal is to return the empirical CDF on the original salary domain. Hence, the plan uses the noisy counts on the reduced domain $\mathbf{y}$ to infer non-negative counts in the original vector space of $\mathbf{x}$ by invoking an *inference* operator NNLS (short for non-negative least squares) (Line 9). $\text{NNLS}(\mathbf{P}, \mathbf{y})$ finds a solution, $\hat{\mathbf{x}}$, to the problem $\mathbf{P}\hat{\mathbf{x}} = \mathbf{y}$, such that all entries of $\hat{\mathbf{x}}$ are non-negative. Lastly, the plan constructs the set of queries, $\mathbf{W}_{\mathbf{pre}}$, needed to compute the empirical CDF (a lower triangular $n \times n$ matrix representing prefix sums) by calling the query selection operator PREFIX(n) (Line 10), and returns the output (Line 11).

3.1.2 ϵKTELO design principles

The design of ϵKTELO is guided by the following principles.

Expressiveness ϵKTELO is designed to be expressive, meaning that a wide variety of state-of-the-art algorithms can be written succinctly as ϵKTELO plans. To ensure expressiveness, we carefully designed a foundational set of operator classes that cover fea-

tures commonly used by leading differentially private algorithms. We demonstrate the expressiveness of our operators by showing in Sec. 3.4 that the algorithms from the recent DPBench benchmark [38] can be readily re-implemented in ϵKTELO.

Privacy "for free" ϵKTELO is designed so that any plan written in ϵKTELO automatically satisfies differential privacy. The formal statement of this privacy property is in Sec. 3.2.3. This means that plan authors are not burdened with writing privacy proofs for each algorithm they write. Furthermore, when invoking privacy-critical operators that take noisy measurements of the data, the magnitude of the noise is automatically calibrated. As described in Sec. 3.2, this requires tracking all data transformations and measurements and using this information to handle each new measurement request.

Reduced privacy verification effort Ensuring that an algorithm implementation satisfies differential privacy requires verifying that it matches the algorithm specification. The design of ϵKTELO reduces the amount of code that must be vetted each time an algorithm is crafted. First, since an algorithm is expressed as a plan and all plans automatically satisfy differential privacy, the code to be vetted is solely the individual operators. Second, operators need to be vetted only once but may be reused across multiple algorithms. Finally, it is not necessary to vet every operator, but only the privacy-critical ones (as discussed in Sec. 3.2, ϵKTELO mandates a clear distinction between privacy-critical and non-private operators). This means that verifying the privacy of an algorithm requires checking fewer lines of code. In Sec. 3.4, we compare the verification effort to vet the DPBench codebase[1] against the effort required to vet these algorithms when expressed as plans in ϵKTELO.

[1] Available at: `https://github.com/dpcomp-org/dpcomp_core`

Transparency In ϵKTELO, all algorithms are expressed in the same form: each is a plan, consisting a sequence of operators where each operator is selected from a class of operators based on common functionality. This facilitates algorithm comparison and makes differences between algorithms more apparent. In Sec. 3.4, we summarize the plan signatures of a number of state-of-the-art algorithms (pictured in Fig. 3.2). These plan signatures reveal similarities and common idioms in existing algorithms. These are difficult to discover from the research literature or through code inspection.

Efficiency and Scalability Many ϵKTELO plans compute on data vectors formed from projections of an input table. The current implementation of ϵKTELO relies on storing these vectors in memory on a single machine. Even under this restriction, it is challenging to get all ϵKTELO operators to run efficiently. Our specialized matrix representation techniques, presented in Sec. 3.5, allow many of the key operators to scale to large data vectors without imposing undue restrictions on plan authors.

We believe that ϵKTELO, by supporting the design principles described above, provides an improved platform for designing and deploying differentially private algorithms.

3.2 Execution Framework And Privacy Enforcement

This section describes the execution environment and then formalizes the claim that any program executed in ϵKTELO satisfies differential privacy.

3.2.1 Protected Kernel and Client Space

The execution framework consists of an untrusted client space and a protected kernel that encloses the private data. An ϵKTELO program, which we call a plan, runs in the unprotected client space. When the plan needs to interact with the private data, it does so through privileged operators that can issue requests to the protected kernel. Such operators may, for example, request that protected kernel apply a data transformation or perhaps return a noisy measurement. The protected kernel services requests from privileged operators, only

executing them if their cost is within the available privacy budget. The distinction between the client space and the protected kernel is a fundamental one in ϵKTELO. It allows authors to write plans that consist of operator calls embedded in otherwise arbitrary code (which may freely include conditionals, loops, recursion, etc.).

The protected kernel is initialized by specifying a single protected data object—an input table D—and a global privacy budget, which we denote as ϵ_{tot}. Note that requests for data transformations may cause the protected kernel to derive additional data sources. Thus, the protected kernel maintains a *data source environment*, which consists of a mapping between data source variables, which are exposed to the client, and the protected data objects, which are kept private. In addition, the data source environment tracks the transformation lineage of each data source. It also maintains the stability of each transformation (defined in Chapter 2). Note that in describing operators (Sec. 3.3), we speak informally of operators having data sources as inputs and outputs rather than data source *variables*. A layer of indirection is always maintained in the implementation but sometimes elided in our descriptions to simplify the presentation.

3.2.2 Operator types

Operators have one of three types, based on their interaction with the protected kernel. The first type is a **Private** operator, which requests that the protected kernel perform some action on the private data (e.g., a transformation) but receives only an acknowledgement that the operation has been performed. The second type is a **Private→Public** operator, which receives information about the private data (e.g., a measurement) and thus consumes privacy budget. The last type is a **Public** operator, which does not interact with the protected kernel at all and can be executed entirely in client space. An example of a public operator would operators that perform inference on the noisy measurements received from the protected kernel. When describing operators in Sec. 3.3, we color code them based on their type.

3.2.3 Privacy Guarantee

In this section, we state the privacy guarantee offered by ϵKTELO. Informally, ϵKTELO ensures that if the protected kernel is initialized with a source database D and a privacy budget ϵ_{tot}, then any plan (chosen by the client) will satisfy ϵ_{tot}-differential privacy with respect to D. Note that if the client exhausts the privacy budget, subsequent calls to Private→Public operators will return an exception, indicating that they are not permitted. Importantly, an exception itself does not leak sensitive information – i.e., the decision to return an exception does not depend on the private state.

A *transcript* is a sequence of operator calls and their responses. Formally, let $r_k = \langle op_1, a_1, \ldots, op_k, a_k \rangle$ denote a length k sequence where op_i is an operator call and a_i the response. We assume that the value of op_i is a deterministic function of $a_1, \ldots, a_{i-1}$. We use $R_k = r_k$ to denote the event that the first k operations result in transcript r_k. Let $\mathcal{R}_k$ be the set of all possible transcripts of length k. We assume that all Private→Public operators output values from an arbitrary, but finite set. Thus, the set of possible transcripts is finite. Let $P(R_k = r_k \mid \textbf{Init}(D, \epsilon_{tot}))$ be the conditional probability of event $R_k = r_k$ given that the system was initialized with input D and a privacy budget of ϵ_{tot}.

Theorem 3.2.1 (Privacy of ϵKTELO plans) *Let D, D' be any two instances such that $|D \oplus D'| = 1$. For all $k \in \mathbb{N}^+$ and $r_k \in \mathcal{R}_k$,*

$$P(R_k = r_k \mid \textbf{Init}(D, \epsilon_{tot})) \leq \exp(\epsilon_{tot}) \times P(R_k = r_k \mid \textbf{Init}(D', \epsilon_{tot})).$$

The proof of Theorem 3.2.1, which appears in the sequel, extends the proof in [27] to support the V-SplitByPartition operator.

While ϵKTELO ensures differential privacy, private information could be leaked via side-channel attacks (e.g., timing attacks). Privacy engineers who design operators are responsible for protecting against such attacks; an analysis of this issue is beyond the scope of this dissertation.

3.2.4 Privacy Proof

This section presents a proof of Theorem 3.2.1. We start by introducing some supporting concepts and notation. (Some notation is adapted from [27].)

Information tracked by the protected kernel The protected kernel maintains the following state, which we denote as S_{kernel}:

- A set of source variables SV.
- A data source environment E maps each source variable $sv \in SV$ to an actual data source S, as in $E(sv) = S$. (Recall that sources can be tables or vectors.)
- A transformation graph: the nodes are SV and there is an edge from sv to sv' if sv' was derived via transformation from sv. (Note: a partition transformation introduces a special dummy data source variable whose parent is the source variable being partitioned and whose children are the variables associated with each partition.)
- A stability tracker St maps each source variable $sv \in SV$ to a non-negative number: $St(sv)$ represents the stability of the transformation that derived data source sv from the initial source, or 1 if sv is the initial source.
- A budget consumption tracker B that maps each source variable $sv \in SV$ to a non-negative number: $B(sv)$ represents the total budget consumption made by queries to sv or *to any source derived from* sv.
- A query history $\mathcal{Q}$ that maps each source variable to information about the state of queries asked about sv or any of its descendants. Specifically, for $sv\ in SV$, $\mathcal{Q}(sv)$ returns of a set of tuples (q, s, σ, v) where the meaning of the tuple is that query q was executed on data source s (which is sv or one of its descendants) with σ noise, the result was v. In the context of the proof a query is any Private→Public operator. Such an operator is assumed to satisfy ϵ-differential privacy with respect to the data source on which it is applied.
- The global privacy budget, denoted ϵ_{tot}.

When the protected kernel is initialized, as in **Init**(T, ϵ_{tot}), it sets global budget to ϵ_{tot}, creates new source variable sv_{root}, sets $E(sv_{root}) = T$, sets $St(sv_{root}) = 1$, and $B(sv_{root}) = 0$, and adds sv_{root} to the transformation graph.

Budget Management When a query request is issued to the protected kernel, the protected kernel uses Algorithm 2 to check whether the query can be answered given the available privacy budget.

Algorithm 2 An algorithm for budget requests

procedure REQUEST(sv, σ)
  if sv is the root **then**
    If $B(sv) + \sigma > \epsilon_{tot}$, return FALSE. Otherwise $B(sv)$ += σ and return TRUE.
  else if sv is a partition variable **then**
    Let sv_{child} be the child from which the request came..
    Let $r = \max\{ B(sv_{child}) + \sigma - B(sv), 0 \}$
    Let ans = REQUEST(parent(sv), r).
    If ans = FALSE, return FALSE. Otherwise, $B(sv)$ += r and return TRUE.
  else
    ans = REQUEST($parent(sv)$, $s \cdot \sigma$) ▷ s is stability factor of sv wrt its parent
    if ans = FALSE, return FALSE.
    $B(sv)$ += σ. Return TRUE.
  end if
end procedure

Configurations A configuration, denoted $\mathbb{C} = \langle S_{client}, S_{kernel} \rangle$, captures the state of the client, denoted S_{client}, and the state of the protected kernel, denoted S_{kernel}. The client state can be arbitrary, but state updates are assumed to be deterministic.

We can define the similarity of two configurations $\mathbb{C}$ and $\mathbb{C}'$ as follows. (Notation: we use X' to refer to component X of configuration $\mathbb{C}'$.) We say that $\mathbb{C} \sim \mathbb{C}'$ iff $S_{client} = S'_{client}$ and $S'_{kernel} \sim S'_{kernel}$ where $S_{kernel} \sim S'_{kernel}$ iff $SV = SV'$ and the transformation graphs are identical and for each $sv \in SV$ the following conditions hold:

- $St(sv) = St'(sv)$, $B(sv) = B'(sv)$, $\mathcal{Q}(sv) = \mathcal{Q}'(sv)$, and $\epsilon_{tot} = \epsilon'_{tot}$.
- $|E(sv) \oplus E'(sv)| \leq St(sv) = St'(sv)$ where $|x \oplus y|$ is measured as symmetric difference when the sources x and y are tables and L_1 distance for vectors.)

We introduce a lemma that bounds the difference probability between query answers. Let $P(q(E(s), \sigma) = v)$ denote the probability that query operator q when applied to data source $E(s)$ with noise σ returns answer v.

Lemma 3.2.2 *Let $\mathbb{C} \sim \mathbb{C}'$. For any $sv \in SV$ with non-empty $\mathcal{Q}(sv)$, the following holds:*

$$\prod_{(q,s,\sigma,v)\in\mathcal{Q}(sv)} P(q(E(s),\sigma) = v) \tag{3.1}$$
$$\leq \exp(B(sv) \times |E(sv) \oplus E'(sv)|) \times \prod_{(q,s,\sigma,v)\in\mathcal{Q}'(sv)} P(q(E'(s),\sigma) = v)$$

Proof 3.2.3 *Proof by induction on a reverse topological order of the transformation graph.*

Base case*: Consider a single sv at the end of the topological order (therefore it has no children). If $\mathcal{Q}(sv)$ is empty, it holds trivially. Assume non-empty. Consider any $(q, s, \sigma, v) \in \mathcal{Q}(sv)$. Since sv has no children, then $s = sv$. Furthermore, because the only budget requests that apply to sv are from direct queries, we have (according to Algorithm 2), $B(sv) = \sum_{(q,s,\sigma,v)\in\mathcal{Q}(sv)} \sigma$. Since we assume that any query operator satisfies ϵ-differential privacy with respect to its source input, we have $P(q(E(s), \sigma) = v) \leq P(q(E'(s), \sigma) = v) \times \exp(\sigma \times |E(s) \oplus E'(s)|)$. Substituting sv for s and taking the product over all terms in $\mathcal{Q}(sv)$, we get Eq.* (3.1).

Inductive case*: Assume Eq.* (3.1) *holds for all nodes later in the topological order. Therefore it holds for any child c of sv. We can combine the inequalities for each child into the following inequality over all children,*

$$\prod_{c\in children(sv)} \prod_{(q,s,\sigma,v)\in\mathcal{Q}(c)} P(q(E(s),\sigma) = v)$$
$$\leq \prod_{c\in children(sv)} \exp(B(c) \times |E(c) \oplus E'(c)|) \times \prod_{(q,s,\sigma,v)\in\mathcal{Q}(c)} P(q(E'(s),\sigma) = v)$$
$$= \exp\left(\sum_{c\in children(sv)} B(c) \times |E(c) \oplus E'(c)|\right)$$
$$\times \prod_{c\in children(sv)} \prod_{(q,s,\sigma,v)\in\mathcal{Q}(c)} P(q(E'(s),\sigma) = v)$$

There are two cases, depending what type of table variable sv is.

First, consider the case when sv is not *a special partition variable. We know by transformation stability that $|E(c) \oplus E'(c)| \leq s \times |E(sv) \oplus E'(sv)|$ where s is the stability factor for the transformation. In addition, $\sum_c B(c) \times s \leq B(sv)$ because, according to*

Algorithm 2, every time a request of σ is made to child c, a request of $s \times \sigma$ is made to sv. Therefore,

$$\sum_{c \in children(sv)} B(c) \times |E(c) \oplus E'(c)| \leq \sum_{c \in children(sv)} B(c) \times s \times |E(sv) \oplus E'(sv)|$$
$$\leq B(sv) \times |E(sv) \oplus E'(sv)|$$

Furthermore, observe that each term in $(q, s, \sigma, v) \in \mathcal{Q}(c)$ also appears in $\mathcal{Q}(sv)$. In addition, $\mathcal{Q}(sv)$ includes any queries on sv directly (and we know from an argument similar to the base case that Eq. (3.1) *holds for these queries). Therefore Eq.* (3.1) *holds on sv.*

Now, consider the case where sv is the special partition variable. Let $m = \max_c B(c)$. We get the following

$$\sum_{c \in children(sv)} B(c) \times |E(c) \oplus E'(c)| \leq \sum_{c \in children(sv)} m \times |E(c) \oplus E'(c)|$$
$$= m \times \sum_{c \in children(sv)} |E(c) \oplus E'(c)| = m \times |E(sv) \oplus E'(sv)|$$
$$= B(sv) \times |E(sv) \oplus E'(sv)|$$

The second to last line follows from the fact that sv is partition transformation. The last line follows from how $B(sv)$ is updated according Algorithm 2.

Main Proof We use $\mathbb{C}_0(T, \epsilon_{tot}, P_0)$ to denote the initial configuration in which the protected kernel has been initialized with **Init**(T, ϵ_{tot}) and the client state is initialized to P_0. We use the notation $\mathbb{C}_0(T, \epsilon_{tot}, P_0) \stackrel{t}{\Rightarrow}_p \mathbb{C}$ to mean that starting in $\mathbb{C}_0$ after t operations, the probability of being in configuration $\mathbb{C}$ is p.

Theorem 3.2.4 *If $T \sim_1 T'$ and $\mathbb{C}_0(T, \epsilon_{tot}, P_0) \stackrel{t}{\Rightarrow}_p \mathbb{C}$ such that $B(sv_{root}) = \epsilon$ in $\mathbb{C}$, then $\epsilon \leq \epsilon_{tot}$ and there exists $\mathbb{C}'$ such that $\mathbb{C}_0(T', \epsilon_{tot}, P_0) \stackrel{t}{\Rightarrow}_q \mathbb{C}'$ where $\mathbb{C} \sim \mathbb{C}'$ and $p \leq q \cdot \exp(\epsilon)$.*

Theorem 3.2.1 follows as a corollary from Theorem 3.2.4.

Proof 3.2.5 *Proof by induction on t.*

Base case*: $t = 0$. This implies that $p = q = 1$, $\epsilon = 0$, and $\mathbb{C} = \mathbb{C}_0(T, \epsilon_{tot}, P_0)$ and $\mathbb{C}' = \mathbb{C}_0(T', \epsilon_{tot}, P_0)$. It follows that $\mathbb{C} \sim \mathbb{C}'$ because we are given that $T \sim_1 T'$ and the rest of the claim follows.*

Inductive case*: Assume the claim holds for t, we will show it holds for $t + 1$. Let $\mathbb{C}_1$ be any configuration such that $\mathbb{C}_0(T, \epsilon_{tot}, P_0) \stackrel{t}{\Rightarrow}_{p_1} \mathbb{C}_1$ where in $\mathbb{C}_1$, we have $B(sv_{root}) = \epsilon_1$.*

The inductive hypothesis tells us that $\epsilon_1 \leq \epsilon_{tot}$ and that there exists a $\mathbb{C}'_1$ such that $\mathbb{C}_0(T', \epsilon_{tot}, P_0) \stackrel{t}{\Rightarrow}_{q_1} \mathbb{C}'_1$ and $\mathbb{C}_1 \sim \mathbb{C}'_1$ and $p_1 \leq q_1 \times \exp(\epsilon_1)$.

Because $\mathbb{C}_1 \sim \mathbb{C}'_1$, it follows that the client is in the same state and so the next operation request from the client will be the same in $\mathbb{C}_1$ and $\mathbb{C}'_1$. The proof requires a case analysis based on the nature of the operator. We omit analysis of transformation operators or operators that are purely on the client side as those cases are straightforward: essentially we must show that the appropriate bookkeeping is performed by the protected kernel. We focus on the case where the operator is a query operator.

For a query operator, there are two cases: (a) running out of budget, and (b) executing a query. For the first case, by the inductive hypothesis $\mathbb{C}_1 \sim \mathbb{C}'_1$ and therefore if executing Algorithm 2 yields False on the protected kernel state in $\mathbb{C}_1$, it will also do so on the protected kernel state in $\mathbb{C}'_1$. For the second case, suppose query q is executed on source sv with noise σ and answer v is obtained. The protected kernel adds the correpsonding entry to the query history $\mathcal{Q}$. Let $\mathbb{C}$ denote the resulting state. Let $\mathbb{C}'$ correspond to extending $\mathbb{C}'_1$ in a similar way. Thus $\mathbb{C} \sim \mathbb{C}'$.

It remains to show two things. First, letting $B(sv_{root}) = \epsilon$, we must show that $\epsilon \leq \epsilon_{tot}$. This follows from Algorithm 2 which does not permit $B(sv_{root})$ to exceed ϵ_{tot}. Second, we must bound the probabilities. Suppose that the probability of this query answer in $\mathbb{C}$ is p_2 and the probability of this answer on $\mathbb{C}'$ is q_2. It remains to show that $p_1 \cdot p_2 \leq \exp(\epsilon) \cdot q_1 \cdot q_2$. For this we rely on Lemma 3.2.2 applied to sv_{root} with the observations that the product of

Transform	
TV	T-Vectorize
TW	T-Where
TPR	T-Project
TP	V-SplitByPartition
TR	V-ReduceByPartition

Inference	
LS	Least squares
NLS	Nneg Least squares
MW	Mult Weights
HR	Thresholding

Partition selection	
PA	AHPpartition
PG	Grid
PD	Dawa
PW	Workload-based
PS	Stripe(attr)
PM	Marginal(attr)
PU	UniformPartition

Query	
LM	Vector Laplace

Query selection	
SI	Identity
ST	Total
SP	Privelet
SH2	H2
SHB	HB
SG	Greedy-H
SU	UniformGrid
SA	AdaptiveGrids
SQ	Quadtree
SHD	HDMM
SS	Stripe(attr)
SW	Worst-approx
SPB	PrivBayes select

Figure 3.1: The operators currently implemented in ϵKTELO. Private operators are red, Private→Public operators are orange, and Public operators are green.

probabilities bounded in Lemma 3.2.2 corresponds to the probabilities in $p_1 \cdot p_2$ that do not trivially equal 1 and that $|E(sv_{root}) \oplus E'(sv_{root})| = 1$.

3.3 Operators and Operator Classes

We now describe in detail the operators and operator classes in ϵKTELO. A full list of operators is shown in Fig. 3.1 where they are arranged into classes and color-coded by type (Private, Private→Public, or Public). Along with descriptions of the operator classes we explain their role in plans and prove supporting properties.

3.3.1 Transformation Operators

Transformation operators take as input a data source variable (either a table or a vector) and output a transformed data source (again, either a table or vector). Transformation operators modify the data held in the kernel, without returning answers. So while they do not expend privacy budget, they can affect the privacy analysis through their *stability* (Chapter 2). Every transformation in ϵKTELO has a well-established stability.

Table Transformations

ϵKTELO supports table transformations PROJECT and WHERE, each with stability 1. The definitions of the operators are nearly identical to those described in PINQ [64] and are not repeated here. As ϵKTELO currently handles programs that use linear queries on single tables, the JOIN operator is not yet supported.

Vectorization

All of the plans in ϵKTELO start with table transformations and typically transform the resulting table into a vector using T-VECTORIZE (and all later operations happen on vectors). The T-VECTORIZE operator is a transformation operator that takes as input a table T and outputs a vector $\mathbf{x}$ that has as many cells as the number of elements in the table's domain (recall the discussion of domain Chapter 2). Each cell in $\mathbf{x}$ represents the number of records in the table that correspond to the domain element encoded by the cell. T-VECTORIZE is a 1-stable transformation.

The vectorize operation can significantly impact the performance of the code, especially in high-dimensional cases, as we represent one cell per element in the domain. For this reason we allow table transformations (WHERE and PROJECT) to reduce the domain size before running T-VECTORIZE. One of the primary reasons for working with the vector representation is to allow for inference operators downstream. Once in vector form, data can be further transformed as described next.

Vector Transformations

ϵKTELO supports transformations on vector data sources. Each vector transformation takes as input a vector $\mathbf{x}$ and a matrix $\mathbf{M}$ and produces a vector $\mathbf{x}' = \mathbf{M}\mathbf{x}$. The linearity of vector transformations is an important feature that is leveraged by downstream inference operators. The stability of vector transformations is equal to the largest L_1 column norm of $\mathbf{M}$.

The V-REDUCEBYPARTITION operator is a 1-stable vector transformation operator that reduces the dimensionality of the data vector $\mathbf{x}$ by eliminating cells from $\mathbf{x}$ or grouping

together cells in $\mathbf{x}$. Such transformations are useful to (a) filter out parts of the domain that are uninteresting for the analyst, (b) reduce the size of the $\mathbf{x}$ vector so that algorithm performance can be improved, and (c) reduce the number of cells in $\mathbf{x}$ so that the amount of noise added by measurement operators is reduced.

V-REDUCEBYPARTITION takes as input a partition defining a grouping of the cells in the $\mathbf{x}$. It can be carried out by representing the partition as a $(p \times n)$ matrix $\mathbf{P}$ where n is the number of cells in $\mathbf{x}$, p is the number of groups in the partition, and $P_{ij} = 1$ if cell j in $\mathbf{x}$ is mapped to group i, and 0 otherwise.

The V-SPLITBYPARTITION operator is the vector analogue of the tabular SPLITBY-PARTITION operator. It takes as input a partition and splits the data vector $\mathbf{x}$ into k vectors, $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}$, each representing a disjoint subset of the original domain. This operator allows us to create different subplans for disjoint parts of the domain. This is a 1-stable vector transform. (Note: V-SPLITBYPARTITION can be expressed as k linear transforms with matrices that select the appropriate elements of the domain for each partition.)

3.3.2 Query Operators

Query operators are responsible for computing noisy answers to queries on a data source. Since answers are returned, query operators necessarily expend privacy budget. Query operators take a data source variable and ϵ as input.

For tables, the NOISYCOUNT operator takes as input a table D and ϵ and returns $|D|+\eta$, where η is drawn from the Laplace distribution with scale $1/\epsilon$. For vectors, the VECTOR LAPLACE operator takes as input a vector $\mathbf{x}$, epsilon, and a set of linear counting queries $\mathbf{M}$ represented in matrix form. Let $\mathbf{M}$ be a matrix of size $(m \times n)$. VECTOR LAPLACE returns $\mathbf{Mx} + \frac{\sigma(\mathbf{M})}{\epsilon}\mathbf{b}$ where $\mathbf{b}$ is a vector of m independently drawn Laplace random variables with scale 1 and $\sigma(\mathbf{M})$ is the maximum L_1 norm of the columns of $\mathbf{M}$.

For both query operators, it is easy to show they satisfy ϵ-differential privacy with respect to their data source input [64, 53]. Note, however, in the case the source is derived

from other data sources through transformation operators, the total privacy loss could be higher. The cumulative privacy loss depends on the stability of the transformations and is tracked by the protected kernel.

3.3.3 Query Selection Operators

Since each query operation consumes privacy budget, the plan author must be judicious about what queries are being asked. Recent privacy work has shown that if the plan author's goal is to answer a *workload* of queries, simply asking these queries directly can lead to sub-optimal accuracy (e.g., when workload queries ask about overlapping regions of the domain). Instead, higher accuracy can be achieved by designing a *query strategy*, a collection of queries whose answers can be used to reconstruct answers to the workload. This approach was formalized by the matrix mechanism [53], and has been a key idea in many algorithms [52, 49, 79, 40, 101, 19, 61]. Among these the recent HDMM algorithm is notable because it uses an optimization-based approach to find the query strategy that most-effectively answers the workload. HDMM effectively scales to multi-dimensional domains, and offers state-of-the-art utility on many workloads [61].

A query selection operator is distinguished by its output type: a set of linear counting queries $\mathbf{M}$ represented in matrix form (i.e., the matrix input to the VECTOR LAPLACE operator described above). As Fig. 3.1 indicates, ϵKTELO supports a large number of query selection operators, most of which are extracted from algorithms proposed in the literature. While these operators agree in terms of their output, they vary in terms of their input: some employ fixed strategies that depend only on the size of $\mathbf{x}$ (e.g., IDENTITY and PREFIX in Algorithm 1), some adapt to the workload (e.g., GREEDY-H), some depend on prior measurements (e.g., ADAPTIVEGRIDS), etc.

Most query selection operators only rely on non-private information (domain size, workload) and therefore are of Public type. But there are a few that consult the private data, and thus have the Private→Public type. For example, WORST-APPROX is an operator

that picks the query from a workload that is the worst approximated by a current estimate of the data. Such an operator is used by iterative algorithms like MWEM [36]. Another is PRIVBAYES SELECT, an operator that privately constructs a Bayes net over the attributes of the data source, and then returns a matrix corresponding to the sufficient statistics for fitting the parameters of the Bayes net. This was used as a subroutine in PrivBayes [109].

3.3.4 Partition Selection Operators

Partition selection operators compute a matrix $\mathbf{P}$ which can serve as the input to the V-REDUCEBYPARTITION and V-SPLITBYPARTITION operators described earlier. Of course the matrix $\mathbf{P}$ must be appropriately structured to be a valid partition of $\mathbf{x}$.

This is an important operator class since much of recent innovation into state-of-the-art algorithms for answering histograms and range queries has used partitions to either reduce the domain size of the data vector by grouping together cells with similar counts, or split the data vector into smaller vectors and leverage the parallel composition of differential privacy to process each subset of the domain independently. ϵKTELO includes partition selection operators AHPPARTITION and DAWA which are subroutines from the AHP [111] and DAWA [49] algorithms, respectively. Both of these operators are data adaptive, and hence are Private→Public. We also introduce new partition selection operators, WORKLOAD-BASED and STRIPE, described in Secs. 3.6.1 and 3.7.2 respectively.

3.3.5 Inference Operators

An inference operator derives new estimates to queries based on the history of transformations and query answers. Inference operators never use the input data directly and hence are Public. Plans typically terminate with a call to an inference operator to estimate a final set of query answers reflecting all available information computed during execution of the plan. Some plans may also perform inference as the plan executes.

Ideally, an inference method should: (i) properly account for measurements with unequal noise; (ii) support inference over incomplete measurements (in which derived an-

swers are not completely determined by available measurements); (iii) should incorporate all available information (including a prior or constraint on the input dataset); and lastly, (iv) inference should efficiently scale to large domains. Many versions of inference have been considered in the literature [40, 51, 48, 36, 3, 111, 79, 76, 100] but none meet all of the objectives above. ϵKTELO currently supports multiple inference methods, in part to support algorithms from past work and in part to offer necessary tradeoffs among the properties above.

All the inference operators supported in ϵKTELO take as input a set of queries, represented as a matrix $\mathbf{M}$, and noisy answers to these queries, denoted $\mathbf{y}$. The output of inference is a data vector $\hat{\mathbf{x}}$ that best fits the noisy answers—i.e., an $\hat{\mathbf{x}}$ such that $\mathbf{M}\hat{\mathbf{x}} \approx \mathbf{y}$. The estimated $\hat{\mathbf{x}}$ can then be used to derive an estimate of any linear query $\mathbf{q}$ by computing $\mathbf{q} \cdot \hat{\mathbf{x}}$. The inference operator may optionally take as input a set of weights, one per query (row) in $\mathbf{M}$ to account for queries with different noise scales.

ϵKTELO supports two variants of least squares inference, the most widely used form of inference in the current literature [40, 51, 79]. ϵKTELO extends these methods and formulates them as general operators, allowing us to replicate past algorithms, and consider new forms of inference that support constraints. The first variant solves a classical least squares problem:

Definition 6 (Ordinary least squares (LS))

$$\hat{\mathbf{x}} = \underset{x \in \mathbb{R}^n}{\arg\min} \|\mathbf{M}\mathbf{x} - \mathbf{y}\|_2 \tag{3.2}$$

Our second variant imposes a non-negativity constraint on $\hat{\mathbf{x}}$:

Definition 7 (Non-negative least squares (NNLS)) *Given scaled query matrix* $\mathbf{M}$ *and answer vector* $\mathbf{y}$*, the non-negative least squares estimate of* $\mathbf{x}$ *is:*

$$\hat{\mathbf{x}} = \underset{x \succeq 0}{\arg\min} \|\mathbf{M}\mathbf{x} - \mathbf{y}\|_2 \tag{3.3}$$

These inference methods can also support some forms of prior information, particularly if it can be represented as a linear query. For example, if the total number of records in the input table is publicly known, or other special queries have publicly available answers, they can be added as "noisy" answers with negligible noise scale and they will naturally incorporated into the inference process and the derivation of new query estimates.

We also support an inference method based on a multiplicative weights update rule, which is used in the MWEM [36] algorithm. This inference algorithm is closely related to the principle of maximum entropy, and is especially effective when one has measured an incomplete set of queries.

Defining inference under vector transformations

Recall that in the discussion above we describe inference as operating on a single vector $\mathbf{x}$ with a corresponding query matrix $\mathbf{M}$. However, plans can include an arbitrary combination of vector transformations, followed by query operators, resulting in a collection of query answers defined over various vector representations of the data. ϵKTELO handles this by taking advantage of the structure of vector transformations and query operators, both of which perform linear transformations, therefore making it possible to map measured queries back on to the original domain (i.e., a vector produced by the VECTORIZE operation) and perform inference there. This allows for the most complete form of inference but other alternatives are conceivable, for example by performing inference locally on transformed vectors and combining inferred queries. This might have efficiency advantages, but would likely sacrifice accuracy, and is left for future investigation.

Inference: impact on accuracy

Because inference is an operator in ϵKTELO algorithm authors are encouraged to use inference consistently, using all available measurements, even if they are measured in different parts of a plan. In contrast, some existing algorithms use inference in an ad-hoc manner, performing inference on one set of measurements separately from another set of

measurements. As we show below, for unbiased plans, this is always sub-optimal and ϵKTELO helps to relieves the algorithm designer of the complexity of integrating measured information properly. The following theorem follows the intuition that any unbiased noisy measurement provides information about the true data that can lower error, in expectation:

Theorem 3.3.1 *Given any (full rank) matrix* $\mathbf{M}$ *of linear measurements and any linear query* $\mathbf{q}$*, the expected error of* $\mathbf{q}$ *is never higher if we include additional linear measurements using least squares inference.*

Proof 3.3.2 *Assuming all measurments have variance* 1*, the expected error on a query* $\mathbf{q}$ *is* $Error_{\mathbf{M}}(\mathbf{q}) = \mathbf{q}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{q}^T$ *[51]. In general, the variance of the measurements depends on the privacy budget and the sensitivity of* $\mathbf{M}$*, but they can always be scaled to have variance* 1*. If we augment* $\mathbf{M}$ *with a new linear query* $\mathbf{b}$*, it becomes* $\mathbf{M}' = \begin{bmatrix} \mathbf{M} \\ \mathbf{b} \end{bmatrix}$*. We can write* $\mathbf{M}'^T\mathbf{M}' = \mathbf{M}^T\mathbf{M} + \mathbf{b}^T\mathbf{b}$ *where* $\mathbf{b}^T\mathbf{b}$ *is the outer product. Using the Sherman-Morrison formula [91], we see that*

$$(\mathbf{M}'^T\mathbf{M}')^{-1} = (\mathbf{M}^T\mathbf{M})^{-1} - \frac{1}{1 + \mathbf{b}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{b}^T}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{b}^T\mathbf{b}(\mathbf{M}^T\mathbf{M})^{-1}$$

Since $(\mathbf{M}^T\mathbf{M})^{-1}$ *is positive-definite,* $\mathbf{b}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{b}^T \geq 0$ *and the fraction is just some positive constant* c*. With some algebraic manipulation, we arrive at the following expression:*

$$Error_{\mathbf{M}'}(\mathbf{q}) = Error_{\mathbf{M}}(\mathbf{q}) - c\mathbf{q}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{b}^T\mathbf{b}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{q}^T$$

Letting $v = \mathbf{q}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{b}^T$*, we can write that as* $Error_{\mathbf{M}'}(\mathbf{q}) = Error_{\mathbf{M}}(\mathbf{q}) - cv^2$ *since* $(\mathbf{M}^T\mathbf{M})^{-1}$ *is symmetric. Clearly* cv^2 *is non-negative, so* $Error_{\mathbf{M}'}(\mathbf{q}) \leq Error_{\mathbf{M}}(\mathbf{q})$*. This completes the proof.*

3.4 Expressing known algorithms

As stated in Sec. 3.1, ϵKTELO represents differentially private algorithms as plans composed over a rich library of operators, and supports not only simple linear sequences but also plans with iteration, recursion and branching. To highlight the expressiveness of ϵKTELO, we re-implemented state-of-the-art algorithms as ϵKTELO plans. Once the necessary operators are implemented, the plan definition for an existing algorithm is typically a few lines of code for combining operators and managing parameters. We performed extensive testing to confirm that reimplementations in ϵKTELO of existing algorithms provide statistically equivalent outputs.

ID	Cite	Algorithm name	Plan signature
1	Dwork et al. 2006	Identity	SI LM
2	Xiao et al. 2010	Privelet	SP LM LS
3	Hay et al. 2010	Hierarchical (H2)	SH2 LM LS
4	Qardaji et al. 2013	Hierarchical Opt (HB)	SHB LM LS
5	Li et al. 2014	Greedy-H	SG LM LS
6	-	Uniform	ST LM LS
7	Hardt et al. 2012	MWEM	I:(SW LM MW)
8	Zhang et al. 2014	AHP	PA TR SI LM LS
9	Li et al. 2014	DAWA	PD TR SG LM LS
10	Cormode et al. 2012	Quadtree	SQ LM LS
11	Qardaji et al. 2013	UniformGrid	SU LM LS
12	Qardaji et al. 2013	AdaptiveGrid	SU LM LS PU TP[SA LM] LS
13	McKenna et al. 2018	HDMM	SHD LM LS
14	NEW	DAWA-Striped	PS TP[PD TR SG LM] LS
15	NEW	HB-Striped	PS TP[SHB LM] LS
16	NEW	HB-Striped_kron	SS LM LS
17	NEW	PrivBayesLS	SPB LM LS
18	NEW	MWEM variant b	I:(SW SH2 LM MW)
19	NEW	MWEM variant c	I:(SW LM NLS)
20	NEW	MWEM variant d	I:(SW SH2 LM NLS)

Figure 3.2: The high-level signatures of plans implemented in ϵKTELO (referenced by ID). All plans begin with a vectorize transformation, omitted for readability. We also omit parameters of operators, including ϵ budget shares. `I`(*subplan*) refers to iteration of a subplan and `TP`[*subplan*] means that *subplan* is executed on each partition produced by `TP`.

We examined 12 differentially private algorithms for answering low dimensional counting queries that were deemed competitive[2] in a recent benchmark study [38], and one new algorithm published after the benchmark study [61]. PLAN #1-13 in Fig. 3.2 abstract their ϵKTELO implementations as *plan signatures* where operators are represented using colored abbreviations.

3.4.1 Re-implementing existing algorithms

The algorithms are listed roughly in the order in which they were proposed in the literature and reflect the evolution of increasingly complex algorithmic techniques. The simplest Algorithm, IDENTITY [26], is a natural application of the Laplace mechanism. It simply measures each component of the data vector. Algorithms 2 through 5 reflect the evolution of more sophisticated measurements selection, targeted toward specific workloads. Many of these techniques were originally designed to support range queries (a small subclass of linear queries) over one- or two-dimensional data. PRIVELET [101] uses a Haar wavelet as its measurements, which allows for sensitivity that grows logarithmically with the domain size, yet allows accurate reconstruction of any range query. The HIERARCHICAL (H2) technique uses measurements that form a binary tree over the domain, achieving effects similar to the wavelet measurements. QUADTREE [19] is the 2-dimensional realization of the hierarchical structures. All the algorithms above follow similar design idioms, allowing us to implement them using operator sequences of the same pattern: *Query selection, Query, and Inference*.

All of the algorithms above are *data-independent*, with constant error rates for any input dataset. More recent algorithms are *data-dependent*, displaying different error rates on different inputs, often because the algorithmic techniques are adapting to features of the data to lower error. The simplest data-dependent algorithm is UNIFORM which simply

[2]This is the subset of algorithms that offered the best accuracy for at least one of the input settings of the benchmark.

estimates the total number of records in the input and assumes uniformity across the data vector. This simple algorithm also follows the simple pattern.

A more complex example is the Multiplicative-Weights Exponential Mechanism (MWEM) [36] which takes a workload of linear queries as input and runs several rounds of estimation, measuring one workload query in each round, and using the multiplicative update rule to revise its estimate of the data vector. In each round, the Exponential Mechanism is used to select the workload query that is most poorly approximated using the current data vector estimate.

Algorithm 3 MWEM

1: $D \leftarrow$ PROTECTED(source_uri) ▷ Init
2: $\hat{x} \leftarrow$ T-VECTORIZE(D) ▷ Transform
3: **for** $i = 1 : T$ **do**
4: $\quad \mathbf{M} \leftarrow$ WORSTAPPROX($\hat{x}$, $\epsilon/2T$) ▷ Query Selection
5: $\quad \mathbf{y} \leftarrow$ VECTORLAPLACE($\mathbf{M}$, $\epsilon/2T$) ▷ Query
6: $\quad \hat{\mathbf{x}} \leftarrow$ MULTWEIGHTS($\mathbf{M}$, $\mathbf{y}$) ▷ Inference
7: **end for**
8: **return** $\hat{\mathbf{x}}$ ▷ Output

In Algorithm 3, we rewrite the algorithm with abstracted subroutines and find out that this algorithm can be represented as several iterations of the *Query selection, Query, and Inference* sequence. In Fig. 3.2, the iteration inherent to PLAN #7 (MWEM) is shown with $I : (..)$.

Other data-dependent algorithms exploit partitioning, in which components of the data vector are merged and estimated only in their entirety, which uniformity assumption imposed within the regions. The DAWA [49] and AHP [111] algorithms have custom partition selection methods which consume part of the privacy budget to identify approximately uniform partition blocks. The partition selection methods work by finding a grouping of the bins in a vector and are the key innovations of the algorithms. We encapsulate these subroutines as new operators in our framework (in the cases above, we added a partition selection operators DAWAPARTITION (PD) in PLAN #9 and AHPPARTITION (PA) in PLAN #8).

UNIFORMGRID and ADAPTIVEGRID [78] focus on 2D data and both end up with partitioned sets of measurements forming a grid over a 2D domain. UNIFORMGRID imposes a static grid, while ADAPTIVEGRID uses an initial round of measurements to adjust the coarseness of the grid, avoiding estimation of small sparse regions.

3.4.2 Re-implementation strategies

The process of re-implementing in ϵKTELO this seemingly diverse set of algorithms consisted of breaking the algorithms down into key subroutines and translating them into operators. To summarize, the translation strategy typically falls into one of three categories.

The first translation strategy was to identify specific implementations of common differentially private operations and replace them with a single unified general-purpose operator in ϵKTELO. For instance, the Laplace mechanism (LM), which adds noise drawn from the Laplace distribution, appears in *every one* of the 13 algorithms. Noise addition can be implemented in a number of ways (e.g. calling a function in the numpy.random package, taking the difference of exponential random variables, etc.). In ϵKTELO, all these plans call the same VECTOR LAPLACE operator with a single unified sensitivity calculation.

Another less obvious example of this translation is for subroutines that infer an estimate of $\mathbf{x}$ using noisy query answers. With the exception of IDENTITY and MWEM, each of the algorithms uses instances of least squares inference, often customized to the structure of the noisy query answers. For instance, PRIVELET uses Haar wavelet reconstruction, hierarchical strategies like HB and DAWA use a tree-based implementation of inference, and others like UNIFORM and AHP use uniform expansion. We replaced each of these custom inference methods with a single general-purpose least squares inference operator (LS operators in Fig. 3.2). It would still be possible to implement a specialized inference operator in ϵKTELO that exploited particular properties of a query set, but, given the efficient inference methods described in Sec. 3.5.6, we did not find this to be beneficial.

Our second translation strategy was to identify higher-level patterns that reflect design idioms that exist across multiple algorithms. In these cases, we replace one or more subroutines in the original code with a sequence of operators that capture this idiom. As shown earlier, PLAN #2, 3, 4, 5, 6, 10, and 11, 13 all consist of the operator sequence: Query selection, Query (LM), and Inference (LS), differing only in Query selection method. For other algorithms, this idiom reappears as a subroutine, as in PLAN #8 (AHP) and PLAN #9 (DAWA).

Finally, we were left with distinct subroutines of algorithms that represented key intellectual advances in the differential privacy literature. We encapsulate such subroutines as new operators (e.g. PA in PLAN #8 (AHP) and PD in PLAN #9 (DAWA)) in the framework.

3.4.3 Benefits

We highlight the benefits of re-implementing known algorithms in ϵKTELO.

Code reuse Once reformulated in ϵKTELO, nearly all algorithms use the VECTOR LAPLACE operator and least squares inference. This means that any improvements to either of these operators will be inherited by all the plans. We show such an example in Sec. 3.5.6.

Reduced privacy verification effort Code reuse also reduces the number of critical operators that must be carefully vetted. The operators that require careful vetting are ones that consume the privacy budget, which are the Private→Public operators in Fig. 3.1. These are: VECTOR LAPLACE, the partition selection operators for both DAWA [49] and AHP [111], a query selection operator used by PrivBayes [109], and a query selection operator used by the MWEM [36] algorithm that privately derives the worst-currently-approximated workload query. In contrast, for the DPBench code base, the entire code has to be vetted to audit the use and management of the privacy budget. The end result is that verifying the privacy of an algorithm requires checking fewer lines of code. For example, to verify the QuadTree algorithm in the DPBench codebase requires checking 163 lines of code. However, with ϵKTELO, this only requires vetting the 30-line VECTOR LAPLACE opera-

tor. (Furthermore, by vetting just this one operator, we have effectively vetted 10 of the 18 algorithms in Fig. 3.2, since the only privacy sensitive operator these algorithms use is VECTOR LAPLACE.). When we consider all of the DPBench algorithms in Fig. 3.2, algorithms 1-12, verifying the DPBench implementation requires checking a total of 1837 lines of code while vetting all the privacy-critical operators in ϵKTELO requires checking 517 lines of code.

Transparency As noted above, ϵKTELO plans make explicit the typical patterns that result in accurate differentially private algorithms. Moreover, ϵKTELO plans help clarify the distinctive ingredients of state-of-the-art algorithms. For instance, DAWA and AHP (PLAN #9 and PLAN #8 respectively in Fig. 3.2) have the same structure but differ only in two operators: partition selection and query selection.

3.5 Implementation: efficient matrix support

Matrices and operations on matrices are central to the implementation of ϵKTELO operators but can become a performance bottleneck. In this section we describe a set of specialized matrix representation techniques, based on the implicit definition of matrices, which allows for greater scalability as the size of the data vector grows.

We review next the types of matrix objects in ϵKTELO and then, in Sec. 3.5.2, the different ways matrices can be represented including implicit matrices. In Sec. 3.5.3, we decompose the common matrix operations in ϵKTELO into a small set of primitive operations which every implicit matrix should support. We then describe in Sec. 3.5.4 a general matrix type from which implicit matrices can be built, and use that matrix type, in Sec. 3.5.5, to implement common query selection operators in ϵKTELO. We conclude with Sec. 3.5.6 describing the implementation of inference using implicit matrices.

3.5.1 Matrix types and their operations

Recall that matrices are used to represent three different objects in ϵKTELO: sets of workload queries, sets of measurement queries, and partitions of the domain. In all cases, the matrices contain one column for each element of a corresponding data vector. In the case of both workload and measurement matrices, rows represent linear queries. A partition matrix describes a linear transformation that can be applied to a data vector or to a workload; one row describes a set of elements of the domain that will be combined after the transformation.

The key computations on each matrix type are shown in Table 3.1 (in the left two columns). Workload matrices and measurement matrices both represent sets of queries and so they share similar computations (such as query evaluation on a data vector and calculation of sensitivity) however we only do inference on measurement matrices. Partition matrices are used to reduce and expand both workload matrices and data vectors.

In common plans, the number of rows in a workload or measurement measurement matrix can be as large or larger than n, the number of elements in the data vector. Partitions have at most n rows, but may still be large. For plans operating on large data vectors, where n approaches the size of memory, these matrices, in standard form, are infeasible to represent in memory and operate on. To address this, ϵKTELO provides flexible and efficient matrix capabilities that can be used for the efficient implementation of operators.

3.5.2 Matrix representations: dense, sparse and implicit

The matrix class in ϵKTELO supports matrices using a combination of the dense, sparse, and implicit matrices. These representations differ in their space utilization, their generality, and the efficiency of the matrix operations they support.

A dense $m \times n$ matrix is the standard representation that stores mn values. Obviously, any matrix can be represented in this manner and all operations in Table 3.1 are supported. A sparse matrix stores only non-zero elements of a matrix. Any matrix can be represented

in sparse form, but its efficiency depends on the number of nonzero entries. Where $nnz(\mathbf{A})$ denotes the number of nonzero elements in matrix $\mathbf{A}$, if $nnz(\mathbf{A}) \approx mn$ then sparse matrices do not offer any benefit, and may even be more expensive to represent than the dense representation. However, if $nnz(\mathbf{A}) << mn$ there may be significant improvements to performance in using this representation.

An implicit matrix is a virtual representation of a matrix that may not explicitly store all (or even any) of the elements of the matrix. Because it is a virtual object, it must define appropriate methods so that computations with the implicit matrix produce correct results. While not all matrices allow for efficient implicit representations, we have found that many of the matrices used in ϵKTELO operators have a structure that can be exploited for efficient implicit representation. Note that implicit matrix representations are lossless: they do not approximate some dense matrix but represent it exactly. Therefore an implicit matrix can always be materialized in sparse or dense form, although the goal is to perform computations without materialization.

As an example of an implicit matrix, recall the Prefix workload, an encoding of an empirical CDF, which was used in the example plan (Algorithm 1) of Sec. 3.1:

Example 1 (The Prefix workload: dense, sparse, and implicit) *In dense form, the prefix workload is defined as a lower-triangular matrix containing* 1*'s. If* $n = 5$ *we have:*

$$\mathbf{W_{pre}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

A sparse representation of $\mathbf{W_{pre}}$ *would store (a list of) only the nonzero elements of this matrix, but the space complexity of both dense and sparse representations remains* $O(n^2)$*. In addition, the time complexity of computing matrix-vector products using the dense or*

Table 3.1: Types of matrix objects in ϵKTELO (workload, measurement, partition) and the key computations performed in plans, along with the primitive methods required to support each computation.

Key computations by matrix type		Primitive methods
Workload matrix, W		
Query evaluation	$\mathbf{W}\hat{\mathbf{x}}$	Matrix-vector product
L_1 Sensitivity	$\|\mathbf{W}\|_1$	Abs, Transpose, Matrix-vector product
L_2 Sensitivity	$\|\mathbf{W}\|_2$	Sqr, Transpose, Matrix-vector product
Gram Matrix	$\mathbf{W}^T\mathbf{W}$	Transpose, Matrix multiplication
Row indexing	$\mathbf{w}_i$	Transpose, Matrix-vector product
Measurement matrix, M		
Query evaluation	$\mathbf{Mx}$	Matrix-vector product
L_1 Sensitivity	$\|\mathbf{M}\|_1$	Abs, Transpose, Matrix-vector product
L_2 Sensitivity	$\|\mathbf{M}\|_2$	Sqr, Transpose, Matrix-vector product
Inference (LS)	$\arg\min_{\mathbf{x}} \|\mathbf{Mx} - \mathbf{y}\|_2$	Transpose, Matrix-vector product
Inference (NNLS)	$\arg\min_{\mathbf{x}\geq 0} \|\mathbf{Mx} - \mathbf{y}\|_2$	Transpose, Matrix-vector product
Inference (MW)	$\hat{\mathbf{x}}^{(k+1)} \propto \hat{\mathbf{x}}^{(k)} \odot \exp(\mathbf{g}/N)$ $\mathbf{g} = 0.5\mathbf{M}^T(\mathbf{M}\hat{\mathbf{x}}^{(k)} - \mathbf{y})$	Transpose, Matrix-vector product
Partition matrix, P		
Reduce workload	$\mathbf{W}' = \mathbf{WP}^+$	Transpose, Matrix multiplication
Reduce data vector	$\mathbf{x}' = \mathbf{Px}$	Matrix-vector product
Expand workload	$\mathbf{W} = \mathbf{W}'\mathbf{P}$	Matrix multiplication
Expand data vector	$\mathbf{x} = \mathbf{P}^+\mathbf{x}'$	Transpose, Matrix-vector product

sparse representation is $O(n^2)$. However, the Prefix matrix can be completely specified by a single parameter, n, which is the only state stored for the implicit version of $\mathbf{W}_{\mathbf{pre}}$. Further, we can evaluate the matrix-vector product $\mathbf{y} = \mathbf{W}_{\mathbf{pre}}\mathbf{x}$ using a simple one-pass algorithm over $\mathbf{x}$: first compute $y_1 = x_1$. Then, for $k = 2 \ldots n$, compute $y_k = y_{k-1} + x_k$. Therefore, by representing the Prefix workload implicitly we can achieve $O(1)$ space complexity and $O(n)$ time complexity for computing matrix-vector products.

3.5.3 Computing with implicit matrices

Most implicit matrices require very little internal state to be stored. The main challenge is therefore to insure that all necessary computations involving an implicit matrix can be carried out efficiently, hopefully without falling back to materialization of the dense form

of the matrix. Before defining additional implicit matrix constructions, we review the key computations ϵKTELO matrix objects must support.

A careful examination of the operators currently implemented in ϵKTELO resulted in the list of key computations in Table 3.1, where the left two columns describe operations on matrices that commonly occur in plans. Importantly, these plan-level matrix computations can be decomposed into just five fundamental matrix methods, which we call *primitive methods*: matrix-vector product, transpose, matrix multiplication, element-wise absolute value (abs), and element-wise square (sqr). Matrix-vector product takes as input a vector and returns a vector. The multiplication of two implicit matrices returns a new implicit matrix, as does transpose, which returns another implicit matrix that represents the transpose linear transformation. Element-wise sqr and abs both return new implicit matrices.

Below we review the key computations on ϵKTELO matrix objects in Table 3.1, how they can be decomposed into primitive methods, and implementation considerations. Our goal will then be to construct implicit matrices that efficiently support the primitive methods.

Query evaluation and data reduction In ϵKTELO plans, matrix-vector multiplication is used by workload and measurement matrices for query evaluation, and by partition matrices for reduction of the data vector.

Sensitivity The sensitivity of measurement and workload matrices can be computed using a combination of primitive methods (abs, sqr, transpose, and matrix-vector product). For a matrix $\mathbf{M}$, we compute the maximum column sum of $abs(\mathbf{M})$ (for L_1 sensitivity) or $sqr(\mathbf{M})$ (for L_2 sensitivity) which can be done by doing a transpose-matrix-vector product with $\mathbf{1}$ – the vector of all 1's:

$$\|\mathbf{M}\|_1 = max(abs(\mathbf{M})^T\mathbf{1}) \qquad \|\mathbf{M}\|_2 = \sqrt{max(sqr(\mathbf{M})^T\mathbf{1})}$$

Inference The most common form of inference in ϵKTELO plans is based on least squares. While classical solutions to the least squares problem involve matrix decompositions and computation of the pseudo-inverse, we will show that iterative methods lead to much greater scalability in combination with our implicit matrix representations. Least squares, non-negative least squares, and multiplicative-weights inference can all be implemented using just the matrix-vector product and transpose primitive methods. We will discuss iterative inference in Sec. 3.5.6.

Gram Matrix Some workload-adaptive mechanisms like GreedyH and HDMM require the (materialized) gram matrix of the workload. For a workload $\mathbf{W}$, the gram matrix is $\mathbf{W}^T\mathbf{W}$. This computation can be implemented in terms of transpose, matrix multiplication, and materialize. For extremely large workloads with special structure, where $\mathbf{W}^T\mathbf{W}$ is much smaller than $\mathbf{W}$ (like the set of all range queries), a more efficient version can be implemented directly that avoids using the primitive methods.

Partition reduction and expansion Partition matrices need to be able to reduce the data vector and workload. They also have to be able to do the reverse expansion operations. As we show in the proof of Prop. 3.6.1, because of the special structure of partition matrices, the pseudo-inverse of any matrix $\mathbf{P}$ can be computed as the product of $\mathbf{P}^T$ and a diagonal matrix $\mathbf{D}$. Thus, partition matrices simply need the three primitive methods: matrix-vector product, transpose, and matrix-multiplication.

Row Indexing The MWEM algorithm and its variants (described in Sec. 3.7.1) use the worst-approximated query selection operator, which requires row indexing, or materialization of the i^{th} row of a matrix. This can be implemented in terms of the primitive methods transpose and matrix-vector product as follows: $\mathbf{w}_i = \mathbf{W}^T\mathbf{e}_i$, where $\mathbf{e}_i$ is the i^{th} column of an identity matrix.

Materialize If a plan requires working with matrices in a manner not supported by the interface of our implicit matrices, the matrix can always be materialized, at which point

standard implementations of matrix methods can be employed. As in row indexing, materialization can be performed by a sequence of matrix vector products with the columns of an identity matrix, i.e., $\mathbf{A}\mathbf{e}_i$ for $i = 1, \ldots n$.

3.5.4 Generalized matrix construction

ϵKTELO contains a matrix class, denoted $EMatrix$, that generalizes dense, sparse, and implicit matrices, supporting flexible matrix construction using a small set of specially designed *core matrices* which may then be combined with combining operations (union, product, and Kronecker product). This provides a flexible and extensible mechanism for constructing a wide range of matrices. The following grammar describes the construction of $EMatrix$ instances:

$$
\begin{aligned}
CoreMatrix &= Identity \mid Ones \mid Prefix \mid Suffix \mid Wavelet \\
EMatrix &= DenseMatrix \mid SparseMatrix \mid CoreMatrix \\
EMatrix &= Union(EMatrix, EMatrix) \\
EMatrix &= Product(EMatrix, EMatrix) \\
EMatrix &= Kronecker(EMatrix, EMatrix)
\end{aligned}
$$

Unless an $EMatrix$ is defined as a single $SparseMatrix$ or $DenseMatrix$, we consider it implicit, since it is not fully materialized.

Core matrices

The $CoreMatrix$ type forms the basic building block for $EMatrix$ and each is defined implicitly. The following are custom core matrices we designed to support ϵKTELO operators:

- **Identity**: Identity is the simplest building block. It is defined as the matrix $\mathbf{I}$ having the property that $\mathbf{Iv} = \mathbf{v}$ for all vectors $\mathbf{v}$. Thus, the implementation of matrix-vector product is trivial. Similarly, transpose, abs, and sqr are simple no-ops.

- **Ones**: Ones is the $m \times n$ matrix of all ones. Matrix-vector products can be efficiently computed by summing up the entries of the input vector, and constructing a m-length vector with that value. The transpose is a $n \times m$ Ones matrix, and abs and sqr are simple no-ops. **Total** is a special case of the Ones matrix where $m = 1$.

- **Prefix and Suffix**: The description of prefix and the algorithm for efficiently computing matrix-vector products is given in Example 1. The transpose of Prefix is Suffix, and abs and sqr are simple no-ops.

- **Wavelet**: Wavelet is the Haar wavelet transform. Efficient algorithms exist for evaluating matrix-vector products implicitly with the Haar wavelet [101]. The transpose has a similar form, but abs and sqr, if needed, must materialize the matrix. However, for this matrix sensitivity may be computed directly, without going through abs and sqr.

The primitive methods described above have very simple and efficient implementations for the core matrices. In Table 3.2 we report the space utilization for each core matrix, along with the time complexity for one of the most important primitive methods (matrix-vector product). We compare the complexity of the core implicit matrices with their standard dense and sparse representations, showing significant reductions in space usage – up to a factor n^2. We observe a similar reduction in time complexity of matrix-vector products by up to a factor of n.

Table 3.2: Comparison of core implicit matrices to their corresponding sparse and dense representations, in terms of space usage and time complexity of a matrix-vector product. For sparse and dense matrices, the time complexity is the same as the space complexity.

	Implicit		Dense	Sparse
Core Matrix	Space Usage	Time (mat-vec)	Space/Time	Space/Time
$Identity$	$O(1)$	$O(n)$	$O(n^2)$	$O(n)$
$Ones$	$O(1)$	$O(m+n)$	$O(mn)$	$O(mn)$
$Prefix$	$O(1)$	$O(n)$	$O(n^2)$	$O(n^2)$
$Suffix$	$O(1)$	$O(n)$	$O(n^2)$	$O(n^2)$
$Wavelet$	$O(1)$	$O(n \log n)$	$O(n^2)$	$O(n \log n)$

Composing matrices

Core matrices and arbitrary sparse or dense matrices can be combined using a Union, Product (including with a constant), and Kronecker Product to form new matrices that are implicit (or partially implicit).

If matrix $\mathbf{M}_1$ and $\mathbf{M}_2$ each represent queries, then Union$(\mathbf{M}_1, \mathbf{M}_2)$ is a matrix that represents the union of the queries of $\mathbf{M}_1$ and $\mathbf{M}_2$. It is useful for building complex workloads and measurement matrices, and it also important in plans to bring together all measured queries for global inference. Product is less frequently needed, but is used for multiplying partition matrices with workload and measurement matrices.

Kronecker product is especially useful for constructing workload and measurement matrices over multi-dimensional domains. Suppose our input is a relation $R(A, B)$, we vectorize its projection of $\pi_A(R)$ to get data vector $\mathbf{x}_A$, and we define a set of queries of interest as matrix $\mathbf{M}_A$. If we similarly form a matrix of queries $\mathbf{M}_B$ over the vectorization of $\pi_B(R)$ then Kronecker$(\mathbf{M}_A, \mathbf{M}_B)$ (denoted $\mathbf{M}_A \otimes \mathbf{M}_B$ in matrix equations) is a matrix that encodes a new set of queries over both attributes A and B and it contains $q_{a_i} \wedge q_{b_j}$ for each q_{a_i} in $\mathbf{M}_A$ and each q_{b_j} in $\mathbf{M}_B$, i.e. it contains the conjunctive combination of all pairs of queries drawn from $\mathbf{M}_A$ and $\mathbf{M}_B$.

The formal definition of the Kronecker product is:

Definition 8 *The Kronecker product* $\mathbf{A} \otimes \mathbf{B}$ *between a* $m_A \times n_A$ *matrix* $\mathbf{A}$ *and a* $m_B \times n_B$ *matrix* $\mathbf{B}$ *is a* $m_A m_B \times n_A n_B$ *matrix defined as:*

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1n_A}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m_A 1}\mathbf{B} & \dots & a_{m_A n_A}\mathbf{B} \end{bmatrix}$$

Kronecker products were first used within the context of a privacy mechanism by McKenna et al. [61].

We present these as binary operations on implicit matrices, but because they are associative, they can also be applied to a collection of k sub-matrices. For example for $k = 3$ we may write $Union(\mathbf{A}, \mathbf{B}, \mathbf{C})$ as shorthand notation for $Union(\mathbf{A}, Union(\mathbf{B}, \mathbf{C}))$.

Example 2 *Suppose our input relation is* $R(age, income, marital\text{-}status)$*, where age and income are discretized into a 100 bins, and marital-status is a categorical attribute with 7 possible values, resulting in a data vector of size 70000. We want to accurately answer range queries on age and income, broken down by various marital statuses. Thus, we may construct the following workload using tools from above:*

$$\begin{aligned} \mathbf{W} = Kronecker(&Prefix, \\ &Prefix, \\ &Union(Total, Identity, Dense)) \end{aligned}$$

where $Dense$ *is a* 2×7 *query matrix with two queries that aggregate over the marital status attribute into two groups: "married" and "unmarried". Using Table 3.3, we see that the only storage required to represent* $\mathbf{W}$ *is the* 2×7 $Dense$ *matrix, and metadata for the* $Prefix$*,* $Total$*, and* $Identity$ *matrices. In contrast, the sparse and dense representation of* $\mathbf{W}$ *would require about 8 GB and 56 GB respectively.*

Table 3.3: Space and time complexity of composed matrices, in terms of the complexity of sub-matrices.

Matrix	Operation	Space Usage	Time (mat-vec)
$Union(\mathbf{A}, \mathbf{B})$	$\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix}$	Space($\mathbf{A}$) + Space($\mathbf{B}$)	Time($\mathbf{A}$) + Time($\mathbf{B}$)
$Product(\mathbf{A}, \mathbf{B})$	$\mathbf{AB}$	Space($\mathbf{A}$) + Space($\mathbf{B}$)	Time($\mathbf{A}$) + Time($\mathbf{B}$)
$Kronecker(\mathbf{A}, \mathbf{B})$	$\mathbf{A} \otimes \mathbf{B}$	Space($\mathbf{A}$) + Space($\mathbf{B}$)	n_B Time($\mathbf{A}$) + m_A Time($\mathbf{B}$)

Supporting the primitive methods

Core, sparse, and dense matrices have native support for the primitive methods discussed previously. When a primitive method is invoked for an $EMatrix$ that results from one or more of the combining operations, the work is delegated to the constituent sub-matrices, and thus the matrices formed by composition inherit the performance characteristics of the sub-matrices. In particular, the key primitive methods can be implemented efficiently on matrices formed from unions, products, and Kronecker products.

The key performance characteristics of composed matrices are summarized in Table 3.3.

3.5.5 Matrix constructions for ϵKTELO operators

Using the generalized matrix construction described above, we can re-implement many of the existing query selection and partition operators currently in ϵKTELO. We will show in Sec. 3.8 that the use of implicit matrices leads to significant improvements in efficiency and scalability, as well as the reduction of space consumption.

Query selection operators based on range queries

A notable class of query matrices that we can efficiently represent using the tools from above is an arbitrary collection of range queries. This type of workload has been extensively studied in the literature and many of the query selection operators in ϵKTELO are specially-designed sets of range queries including H2, Hb, QuadTree, UniformGrid, and AdaptiveGrid.

Recall that a single range query over a 1-dimensional domain can be specified by a pair of indices (i, j), and a workload of range queries can be represented as a list of these pairs. This suggests we can store any range query workload using only $O(m)$ space where m is the number of queries. Furthermore, one way to evaluate matrix-vector products is by iterating through each query one-by-one and evaluating $\sum_{k=i}^{j} x_k$. The time complexity of this approach is $O(mn)$ in the worst case, which is equivalent to the sparse and dense representations.

Our general matrix construction allows us to do even better by exploiting the fact that any range query can be expressed as the difference of two prefix queries. Thus the matrix can be represented as $Product(Sparse, Prefix)$ where $Sparse$ is a $m \times n$ sparse matrix with two non-zero entries per row. An illustrating example is shown below:

Example 3 (Range Queries) *A collection of four range queries over a domain of size five, represented implicitly as the product of a $Sparse$ matrix and the $Prefix$ matrix (displayed here in dense form for illustration purposes):*

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Using this construction, we can evaluate matrix-vector products in $O(n + m)$ time, which is a substantial improvement over the other representations. The range query construction can be naturally extended to multi-dimensional domains by replacing $Prefix$ with $Kronecker(Prefix, \ldots, Prefix)$ and replacing $Sparse$ with a sparse matrix with up to 2^d nonzero entries per row, where d is the number of dimensions of the domain.

A special case of this range query construction is hierarchical and grid-based matrices used by H2, Hb, and QuadTree. These matrices always have an Identity matrix, and while

they can be represented using the above construction, it is more efficient to represent them in a slightly different way as $Union(Identity, Product(Sparse, Prefix))$, which is the representation used in our empirical evaluation.

Note that even though $Product$ does not natively support abs and sqr, for the case of range queries, or more generally any matrix with binary values, abs and sqr are simple no-ops.

Representing marginals

A common task for multi-dimensional data analysis is computing the marginals of a dataset. Marginals may be used both as part of workloads and measurement matrices. They can be efficiently represented using the tools from above, as demonstrated in Example 4.

Example 4 (Marginals) *Any marginal can be represented as a Kronecker product of $Identity$ and $Total$ building blocks. For example, the two-way marginal that sums out the second attribute can be encoded as:*

$$\mathbf{W}_{13} = Kronecker(Identity, Total, Identity)$$

Further, an arbitrary collection of marginals can be encoded as $Union$ of these Kronecker products. All 2-way marginals is:

$$\begin{aligned}\mathbf{W}_{2way} = Union(&Kronecker(Identity, Identity, Total),\\ &Kronecker(Identity, Total, Identity),\\ &Kronecker(Total, Identity, Identity))\end{aligned}$$

Partition operators

The matrices used by partition operators are represented simply as $Sparse$ matrices. While an implicit definition is possible, it would not offer any improvement in space or time over the sparse representation.

3.5.6 Implementing inference

Inference is a fundamental operator that can improve error with no cost to privacy and, accordingly, we saw that it appeared in virtually every algorithm re-implemented in ϵKTELO (as shown in Fig. 3.2). But inference can be a costly operation. Recall that the input to inference is a measurement matrix, denoted by $\mathbf{M}$, containing m queries defined over a data vector of size n, and the list of noisy answers $\mathbf{y}$. The least squares solution (Eq. (3.2)) is given by the solution to the normal equations $\mathbf{M}^T\mathbf{M}\hat{\mathbf{x}} = \mathbf{M}^T\mathbf{y}$. Assuming $\mathbf{M}^T\mathbf{M}$ is invertible, then the solution is unique and can be expressed as $\hat{\mathbf{x}} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{y}$. Often explicit matrix inversion is avoided, in favor of suitable factorizations of $\mathbf{M}$ (e.g., QR or SVD). However, the time complexity of such "direct" methods is still generally cubic in the domain size when $m = O(n)$. In practice we have found that the runtime of such direct methods is unacceptable when n is greater than about 5000.

Algorithms in prior work [101, 40, 79, 78] have used least squares inference on large domains by restricting the selection of queries, namely to those representing a set of hierarchical queries. This allows for inference in time linear in the domain size, avoiding the explicit matrix representation of the queries. We avoid this approach in ϵKTELO because it means that a custom inference method may be required for each query selection operation, and because it limits the measurement sets that can be used. In addition, hierarchical methods only work for least squares but not least squares with non-negativity constraints.

An alternative approach to least squares inference is to use an iterative gradient-based method, which solves the normal equations by repeatedly computing matrix-vector products $\mathbf{M}\mathbf{v}$ and $\mathbf{M}^T\mathbf{v}$ until convergence. The time complexity of these methods is $O(kn^2)$ for dense matrix representations where k is the number of iterations. In experiments we use a well-known iterative method, LSMR [30]. Empirically, we observe LMSR to converge in far fewer than n iterations when $\mathbf{M}$ is well-conditioned, which is the case as long as the queries are not taken with vastly different noise scales, and thus we expect $k << n$.

Iterative approaches, using implicit matrices, are well-suited to the other inference methods in ϵKTELO: least squares with non-negativity constraints (Eq. (3.3)) and multiplicative weights. For the former, we use the limited memory BFGS algorithm with bound constraints [13]. The time complexity of this algorithm is the same as LSMR, although the number of iterations needed for convergence may be different, and there is a constant factor overhead for storing the low-rank approximation to the inverse Hessian matrix. The multiplicative weights inference algorithm is defined in an iterative manner and requires the same primitive methods as ordinary least squares and non-negative least squares.

The combination of our general matrix construction techniques with iterative inference result in flexible inference capabilities for plan authors. With relative freedom, they can construct measurement matrices, or combine measurements from parts of a plan, and apply a single generic inference operator, which will run efficiently. In Sec. 3.8 we show that using iterative least squares on implicitly represented matrices, we can scale inference to domains consisting of hundreds of millions of cells while staying within modest runtime bounds, well beyond what is possible with sparse or dense representations.

3.6 Workload-based partition selection

In many cases, the goal of a differentially private algorithm (and its corresponding ϵKTELO plan) is to answer a given workload of queries, $\mathbf{W}$, defined in terms of a data vector $\mathbf{x}$. We describe next a method for reducing the representation of the $\mathbf{x}$ vector to precisely the elements required to correctly answer the workload queries. This is a new partition selection operator, called *workload-based partition selection*, which can be used as input to a V-REDUCEBYPARTITION transformation.

We define the partition below and prove that, under reasonable assumptions, using such domain reduction can never hurt accuracy. We provide an algorithm for computing the partition, which can be executed using implicit workload representations. Later, in Sec. 3.8.3,

we will show empirically that using this partition in plans can offer significant improvement in both runtime and error.

3.6.1 The workload-based partition and its properties

For a workload $\mathbf{W}$ of linear queries described on data vector $\mathbf{x}$, it is often possible to define a reduction of $\mathbf{x}$, to a smaller $\mathbf{x}'$, and appropriately transform the workload to $\mathbf{W}'$, so that all workload query answers are preserved, i.e. $\mathbf{W}\mathbf{x} = \mathbf{W}'\mathbf{x}'$. Intuitively, such a reduction is possible when a set of elements of $\mathbf{x}$ is not distinguished by the workload: each linear query in the workload either ignores it, or treats it in precisely the same way. In that case, that portion of the domain need not be represented by multiple cells, but instead by a single cell in a reduced data vector. It is in this sense that the reduction is lossless with respect to the workload. Following this intuition, the domain reduction can be computed from the matrix representation $\mathbf{W}$ of the workload by finding groups of identical columns: elements of these groups will be merged in $\mathbf{W}$ to get $\mathbf{W}'$ while the corresponding cells in $\mathbf{x}$ are summed.

Example 5 *Consider a table with schema Census(age, sex, salary). If the workload consists of queries Q1*$(salary \leq 100K, sex = M)$ *and Q2*$(salary > 100K, sex = F)$ *the workload only requires a data vector consisting of 2 cells. If the workload consists of all 1-way marginals then no workload-based data reduction is possible.*

Note that calculating this partition only requires knowledge of the workload and is therefore done in the unprotected client space (and does not consume the privacy budget). The partition is then input to a V-REDUCEBYPARTITION transformation operator carried out by the protected kernel and its stability is one.

The new workload-based partition selection operator can be formalized in terms of a linear matrix operator, as follows:

Definition 9 (Workload-based partition selection) *Let* $\mathbf{w}_1, \ldots, \mathbf{w}_n$ *denote the columns of* $\mathbf{W}$ *and let* $\mathbf{u}_1, \ldots, \mathbf{u}_p$ *denote those that are unique. For* $h(\mathbf{u}) = \{j \mid \mathbf{w}_j = \mathbf{u}\}$*, define*

the transformation matrix $\mathbf{P} \in \mathbb{R}^{p \times n}$ *to have* $P_{ij} = 1$ *if* $j \in h(\mathbf{u}_i)$ *and* $P_{ij} = 0$ *otherwise. The reverse transformation is the pseudo-inverse* $\mathbf{P}^+ \in \mathbb{R}^{n \times p}$.

The matrix $\mathbf{P}$ defines a partition of the data, which can be passed to V-REDUCEBYPARTITION to transform the data vector, and $\mathbf{P}^+$ can be used to transform the workload accordingly. When $\mathbf{P}$ is passed to V-REDUCEBYPARTITION , the operator produces a new data vector $\mathbf{x}' = \mathbf{P}\mathbf{x}$ where x'_i is the sum of entries in $\mathbf{x}$ that belong to i^{th} group of $\mathbf{P}$. When viewed as an operation on the workload, $\mathbf{P}^+$ merges duplicate columns by taking the row-wise average for each group. This is formalized as follows:

Proposition 3.6.1 (properties: workload-based reduction) *Given transform matrix* $\mathbf{P}$ *and its pseudo-inverse* $\mathbf{P}^+$*, the following hold:*

- $\mathbf{x}' = \mathbf{P}\mathbf{x}$ *is the reduced data vector;*
- $\mathbf{W}' = \mathbf{W}\mathbf{P}^+$ *is the workload matrix, represented over* $\mathbf{x}'$*;*
- *The transformation is lossless:* $\mathbf{W}\mathbf{x} = \mathbf{W}'\mathbf{x}'$

Proof 3.6.2 *First note that* $\mathbf{P}^+ = \mathbf{P}^T\mathbf{D}^{-1}$ *where* $\mathbf{D}$ *is the* $p \times p$ *diagonal matrix with* $D_{ii} = |h(\mathbf{u}_i)|$ *for* h *defined in Def. 9. Since* $\mathbf{P}$ *has linearly independent rows,* $\mathbf{P}^+ = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}$ *and* $\mathbf{P}\mathbf{P}^T = \mathbf{D}$ *because* $h(\mathbf{u}_i)$ *and* $h(\mathbf{u}_j)$ *are disjoint for* $i \neq j$*. By definition of* $\mathbf{P}$*, we see that* $x'_i = \sum_{j \in h(\mathbf{u}_i)} x_j$ *for* $1 \leq i \leq p$*. Similarly, the* i^{th} *column of* $\mathbf{W}'$ *is given by* $\mathbf{w}'_i = \frac{1}{|h(\mathbf{u}_i)|} \sum_{j \in h(\mathbf{u}_i)} \mathbf{w}_j$*. Since* $\mathbf{w}_j = \mathbf{u}_i$ *when* $j \in h(\mathbf{u}_i)$*, we have* $\mathbf{w}'_i = \mathbf{u}_i$*, which shows that* $\mathbf{W}'$ *is just* $\mathbf{W}$ *with the duplicate columns removed. Using these definitions, we show that the transformation is lossless:*

$$\mathbf{W}\mathbf{x} = \sum_{i=1}^{n} \mathbf{w}_i x_i = \sum_{i=1}^{p} \mathbf{u}_i \sum_{j \in h(\mathbf{u}_i)} x_j = \sum_{i=1}^{p} \mathbf{w}'_i x'_i = \mathbf{W}'\mathbf{x}'$$

As noted in Example 5, not all workloads allow for reduction (in some cases, the $\mathbf{P}$ matrix computed above is the identity). But others may allow a significant reduction, which

improves the efficiency of subsequent operators. Less obvious is that workload-based data reduction would impact accuracy. In fact, many query selection methods from existing work depend implicitly on the representation of the data in vector form, and these approaches may be improved by domain reduction. In Sec. 3.6.3 we measure the impact of this transform on accuracy and efficiency.

We show next that this reduction does not hurt accuracy: for any selected set of measurement queries, their reduction will provide lower error after transformation.

Theorem 3.6.3 *Given a workload* $\mathbf{W}$ *and data vector* $\mathbf{x}$*, let* $\mathbf{M}$ *be any query matrix that answers* $\mathbf{W}$*. Then if* $\mathbf{q}' = \mathbf{q}\mathbf{P}^+$ *is a reduced query and* $\mathbf{M}' = \mathbf{M}\mathbf{P}^+$ *is the query matrix on the reduced domain,* $Error_{\mathbf{q}'}(\mathbf{M}') \leq Error_{\mathbf{q}}(\mathbf{M})$ *for all* $\mathbf{q} \in \mathbf{W}$*.*

Proof 3.6.4 *We use the definition of squared error from [53] which shows that* $\forall \mathbf{q} \in \mathbf{W}$*,* $Error_{\mathbf{q}}(\mathbf{M}) \propto \|\mathbf{M}\|_1^2 \|\mathbf{q}\mathbf{M}^+\|_2^2$ *as long as* $\mathbf{M}$ *supports* $\mathbf{W}$*. Let* $\hat{\mathbf{m}}_j$ *and* $\mathbf{m}_j$ *denote the* j^{th} *column of* $\hat{\mathbf{M}}$ *and* $\mathbf{M}$ *respectively. First we show that* $\left\|\hat{\mathbf{M}}\right\|_1 \leq \|\mathbf{M}\|_1$*:*

$$\left\|\hat{\mathbf{M}}\right\|_1 = \max_{1 \leq i \leq p} \|\hat{\mathbf{m}}_j\| \tag{3.4}$$

$$= \max_{1 \leq i \leq p} \left\| \frac{1}{|h(\mathbf{u}_j)|} \sum_{j \in h(\mathbf{u}_i)} \mathbf{m}_j \right\| \tag{3.5}$$

$$\leq \max_{1 \leq i \leq p} \frac{1}{|h(\mathbf{u}_i)|} \sum_{j \in h(\mathbf{u}_i)} \|\mathbf{m}_j\|_1 \tag{3.6}$$

$$\leq \max_{1 \leq i \leq p} \max_{j \in h(\mathbf{u}_i)} \|\mathbf{m}_j\|_1 = \tag{3.7}$$

$$= \max_{1 \leq i \leq n} \|\mathbf{m}_i\|_1 \tag{3.8}$$

$$= \|\mathbf{M}\|_1 \tag{3.9}$$

where $h(\mathbf{u})$ *and* $\mathbf{u}_i$ *are defined in definition 9. Now we show that* $\|\mathbf{q}\mathbf{M}^+\|_2 \leq \left\|\hat{\mathbf{q}}\hat{\mathbf{M}}^+\right\|_2$*. Observe that it is possible to to write* $\mathbf{q}$ *as a linear combination of the rows of* $\mathbf{M}$ *since* $\mathbf{M}$

supports $\mathbf{W}$*. That is, there exists a* $\mathbf{z}$ *satisfying* $\mathbf{zM} = \mathbf{q}$*. In general, there may be infinitely many solutions to this linear system, but* $\mathbf{z} = \mathbf{qM}^+$ *is the* minimum-norm *solution [74]. On the reduced domain, we also know there exists a* $\hat{\mathbf{z}}$ *satisfying* $\hat{\mathbf{z}}\hat{\mathbf{M}} = \hat{\mathbf{q}}$*, or equivalently* $\hat{\mathbf{z}}\mathbf{MP}^+ = \mathbf{qP}^+$*. By making the substitution* $\mathbf{zM} = \mathbf{q}$*, it's easy to see that* $\hat{\mathbf{z}} = \mathbf{z}$ *is one solution to this linear system. The minimum norm solution to this linear system is* $\hat{\mathbf{z}} = \hat{\mathbf{q}}\hat{\mathbf{M}}^+$*, which implies* $\|\hat{\mathbf{z}}\|_2 \leq \|\mathbf{z}\|_2$*. This shows that* $\left\|\hat{\mathbf{q}}\hat{\mathbf{M}}^+\right\|_2 \leq \|\mathbf{qM}^+\|_2$*, and it immediately follows that* $Error_{\hat{\mathbf{q}}}(\hat{\mathbf{M}}) \leq Error_{\mathbf{q}}(\mathbf{M})$ *as desired.*

3.6.2 Computing the partition

The computation of the partition $\mathbf{P}$ in Def. 9 is conceptually straightforward: it simply requires finding the unique columns of $\mathbf{W}$ and grouping them. Finding the unique columns of $\mathbf{W}$ exactly by inspecting the entries of $\mathbf{W}$ requires an explicit matrix in dense form, or materializing an implicit matrix. Algorithm 4 provides an efficient method for finding the column groupings that does not require a explicit matrix representation, relying instead only on the primitive methods of transpose and matrix-vector product. This approach is highly scalable and is compatible with the implicit matrix representations of the workload discussed in section 3.5.

Algorithm 4 An algorithm for workload-based data reduction

procedure COMPUTE REDUCTION MATRIX(W)
    Input: $m \times n$ matrix $\mathbf{W}$
    Output: $p \times n$ matrix $\mathbf{P}$ where $p \leq n$
    set $\mathbf{v}$ = vector of m samples from Uniform$(0, 1)$ ▷ $1 \times m$
    compute $\mathbf{h} = \mathbf{W}^T\mathbf{v}$ ▷ $1 \times n$
    let $G = g_1, \ldots, g_p$ be groups of common values in $\mathbf{h}$
    initialize matrix $\mathbf{P}$ with zeros ▷ $p \times n$
    for g_i in G **do**
        set row i of $\mathbf{P}$ to 1 in each position of g_i
    end for
    return $\mathbf{P}$
end procedure

By grouping the elements of $\mathbf{h}$ (line 6) we recover the column groupings of $\mathbf{W}$, because if $\mathbf{w}_i = \mathbf{w}_j$ then $h_i = h_j$ and if $\mathbf{w}_i \neq \mathbf{w}_j$ then $P(h_i = h_j) = 0$ since h_i and h_j are continuous random variables. While algorithm Algorithm 4 is a randomized algorithm, it returns the correct result almost surely. The probability of incorrectly grouping two different columns is approximately 10^{-16} with a 64-bit floating point representation, but if needed we can repeat the procedure k times until the probability of failure ($\sim 10^{-16k}$) is vanishingly small.

3.7 Case studies: ϵKTELO in action

In this section we put ϵKTELO into action by developing new algorithms. First, we show that it is easy to re-design and improve existing algorithms by combining operators in new ways. In particular, we develop a variant of the MWEM algorithm with significantly improved accuracy. Then we use ϵKTELO to tackle two practical use-cases, constructing new plans which offer state-of-the-art accuracy. We evaluate all of the proposed plan in Sec. 3.8.

3.7.1 Recombination of operators to improve MWEM

Using ϵKTELO, we design new variants of the well-known *Multiplicative Weights Exponential Mechanism* (MWEM) [36] algorithm. MWEM repeatedly derives the worst-approximated workload query with respect to its current estimate of the data, then measures the selected query, and uses the multiplicative weights update rule to refine its estimate, often along with any past measurements taken. This repeats a number of times, determined by an input parameter.

When viewed as a plan in ϵKTELO, a deficiency of MWEM becomes apparent. Its query selection operator selects a *single* query to measure whereas most query selection operators select a set of queries such that the queries in the set measure disjoint partitions of the data. By the parallel composition property of differential privacy, measuring the

entire set has the same privacy cost as asking any single query from the set. This means that MWEM could be measuring more than a single query per round (with no additional consumption of the privacy budget). To exploit this opportunity, we designed an augmented query selection operator that adds to the worst-approximated query by attempting to build a binary hierarchical set of queries over the rounds of the algorithm. In round one, it adds any unit length queries that do not intersect with the selected query. In round two, it adds length two queries, and so on.

Adding more measurements to MWEM has an undesirable side effect on runtime, however. Because it measures a much larger number of queries across rounds of the algorithm and the runtime of multiplicative weights inference scales with the number of measured queries, inference can be considerably slower. Thus, we also use replace it with a version of least-squares with a non-negativity constraint (NNLS) and incorporate a high-confidence estimate of the total which is assumed by MWEM.

In total, we consider three MWEM variants: an alternative query selection operator (PLAN #18), an alternative inference operator (PLAN #19), and the addition of both alternative operators (PLAN #20). These are shown in Fig. 3.2 and evaluated in Sec. 3.8.

3.7.2 Census case-study

The U.S. Census Bureau collects data about U.S. citizens and releases a wide variety of tabulations describing the demographic properties of individuals. We consider a subset of the (publicly released) March 2000 Current Population Survey. The data report on 49,436 heads-of-household describing their income, age (in years), race, marital status, and gender. We divide Income into 5000 uniform ranges from $(0, 750000)$, age in 5 uniform ranges from $(0, 100)$, and there are 7, 4 and 2 possible values for status, race and gender.

We author differentially private plans for answering a workload of queries similar to Census tabulations. This is challenging because the data domain is large and involves multiple dimensions. The workloads we consider are: (a) the Identity workload (or counts

on the full domain of 1.4M cells), (b) a workload of all 2-way marginals (age $\times$ gender, race $\times$ status, and so on), and (c) a workload suggested by U.S. Census Bureau staff: Prefix(Income) which consists of all counting queries of the form (income $\in (0, i_{high})$, age=a, marital=m, race=r, gender=g) where $(0, i_{high})$ is an income range, and a, m, r, g may be values from their resp. domains, or $<any>$.

There are few existing algorithms suitable for this task. We were unable to run the DAWA [49] algorithm directly on such a large domain. In addition, it was designed for 1d- and 2d- inputs. One of the few algorithms designed to scale to high dimensions is PrivBayes [109]. While not a workload-adaptive algorithm, PrivBayes generates synthetic data which can support the census workloads above. We use PrivBayes as a baseline and we use ϵKTELO to construct three new plans composed of operators in our library. The proposed plans are: HB-STRIPED (PLAN #15), DAWA-STRIPED (PLAN #14), and PRIVBAYESLS (PLAN #17). The first two "striped" plans showcase the ability to adapt lower dimensional techniques to a higher dimensional problem avoiding scalability issues. The third plan considers improving on PrivBayes by changing its inference step.

Both HB-STRIPED and DAWA-STRIPED use the same plan structure: first they partition the full domain, then they execute subplans to select measurements for each partition, and lastly, given the measurement answers, they perform inference on the full domain and answer the workload queries. The partitioning of the initial step is done as follows: given a high dimensional dataset with N attributes and an attribute A of that domain, our partitions are parallel "stripes" of that domain for each fixed value of the rest of the $N-1$ attributes, so that the measurements are essentially the one-dimensional histograms resulting from each stripe. In the case of HB-STRIPED (fully described in Algorithm 5), the subplan executed on each partition is the HB algorithm [79], which builds an optimized hierarchical set of queries, while in the case of the DAWA-STRIPED the subplan is DAWA algorithm [49]. Note that while the data-independent nature of the HB subplan means that all the measurements from each stripe are the same, that is not the case with DAWA, which poten-

tially selects different measurement queries for each stripe, depending on the local vector it sees. For our experiments, the attribute chosen was Income, and for DAWA-STRIPED we set the DAWA parameter ρ to 0.25.

Algorithm 5 HB-STRIPED

$D \leftarrow$ PROTECTED(source_uri) ▷ Init
$\mathbf{x} \leftarrow$ T-VECTORIZE(D) ▷ Transform
$\mathbf{R} \leftarrow$ StripePartition(Attr) ▷ Partition Selection
$\mathbf{x}_R \leftarrow$ V-SPLITBYPARTITION ($\mathbf{x}$, $\mathbf{R}$)
$\mathbf{M} \leftarrow \emptyset$
$\mathbf{y} \leftarrow \emptyset$
for $\mathbf{x}' \in \mathbf{x}_R$ **do**
    $\mathbf{M} \leftarrow \mathbf{M} \cup$ HB($\mathbf{x}'$) ▷ Query Selection
    $\mathbf{y} \leftarrow \mathbf{y} \cup$ VECLAPLACE($\mathbf{x}'$, $\mathbf{M}$, ϵ) ▷ Query
end for
$\hat{\mathbf{x}} \leftarrow$ LS($\mathbf{M}, \mathbf{y}$)
return $\hat{\mathbf{x}}$ ▷ Output

Selection in HB-STRIPED's subplans are data-independent, unlike in DAWA-STRIPED, so the exact same set of measurements will be selected on each partition. As introduced in Sec. 3.5.4, this set of measurements can be represented compactly as a Kronecker product. So we introduce a new selection operator STRIPE(ATTR) where a global measurement is composed by constructing the Kronecker product of HB measurements on the stripe dimension and Identity matrices on other dimensions. HB-STRIPED_KRON (PLAN #16) is a sequence starting with the new SS selection operator, followed by Laplace measurement and LS inference. The complete plan is shown in Algorithm 6. This non-iterative alternative implementation is more efficient and we compare the efficiency and scalability of the two implementations in Sec. 3.8.1.1.

Algorithm 6 HB-STRIPED_KRON

$D \leftarrow$ PROTECTED(source_uri) ▷ Init
$\mathbf{x} \leftarrow$ T-VECTORIZE(D) ▷ Transform
$\mathbf{M} \leftarrow$ STRIPESELECT(Attr) ▷ Query Selection
$\mathbf{y} \leftarrow$ VECLAPLACE($\mathbf{x}$, $\mathbf{M}$, ϵ_3) ▷ Query
$\hat{\mathbf{x}} \leftarrow$ LS($\mathbf{M}, \mathbf{y}$)
return $\hat{\mathbf{x}}$ ▷ Output

Our final plan is a variant of PRIVBAYES in which we replace the original inference method with least squares, retaining the original PRIVBAYES query selection and query steps. We call this algorithm PRIVBAYESLS and it's fully described in Algorithm 7.

Algorithm 7 PRIVBAYESLS

1: $D \leftarrow$ PROTECTED(source_uri)	▷ Init
2: $\mathbf{x} \leftarrow$ T-VECTORIZE(D)	▷ Transform
3: $\mathbf{M} \leftarrow$ PBSELECT($\mathbf{x}, \epsilon_2$)	▷ Query Selection
4: $\mathbf{y} \leftarrow$ VECLAPLACE($\mathbf{x}$, $\mathbf{M}$, ϵ_3)	▷ Query
5: $\hat{\mathbf{x}} \leftarrow$ LS($\mathbf{M}$, $\mathbf{y}$)	▷ Inference
6: **return** $\mathbf{W} \cdot \hat{\mathbf{x}}$	▷ Output

We evaluate the error incurred by these plans in Sec. 3.8.2.2, and show that the best of our plans outperforms the state-of-the-art PRIVBAYES by at least 10× in terms of error.

3.7.3 Naive Bayes case-study

We also demonstrate how ϵKTELO can be used for constructing a Naive Bayes classifier. To learn a NaiveBayes classifier that predicts a binary label attribute Y using predictor variables $(X_1, \ldots, X_k)$ requires computing 2k+1 1d histograms: a histogram on Y, histogram on each X_i conditioned on each value on Y. We design ϵKTELO plans to compute this workload of 2k+1 histograms, and use them to fit the classifier under the Multinomial statistical model [47].

We develop two new plans and compare them to two plans that correspond to algorithms considered in prior work. WORKLOAD represents the 2k+1 histograms as a matrix, and uses VECTOR LAPLACE to estimate the histogram counts. This corresponds to a technique proposed in the literature [18]. The other baseline is IDENTITY (Plan 1): it estimates all point queries in the contingency table defined by the attributes, adds noise to it, and marginalizes the noisy contingency table to compute the histograms.

The first new plan is WORKLOADLS which runs WORKLOAD followed by a least squares inference operator, which for this specific workload would make all histograms have consistent totals. Our second plan is called SELECTLS (fully described in Algo-

rithm 8) and selects a different algorithm (subplan) for estimating each of the histograms. SELECTLS first runs 2k+1 domain reductions to compute 2k+1 vectors, one for each his-

Algorithm 8 SELECTLS

```
D ← PROTECTED(source_uri)                                   ▷ Init
x ← T-VECTORIZE(D)                                          ▷ Transform
R ← MARGREDUCTION(x, Att)                                   ▷ Partition Selection
M ← ∅, y ← ∅
for i = 1 : k do                                            ▷ Iterate over Dimensions
    x' ← V-REDUCEBYPARTITION (x, R_i)
    if DomainSize_i > 80 then
        R' ← RDAWA (x', ε1/k)                               ▷ Partition Selection
        x'_R ← V-REDUCEBYPARTITION (x, R')
        M ← GREEDYH(x'_R)                                   ▷ Query Selection
        y ← y∪ VECLAPLACE(x'_R, M, ε2/k)                    ▷ Query
    else
        M ← M∪ IDENTITY(x')                                 ▷ Query Selection
        y ← y∪ VECLAPLACE(x', M, ε/k)                       ▷ Query
    end if
    x ← V-REDUCEBYPARTITION (x', R_i)                       ▷ Domain Expansion
end for
x̂ ← LS(M, y)
return x̂                                                    ▷ Output
```

togram. Then, for each vector, SELECTLS uses a conditional statement to select between two subplans: if the vector size is less than 80, IDENTITY is chosen, else a subplan that runs DAWA partition selection followed by IDENTITY is chosen. We combine the answers from all subplans and use least squares inference jointly on all measurements. The inputs to the inference operator are the noisy answers and the workload of effective queries on the full domain. In Sec. 3.8.2.3 we show that our new plans not only outperform existing plans, but also approach the accuracy of the non-private classifier in some cases.

3.8 Experimental evaluation

Our prototype implementation of ϵKTELO, including all algorithms and variants used below, consists of $7.9k$ lines of code: 25% is the framework itself, 46% consist of operator implementations, 14% consist of definitions of plans used in our experiments and the

remaining 15% are tests and examples provided for the users. We conduct experimental evaluation an Amazon EC2 2xLarge instance with 32GB of RAM running Ubuntu 14.04.

3.8.1 Performance improvements of implicit matrices

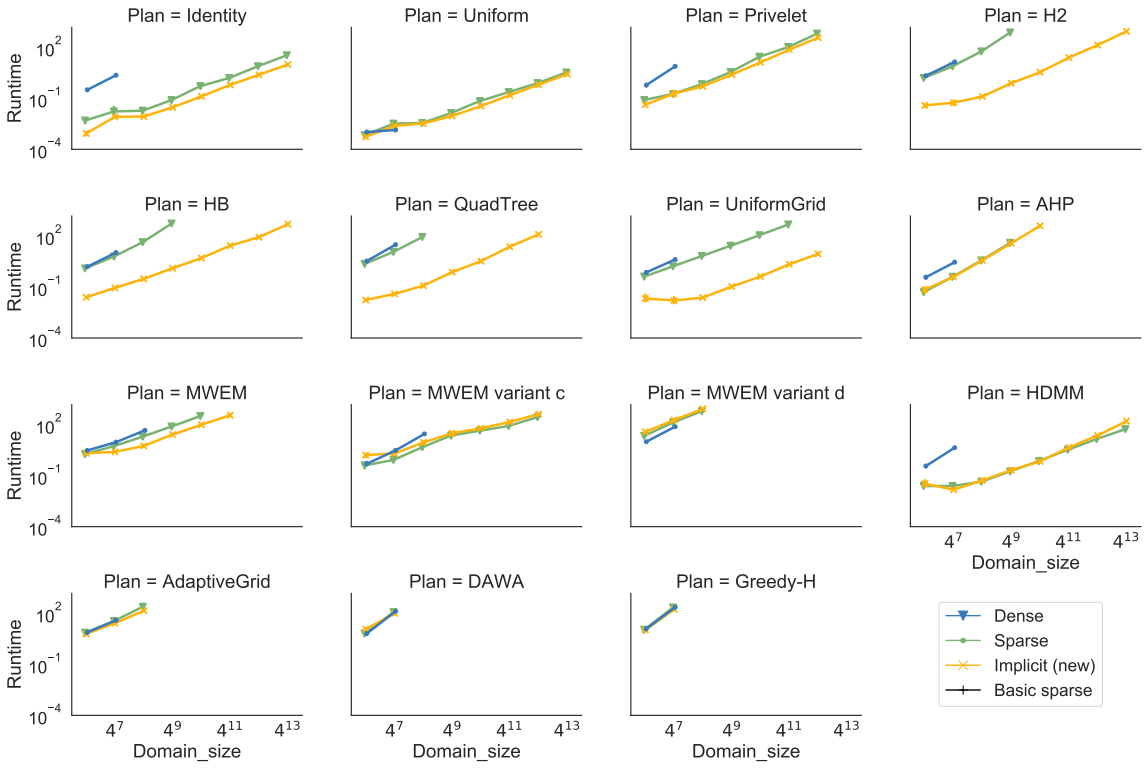
As discussed in Sec. 3.5, most ϵKTELO operators involve performing operations on matrices. All of them are lossless representations of the underlying matrix, so the choice of implementation does not influence plan accuracy. However, it could impact the efficiency and scalability.

In this section, we compare these alternative implementations. We first evaluate how the choice of matrix implemenation impacts scalability and efficiency of plans. Then we have a focused experiment on a key matrix operation: inference.

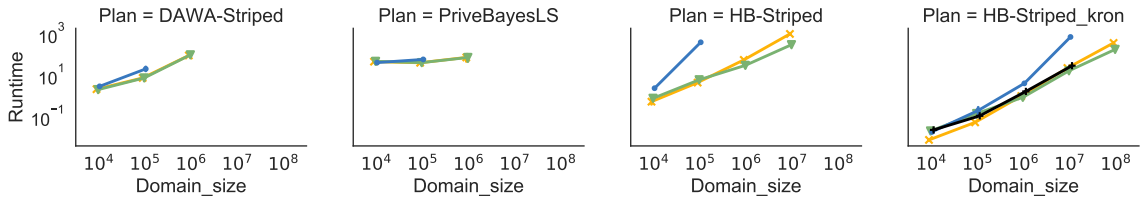
3.8.1.1 Scalability and efficiency of plans

To understand the impact of different physical implementations for plans in Fig. 3.2, we compare the runtime of plans using implicit measurement matrices with direct matrix implementations, which can use either dense or sparse matrices. For the plans HDMM and HB-STRIPED_KRON, which contain operators that were not supported in the previous implementation, we make the comparison by converting implicit matrices to their sparse and dense representations. We measure the average end-to-end execution time over 5 random trials for three implementations of each plan along increasing domain sizes. We stop any execution when it runs for more than 1000s.

Fig. 3.3a shows runtime for low-dimensional plans. All plans are applied on two-dimensional domain square domains, except for DAWA and GREEDY-H, which are designed for one-dimensional domains. PLAN #19 MWEM VARIANT B is omitted because it timed out even at the smallest domain we tested here. The results show that for most plans, the implicit implementation has the best scalability. Also, looking at a fixed domain size, the implicit representation usually leads to faster runtime than its dense and sparse counterparts.



(a) 1-D and 2-D plans



(b) Multi-dimensional plans

Figure 3.3: Plan execution time with different implementations of measurement matrices with Identity workload. Implicit represents plans implemented with the implicit matrices. Dense and sparse represent direct matrix implementations, and basic sparse is an alternative to Kronecker product (a type of implicit matrices) materialized as sparse matrices. Results show the new implicit representation can increase scalability by a factor up to 1000x.

The performance improvement is most pronounced with plans HB, QUADTREE and UNIFORMGRID where implicit representation can scale to domains larger by a factor of 1000x. These algorithms construct hierarchical/grid-based measurement matrices and can be represented as *Range Queries*, a special instance of the implicit matrices. As discussed in Sec. 3.5.5, this representation is compact and supports faster matrix-vector products. There are few cases where the difference between implementations is less pronounced. DAWA and GREEDY-H share the same special selection subroutine which needs to materialize the matrix. ADAPTIVEGRID has a plan that requires iterating through a potentially large number of partitions, and this step appears to dominate the runtime.

Results for high-dimensional plans are shown in Fig. 3.3b.

For the first three plans, sparse and implicit representations exhibit similar performance and scale to domains at least 10x larger than using dense. For the last plan, HB-STRIPED_KRON, recall from Sec. 3.7.2 that this plan is an alternative way of expressing the same algorithm as the HB-STRIPED plan, but instead of partitioning the data, it uses Kronecker products to express queries compactly in terms of submatrices. By comparing adjacent figures, we can see the approach based on Kronecker products allows plans to scale to at least 10x larger domains across implementations of the submatrices. As another comparison point that illustrates the benefits of Kronecker products, in the last figure, we include "Basic sparse", an alternative implementation of the HB-STRIPED_KRON plan where the query Kronecker product matrix is replaced with a materialized sparse matrix over the full domain.

3.8.1.2 Scalability of inference

Inference is one of the most computation-intensive operators in ϵKTELO especially for large domains resulting from multidimensional data. Next, we show the impact of implementation choices on the scalability of inference. Fig. 3.4 shows the computation

time for running our main inference operators (LS and NNLS) as a function of data vector size.

Recall that the methods described in Sec. 3.5.6 provide efficiency improvements by using iterative solution strategies (*iterative* instead of *direct* in the figure) and exploiting sparsity in the measurement matrix (*sparse* or *implicit* as opposed to *dense* in the figure). For this experiment, we fix the measured query set to consist of binary hierarchical measurements [40]. Fig. 3.4 shows that using sparse matrices and iterative methods allow inference to scale to data vectors consisting of millions of counts on a single machine in less than a minute. The use of implicit matrices permits additional scale-up for both LEASTSQUARES and NNLS. We also compare against the inference method introduced by Hay et al., denoted 'Tree-based' in the figure. It is an algorithm that is logically equivalent to LEASTSQUARES but specialized for hierarchically structured measurements. The general-purpose LEASTSQUARES implementation is able to scale to much larger domains.

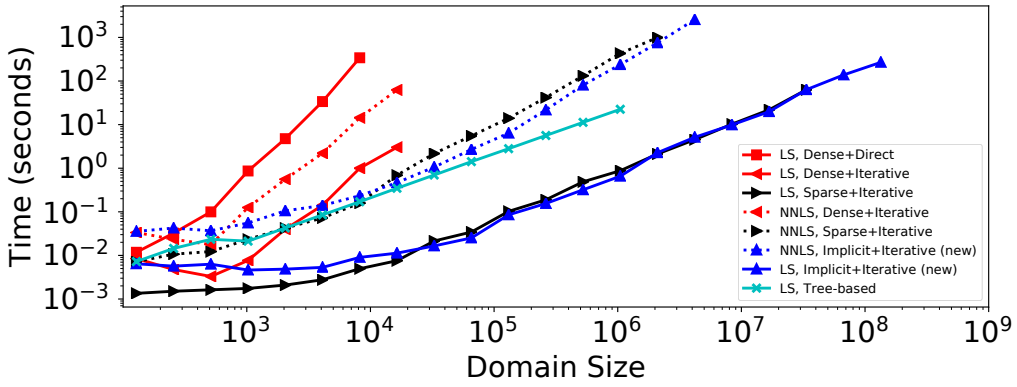


Figure 3.4: For a given computation time, the proposed iterative and implicit inference methods permit scaling to data vector sizes as much as $1000\times$ larger than previous techniques using direct approaches and dense matrices. Dense and sparse implementations are from [105] and tree-based is from [40].

3.8.2 Case studies

3.8.2.1 MWEM: improved query selection & inference

We evaluate the three new plans described in Sec. 3.7.1 which were variants of ϵKTELO plan for the MWEM [36] algorithm. Recall that the variants were achieved by replacing key operators in the MWEM plan. These algorithms are data-dependent algorithms so we evaluate them over a diverse collection of 10 datasets taken from DPBench [38]. The results are shown in Table 3.4.

Table 3.4: For three new algorithms, (b), (c), and (d), the multiplicative factors by which error is improved, presented as (min, mean, max) over datasets. For runtime, the mean is shown, normalized to the runtime of standard MWEM. (1D, n=4096, W=RandomRange(1000), $\epsilon = 0.1$)

	MWEM Variants		ERROR IMPROVEMENT			RUNTIME
	Query Selection	**Inference**	min	mean	max	mean
(a)	worst-approx	MW	1	1	1	1
(b)	worst-approx + H2	MW	1.03	2.80	7.93	354.9
(c)	worst-approx	NNLS, known total	0.78	1.08	1.54	1.0
(d)	worst-approx + H2	NNLS, known total	0.89	2.64	8.13	9.0

The performance of the first variant, line (b), shows that the augementing query selection with H2 can significantly improve error: by a factor of 2.8 on average (over various input datasets) and by as much as a factor of 7.9. (Error and runtime measures are normalized to the values for the original MWEM; min/mean/max error values represent variation across datasets.) Unfortunately, this operator substitution has a considerable impact on performance: the added queries slow down by a factor of more than 300. But combining augmented query selection with NNLS inference, line (d), reduces runtime significantly: it is still slower than the original MWEM algorithm, but by only a factor of 9. Using the original MWEM query selection with NNLS inference, line (c), has largely equivalent error and runtime to the original MWEM. Thus, the performance gains of NNLS inference over MW appear to be most pronounced when the number of measured queries is large.

Table 3.5: Results on Census data; domain size 1,400,000; scale of error is indicated under each workload.

	Workload		
	Identity	2-way Marg.	Prefix (Income)
Algorithm	$(1e{-}9)$	$(1e{-}7)$	$(1e{-}7)$
IDENTITY	24.18	12.04	18.97
PRIVBAYES	76.93	65.31	28.70
PRIVBAYESLS	5.86	13.29	36.81
HB-STRIPED	70.31	21.91	4.13
DAWA-STRIPED	**3.43**	**1.96**	**2.50**

3.8.2.2 Census data analysis

In this section we compare the ϵKTELO plans proposed in Sec. 3.7.2, measuring their effectiveness in computing workloads inspired by Census tabulations. We compare our three new plans PRIVBAYESLS, HB-STRIPED, and DAWA-STRIPED with baseline algorithms IDENTITY (PLAN #1 in Fig. 3.2) and PRIVBAYES, our ϵKTELO reimplementation of a state-of-the-art algorithm for high dimensional data [109].

Table 3.5 presents the results for each workload. We use scaled, per-query L2 error to measure accuracy. First, we find that PRIVBAYES performs worse than IDENTITY on all workloads. Interestingly, on Identity and 2-way marginal workloads, it is improved by our new plan PRIVBAYESLS that replaces its inference step with least squares. PRIVBAYES may be more suitable to input data with higher correlations between the attributes. Second, our striped plans HB-STRIPED and DAWA-STRIPED offer significant improvements in error. DAWA-STRIPED is the best performer: the data-dependent nature of DAWA exploits uniform regions in the partitioned data vectors. This shows the benefit from ϵKTELO in allowing algorithm idioms designed for lower-dimensional data to be adapted to high dimensional problems.

3.8.2.3 Naive Bayes classification

We evaluate the performance of the Naive Bayes classifier on *Credit Default* [102], a credit card clients dataset which we use to predict whether a client will default on their

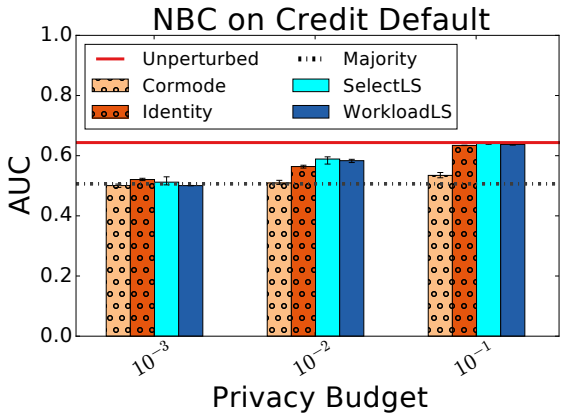


Figure 3.5: New ϵKTELO plans WORKLOADLS and SELECTLS result in NaiveBayes classifiers with lower error than plans that correspond to algorithms from prior work, and approach the accuracy of a non-private classifier for various ϵ values.

payment or not. The data consists of 30k tuples and 24 attributes from which one is the target binary variable "Default" and the rest are the predictive variables. We used the predictive variables $X_3 - X_6$ for a total combined domain size of $17,248$.

In our experiments we measure the average area under the curve (AUC) of the receiver operating characteristic curve across a 10-fold cross validation test. The AUC measures the probability that a randomly chosen positive instance will be ranked higher than a randomly chosen negative instance. We repeat this process 10 times (for a total of 100 unique testing/training splits) to account for the randomness of the differentially private algorithms and report the $\{25, 50, 75\}$-percentiles of the average AUC. As a baseline we show the majority classifier, which always predicts the majority class of the training data and also show the unperturbed classifier as an upper bound for the utility of our algorithms.

In Fig. 3.5 we report our findings: each group of bars corresponds to a different ϵ value and each bar shows the median value of the AUC for an algorithm. For each DP algorithm we also plot the error bars at the 25 and 75 percentiles. The dotted line is plotted at 0.5067 and shows the AUC of the majority classifier. The continuous red line is the performance of the non-private classifier (Unperturbed). For larger ϵ values we see that our plans significantly outperform the baseline and reach AUC levels close to the unperturbed. As ϵ decreases, the quality of the private classifiers degrades and for $\epsilon = 10^{-3}$ the noise

added to the empirical distributions drowns the signal and the AUC of the private classifiers reach 0.5, which is the performance of a random classifier. Our plan WORKLOADLS is essentially the algorithm of [19] with an extra inference operator. This shows that the addition of an extra operator to a previous solution significantly increases its performance.

3.8.3 Workload-driven data reduction

Next, we evaluate the impact of workload-driven data reduction, as described in Section 3.6.1. For selected algorithms, Table 3.6 shows that performing workload-driven data reduction improves *both* error and runtime, almost universally.

Table 3.6: Runtime (sec) and error improvements resulting from workload-based domain reduction. (W=RandomRange, small ranges. Original domain size: AHP (128,128), DAWA 4096, Identity (256,256), HB 4096)

Algorithm	Original Domain		Reduced Domain		Factor Improved	
	Error/Runtime		Error/Runtime		Error/Runtime	
AHP	1.68e−5	777.10	1.30e−5	145.00	1.29	5.36
DAWA	1.06e−5	0.23	1.07e−5	0.25	0.99	0.92
IDENTITY	4.74e−5	0.66	1.64e−5	0.90	2.89	0.73
HB	3.20e−5	0.05	2.38e−5	0.08	1.34	0.62

The biggest improvement in error (a factor of 2.89) is witnessed for the IDENTITY algorithm. Without workload-driven reduction, groups of elements of the domain are estimated independently even though the workload only uses the total of the group. After reduction, the sum of the group of elements is estimated and will have lower variance than the sum of independent measurements.

The biggest improvement in runtime occurs for the AHP algorithm. This algorithm has an expensive clustering step, performed on each element of the data vector. Workload-driven reduction reduces the cost of this step, since it is performed on a smaller data vector. It also tends to improve error because higher-quality clusters are found on the reduced data representation.

3.8.4 Summary of Findings

The experiments evaluate the accuracy, scalability, and efficiency of ϵKTELO. The study of scalability and efficiency found that implicit matrix representation can lead to huge performance gains, increasing scalability by a factor 1000x in some cases. The generalized inference implementation scales well and outperforms specialized algorithms. The case studies show that ϵKTELO can lead to more accurate algorithms with relatively little effort from the programmer: the MWEM algorithm can be improved significantly by replacing a few key operators; for the Census and Naive Bayes case studies, ϵKTELO can be used to design novel algorithms from existing building blocks, offering state-of-the-art error rates. Finally, the evaluation shows the workload-driven data reduction improves accuracy and runtime, almost universally, so that it can be added to all workload-based plans with little cost and significant potential for gains.

3.9 Related work

A number of languages and programming frameworks have been proposed to make it easier for users to write private programs [64, 77, 27, 84]. The *Privacy Integrated Queries* (PINQ) platform began this line of work and is an important foundation for ϵKTELO. We use the fundamentals of PINQ to ensure that plans implemented in ϵKTELO are differentially private. In particular, we adapt and extend a formal model of a subset of PINQ features, called Featherweight PINQ [27], to show that plans written using ϵKTELO operators satisfy differential privacy. Our extension adds support for the partition operator, a valuable operator for designing complex plans.

Additionally, there is a growing literature on formal verification tools that prove that an algorithm satisfies differential privacy [31, 9, 108, 4]. For instance, LightDP [108] is a simple imperative language in which differentially private programs can be written, allowing verification with little manual effort. LightDP's goal is orthogonal to that of ϵKTELO:

it simplifies proofs of privacy, while ϵKTELO's goal is to simplify the design of algorithms that achieve high accuracy.

Concurrently with our work, Kellaris et al. [45] observe that algorithms for single-dimensional histogram tasks share subroutines that perform common functions. The use of inference appears in many differentially private algorithms [100, 40, 8, 101, 49, 53, 111, 3, 19, 76, 48]. Proserpio et al. [76] propose a general-purpose inference engine based on MCMC that leverages properties of its operators to offset the otherwise high time/space costs of this form of inference. Our work is complementary in that we focus on a different kind of inference (based on least squares) in part because it is used, often implicitly, in many published algorithms.

A full treatment of automated plan optimization is an important future goal for ϵKTELO, however ϵKTELO could directly incorporate limited forms of automation already proposed in the literature. The matrix mechanism [53, 62] formulates an optimization problem that corresponds to automated query selection in ϵKTELO. Other recent work [47, 57] considers the problem of data-dependent algorithm selection. These methods could be adapted to automatically select from a set of predefined plans in ϵKTELO.

3.10 Discussion and limitations

In this section we discuss current limitations of ϵKTELO and opportunities for future work that will extend its capabilities.

3.10.1 More complex queries and tasks

ϵKTELO is extensible and we hope to expand the classes of tasks that can be supported. ϵKTELO is currently focused on statistical queries on a single table, or those tasks, such as naive Bayes classification, which can be directly supported by such statistics. Answering more expressive aggregate queries, for example those expressible as SQL queries over

multi-relational schemas, is possible using the general framework of ϵKTELO, but will require a number of extensions.

Another possible extension for ϵKTELO is adding support for aggregate functions over continuous and numerical domains. Queries like SUM, STDDEV, and MEAN can be used as building blocks for a wider class of algorithms (e.g., the naive Bayes classifier of [93] for continuous variables). To support those queries, ϵKTELO will need to know the domain of each attribute involved in these aggregations to correctly compute their respective sensitivities. For instance, given attribute A, with upper bound a_{max}, ϵKTELO can correctly compute the sensitivity of SUM(A) as a_{max}. Similarly and given domain information, ϵKTELO can provide upper bounds for the sensitivity of STDDEV, and MEAN. Lastly, note that support for user-defined queries for which sensitivity computation is non-trivial is still viable, as long as these queries are accompanied with a vetted sensitivity estimate.

3.10.2 Scaling to higher dimensions

Many of the ϵKTELO plans described eariler operate on a single materialized data vector, and the current version of ϵKTELO always keeps this data vector in memory. Recall that a data vector can grow as large as the product of the domain sizes of the attributes in the input table. We have proposed a number of innovations that allow even the most expensive operators (like inference) to scale to vectors as large as 10^8. But it is nevertheless clear that for high-dimensional data (or even data with a modest number of dimensions, but large attribute domains) plans attempting to fully materialize the data vector will be infeasible.

ϵKTELO can still be used to define and execute plans in the high-dimensional case. The principal difference from most plans explored in this work is that the plans will operate on *multiple* data vectors, each defined over a subset of the attributes of the input table. Specifically, such a plan would begin by applying multiple PROJECT transformations to the input table, to perform a relational projection onto a subset of the attributes. Then each would be followed by T-VECTORIZE, ultimately resulting in a collection of data vectors. A

plan could then apply Query Selection and Query operators to each vector independently, resulting in sets of differentially private estimates, each over a projection of the original data.

In fact, plans of precisely this form were designed in response to the recent *Differential Privacy Synthetic Data Challenge* [1] created by the U.S. National Institute of Standards and Technology (NIST). The challenge required competitors to generate differentially private synthetic data from a source dataset with 98 attributes. The resulting data vector would have approximately 5×10^{205} elements. Not only is this infeasible to represent, but, given that the input data contained just 660 thousand records, measurements offering reasonable utility under differential privacy will tend to be those that aggregate over many dimensions. Among the top four solutions to the final round of the challenge, none measured more than a three-attribute projection. The winning solution constructed approximately 200 such low-dimensional projections, performed query selection, and then post-processing.

The true challenge is indeed post-processing many independently-estimated noisy sets of queries. The methods proposed in this work, while substantially expanding the efficiency of inference, cannot support global inference over many data vectors because they rely on full materialization to relate the measured queries to one another. Some existing techniques have proposed methods for post-processing on extremely large domains by restricting the structure of the noisy queries [109, 36, 81], but they are not general solutions to the problem that can be effectively used within different types of plans. A more general solution to the problem was recently proposed that uses graphical models as a represention tool for the high-dimensional data distribution [63]. Such a technology could be used within ϵKTELO plans, allowing them to effectively combine evidence about multiple data vectors derived from the same data source.

In summary, while we believe ϵKTELO is a promising platform for plans involving high-dimensional data, further research and development is required to fully incorporate scalable post-processing methods, as well as to reconsider query selection operators for

plans that involve may projections over a single dataset. We note that recent research into (non-private) summarization of high-dimensional data may have a role to play here [71, 73].

3.10.3 Automated optimization

The ϵKTELO system is analogous to a query executor in a relational system: it allows plans to be specified and executed, but it does not create plans in response to a given query or task. Some operators perform *operator-level* optimization (e.g. the HDMM and Greedy-H query selection operators both automatically adapt to a given input workload). However, a compelling future goal for ϵKTELO is automatic *plan-level* optimization.

Much like a relational optimizer, we envision adding a component that can explore the plan space implicitly defined by the collection of operators implemented in ϵKTELO. But optimization here has dual objectives (both accuracy and efficiency must be considered) and, in addition, it may be important to accommodate user-defined accuracy metrics. Further, the accuracy of some plans depends on the input data and may incur a privacy cost if it is used naively by the system during optimization. Lastly, in classical optimization, the set of logically equivalent plans is well-defined, but for private algorithms, which are randomized, an appropriate definition of plan equivalence requires further investigation.

CHAPTER 4

INVESTIGATING STATIC VISUAL ANALYSIS OF DIFFERENTIALLY PRIVATE DATA

We've shown how frameworks like ϵKTELO can provide support in designing and implementing differentially private algorithms in the previous chapter. However, deploying those well-designed and carefully-implemented algorithms in real-world applications stays nontrivial. Among the wide range of potential applications, data visualization and visual analysis have been proven useful in many domains like health care, civic decision making. Yet how to generate private visualization or perform visual analysis in a private-preserving way remains a hard question. In the rest of this dissertation, we focus on the topic of privacy-preserving visualization topics. More specifically, in this chapter, we look deeper into issues with static visualizations and deal with the interactive case in Chapter 6.

Theoretically, it is possible to produce robust privacy-preserving visualizations by directly plotting differentially private data. However, noise-induced data perturbations can alter visual patterns and impact the utility of a private visualization. We still know little about the challenges and opportunities for visual data exploration and analysis using private visualizations.

Depending on the magnitude of data perturbation, moderate to extreme visual discrepancies can happen between a private visualization and its non-private counterpart. Recall the example in the introduction (Fig. 1.1). The notion of "success" between private and non-private visualizations is different. A user's success performing a task on private visualization depends not only on perceptual accuracy but also on the magnitude of data perturbation. Even when a user achieves perceptual accuracy and correctly identifies the

visual artifact of interest, any readings from the target will still be different from the non-noisy data. Depending on the distance between noisy and non-noisy values, the user can achieve degrees of success. In this case, accuracy is a fuzzy variable with degrees of a pass or fail. This phenomenon imposes a grand challenge and raises a critical question: is it possible to perform visual data analysis on differentially private visualizations and trust the outcomes? For instance, do patterns in a private line chart indicating improvements in patients' conditions on specific treatment match similar patterns in the non-private line chart?

Existing work in the confluence of privacy and visual data analysis has been mainly focused on the use of syntactic privacy models such as k-anonymity and l-diversity (e.g.,[96, 15, 14]) and we know little about the challenges and opportunities of supporting visual data analysis under differential privacy. To fill this gap, we investigated the following two research questions:

- (RQ1) What is the relationship between the noise-injection level, visualization type, data analysis task, and users' performance (accuracy and time to complete tasks)?
- (RQ2) Is it possible to tune noise injection to improve the utility of private visualizations?

To investigate RQ1, we performed a crowd-sourced user study and examined the effects of three privacy levels (high, low, non-private) for combinations of eight analysis tasks and four visualization types (bar chart, pie chart, line chart, scatter plot). A central challenge in this phase was the assessment of a user's accuracy and task success. Injection of noise and consequent perturbations of data and visual patterns can result in erroneous findings, even if a user's answer to a task is correct based on the private visualization presented to the user. We set forward a dichotomous assessment method that measures accuracy and task success based on the notions of perceptual and perturbation accuracy. In our study, we only considered univariate visualization. The main rationale behind this decision was

to eliminate the possible interactions between two sets of noisy variables and their possible effects on participants' performance. We measured the task success rate and response time for 204 participants. We found that the rate of user success dropped for all tasks as the noise level increased. However, the rate of decline was not consistent across all tasks. In particular, "summary tasks" (e.g., Characterize Distribution) seem to be less sensitive to the injection of noise in comparison to "value tasks" (e.g., Compute Derived Value).

There has been prior research on designing differentially private algorithms [26, 35, 49], and on the comparison of algorithm performance [39] for answering range queries. However, it is unclear how noise injection influences downstream visual tasks. Given a certain privacy protection level, various differentially private noise injection algorithms could produce completely different outputs. When visualized, some outputs could contain visual artifacts that make visual tasks substantially harder for users. We investigated the possibility of tuning noise injection through wisely choosing algorithms to improve perceptual accuracy for specific visual tasks. We introduced three basic *distribution metrics* to quantify the shape of noisy algorithm outputs, measuring to what extent they preserve prominent visual features essential for a specific task. *Peakedness Score*, *Anomaly Score*, and *Clusteredness Score* respectively quantify to what extent there exists a single peak, an anomaly data point, and clear cluster boundaries. Then we performed several rounds of simulations using three popular differentially private algorithms Laplace[26], DAWA[49], and MWEM[35] and compared the perceptual distribution metrics of their noisy output. The results of these simulations indicate that the Laplace mechanism works better across different tasks compared with the more complex DAWA and MWEM algorithms.

In the rest of this chapter, we first provide a review of related literature. Next, we present detailed descriptions of work performed to investigate both research questions. Then, we present a comprehensive discussion of findings and a set of empirical guidelines. We conclude the chapter with the limitations of our work and interesting future directions.

4.1 Private Visualizations in the Literature

4.1.1 Privacy-preserving Visualizations

The goal of a privacy-preserving visualization is to protect individuals' identities and sensitive information from exposure while still allowing users to make sense and gain knowledge from it. Some prior work in this area has investigated the use of visual uncertainty for preserving privacy. Dasgupta and Kosara [22] introduced a pixel-based clustering technique for parallel coordinates called "Screen-Space Sanitization" that combines pixels to increase visual uncertainty in areas of visualization where privacy could be breached. Using a similar approach, Archambault et al. [7] and Oksanen et al. [69] suggest the aggregation of visual components for building privacy-preserving histograms and heatmaps. Deliberate reduction of visual accuracy increases visualization uncertainty and reduces the possibility of guessing the exact values, but does not satisfy a rigorous definition of privacy that can provably resist attacks.

Data sanitization techniques (e.g., k-anonymity, l-diversity, and t-closeness) have also been investigated for building private visualizations. GraphProtector [96] supports building privacy-preserving graphs of social networks. Users can combine multiple privacy protection schemes as a hybrid approach to fine-tune privacy protection. Chou et al. utilize data sanitization to build private Sankey and Iceplot visualizations for representing temporal event sequence data [15] and constructing private network visualizations [14].

Bhattacharjee et al. [10] provided a thorough and systematic analysis of state-of-the-art approaches, methods, and techniques used in privacy-preserving data visualization, and reflected on a wide range of challenges and research opportunities. Prior work mainly assumes that the data owner designs and deploys privacy mechanisms specific to the domain and visualization types, and the end-user consumes the private visualization product. This approach enables building effective private visualizations for specific tasks, visualizations types, and data domains. However, it may not work in exploratory data analysis where a user's questions and tasks are not known in advance. Currently, we lack a domain-agnostic

understating of the relationship between tasks, visualizations, and privacy. The majority of prior work has also been focusing on using syntactic privacy models based on anonymization, which have fallen prey to a range of attacks [24, 68, 67].

To fill these gaps, we investigate the use of differential privacy in exploratory data analysis. Wang et al. [97] developed a visualization technique that helps users to dynamically gauge the loss of utility for a single task by anonymizing multi-attribute tabular data through building matrix-based and tree-based models for utility and privacy. In this work, we aim to better understand the relationships between noise level, task, visualization, and participants' performance. Instead of proposing an approach to modify a specific visualization technique that meets the privacy guarantee, we investigate a more general pipeline of private visualization where we can easily swap in different private algorithms or visualization techniques.

4.1.2 Privacy-utility Trade-off

There is an inevitable privacy-utility trade-off associated with all privacy-preserving mechanisms. Typically, a stricter guarantee of privacy results in more loss of information and hence lowers the accuracy and reduces analysis utility. For example, in the differential privacy model, using a smaller ϵ provides a stronger privacy guarantee, but reduces the accuracy of data analysis due to larger perturbation of data.

Deploying privacy mechanisms while balancing the privacy-utility tradeoff is a non-trivial task. It requires proper measurements and an understanding of both privacy and utility. The data mining community has proposed several metrics for evaluating the utility and quality of data after anonymization/privacy-preservation methods are applied. For example, DPBENCH [39] is a principled framework for evaluating differential privacy algorithms for answering 1-D and 2-D range queries. Dasgupta et al. [22] consider discernibility as a utility metric that measures the number of records that cannot be distinguished from one another. In the visualization field, Dasgupta et al. [21] state that utility can be re-

garded as a function of visual uncertainty. They introduce metrics for quantifying the visual uncertainty in cluster-based visualizations such as scatterplot and parallel coordinates.

An investigation by Zhang et al. [103] shows that the utility of performing visual tasks does not line up well with accuracy measures commonly used for algorithms answering range queries. In fact, the authors of [11, 56, 15] argue that the evaluation of utility should depend on user analytical tasks and how accurately they can be performed. Prior research (e.g., [21, 15, 14] has only investigated the impact of privacy on utility for a specific task, data, and visualization type. Taking a domain- and data-agnostic approach, in this work, we investigate the privacy-utility tension for basic data exploration tasks and visualization types.

Outside the realm of privacy, Saket et al. [85] conducted a study to investigate the effectiveness of five basic visualization types in relation to ten common data exploration tasks. The effectiveness of visualization consists of three main metrics: the success rate of performing the task, the performance time, and user preference. Results show that the effectiveness of visualization varies significantly across tasks. To understand the effectiveness of private visualizations and how conclusions might change under privacy, we use a similar experimental design like the set of tasks and visualization.

4.2 RQ1: Investigating the Utility of Private Visualizations

To investigate RQ1, we conducted a crowd-sourced empirical study on Amazon Mechanical Turk[1]. The rest of this section provides detailed information about the design of the study, the data analysis process, and our findings.

4.2.1 Dataset

IPUMS-CPS[89] is a collection of datasets that harmonizes microdata from the monthly U.S. labor force survey and the Census Current Population Survey (CPS), covering the

[1]https://www.mturk.com

period 1962 to the present. We used a data extract which is a subset of the Annual Social and Economic (ASEC) data from 2010. We chose this population survey data for two main reasons. First, it is similar to the data that could soon be protected (by the U.S. Census Bureau) using differential privacy. Second, it contains data attributes with which many study participants will be familiar, hence reducing the chance of failed tasks due to a user's unfamiliarity with the data semantics.

The data extract contains personal survey information on 159,277 individuals, each contributing one row to the dataset. For our experiment, we selected a subset of numerical and categorical attributes: Age, Sex, Race, Marital status, Education status, and Total income. Selected data were organized in a tabular format where each row represented information about an individual.

4.2.2 Private Algorithm

We selected the Laplace Mechanism as the privacy algorithm in our study. Although there are several other algorithms (i.e., [49, 35]) for generating private histograms, Laplace is relatively simple, fast, and competitive for the task of generating a single private histogram [39]. It offers a good compromise in terms of speed and utility and acts as a building block for many more complex algorithms.

In a post-processing step, we modified the output of the Laplace mechanism and replaced all the negative counts with zeros. The rationale behind this decision was to eliminate the change of producing private histograms with negative values for some of the bins which are clearly invalid. As introduced in Chapter 2, the differential privacy guarantee will not be compromised by this simple post-processing step. We consider the non-negative histogram to be the final privatized data and use it in later visualization steps. Since all differentially private algorithms are randomized, any single output is only a random sample from a distribution. For each experimental setting, we choose five random seeds that define

randomized trials. Visualizations and tasks are judged on the average performance over these random trials.

4.2.3 Privacy Parameter Setting

In this study, we considered three privacy levels of privacy parameter: 1) $\epsilon = \infty$, 2) $\epsilon = 0.01$, and 3) $\epsilon = 0.001$. A smaller privacy parameter enforces stronger privacy. An $\epsilon = \infty$ results in a non-private visualization which offers no privacy guarantee but also implies no noise. This non-private setting provided a baseline against which we assessed the utility of private visualizations. The value $\epsilon = 0.001$ offers a stronger privacy guarantee and requires more distortion of the data compared to the value $\epsilon = 0.01$. Both values might be considered low ϵ in practice. However, we are studying the release of a single visualization while in practice many releases would be made and a global ϵ bound needs to govern all interactions. In addition, the impact of noise is dependent on the size of the dataset thus the choice of ϵ is also dependent on the dataset. The selection of these ϵ thresholds was based on experimentation and preliminary testing with the study dataset. In particular, we paid careful attention that selected ϵ values offer strong privacy but are not too strict that the noise added leads to completely useless private visualizations.

Our goal in this work is to understand the regime in which noise from the privacy mechanism impacts the utility of private visualizations. As such, these carefully engineered ϵ values suit our purpose.

4.2.4 Participants

We recruited a total of 204 subjects based in the U.S., with an approval rate greater or equal to 95%. Each subject was allowed to participate in the study only once.

4.2.5 Tasks

Following prior research on task-based effectiveness of basic visualizations [85], we used the Amar et al. [5] taxonomy of low-level data analysis tasks for selection of study

tasks. Those tasks are real-world tasks users came up with while exploring datasets with different visualizations tools and have been used in different studies for visual effectiveness evaluation. We excluded two tasks Find Correlation and Order which involve two variables and could not be performed on univariate visualizations. The 8 selected low-level tasks act as building blocks of more complex tasks. The following is a list of selected tasks. For each task, we also provide a concise explanation of how they were used, along with an example. We use the term "visualization feature" in the following descriptions to refer to element(s) in visualization such as: a bar in a bar chart, a group of points in a scatter plot, or a peak in a line chart:

Retrieve Value We asked participants to retrieve the value of a certain visualization feature. For example, *what is the number of people in the Age Range 15-20?*

Filter Given a range, we asked participants to identify visualization features in that range. For example, *which Age Ranges have Group Size between 25,000 and 35,000?*

Compute Derived Value We asked participants to derive a new value using the visualization. For example, *what is the sum of Group Sizes for Marital Status Single and Divorced?*

Find Maximum We asked participants to find the visualization feature with the largest value. For example, *which Income Range has the largest Group Size?*

Determine Range For a set of visualization features, we asked participants to identify the range of their values. For example, *what is the range of Group Sizes for all Age Ranges? Select the correct pair of minimum and maximum Group Size.*

Characterize Distribution Given a condition, we asked participants to identify the distribution of values based on the condition. For example, *what is the percentage of Income Ranges that have Group Size larger than 25K?*

Find Anomalies We asked participants to find visualization features with abnormal values. We manually modified the visualizations to include easy-to-detect anomalies like zero counts and extremely large counts. For example, *which Age Range has abnormal Group Size?*

Cluster We asked participants to put visualization features of similar values into the same cluster and report the number of clusters. For example, *what is the number of clusters based on the Group Sizes of Income groups?*

4.2.6 Visualization Types

In this study, following prior empirical work [85], we examined four visualization techniques that are commonly incorporated in various visualization dashboards [**?**]: bar chart, pie chart, line chart, and scatterplot. To generate the visualizations, we took the private output of the Laplace mechanism and used the Matplotlib [**?**] visualization library to plot the privatized data. To maintain visual consistency, we fixed the chart size to be 500 by 350 pixels and the font size 12. We used Matplotlib's default color palette to generate pie charts and the same blue color for visual elements in other plots. The participants only viewed the final visualization and were not aware of the existence (or lack thereof) of noise in the encoded data.

4.2.7 Experimental Procedure

Instructions and warm-up tasks At the beginning of each study session, a participant was given a brief written description of the purpose of the study, the data that would be collected, and their rights, together with a consent form. Upon consenting to participate, the participant was given information about the workflow of the study and a quick optional tutorial explaining the visualization techniques used in the study. Next, the participant performed a short warm-up session answering one sample question for each task. Warm-up questions were similar to the actual study questions but performed on a manually synthe-

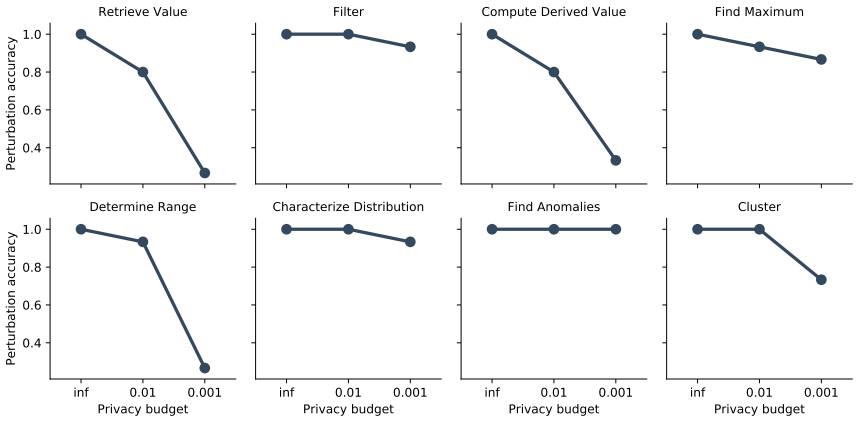


Figure 4.1: Perturbation accuracy of different tasks for the non-private case (ϵ=inf), a low privacy level (privacy parameter $\epsilon = 0.01$) and a high privacy level (privacy parameter $\epsilon = 0.001$). Accuracy decreases as we spend less privacy budget. But the accuracy drop is more severe for tasks involving numerical value retrieval or estimation (e.g. Retrieve Value, Compute Derived Value)

sized dataset. During the warm-up practice, the participant received feedback on whether their answer was correct along with corresponding explanations. After the successful completion of the warm-up session, the participant moved on to the actual study tasks.

Main questions For each combination of task, visualization, and privacy level, we sampled 3 univariate histograms from 3 different attributes. As introduced in Chapter 2, differentially private algorithms are randomized, making every output a single sample of a probability distribution. Thus, for each experimental configuration, we generated five differentially private outputs using different random seeds. This results in a total of 180 different questions ($4\ Visualizations \times 3\ Histograms \times 3\ Privacy\ Levels \times 5\ Random\ Seeds$) for each task.

In the main experiment, each participant answered a sequence of 48 multiple-choice questions, organized as 6 randomly sampled questions for each of the 8 tasks. For each question, we showed the user a single private visualization together with a brief description of the task and data. Given the visualization, we asked the participant to answer a single question associated with one of the eight tasks outlined above. The participant would move

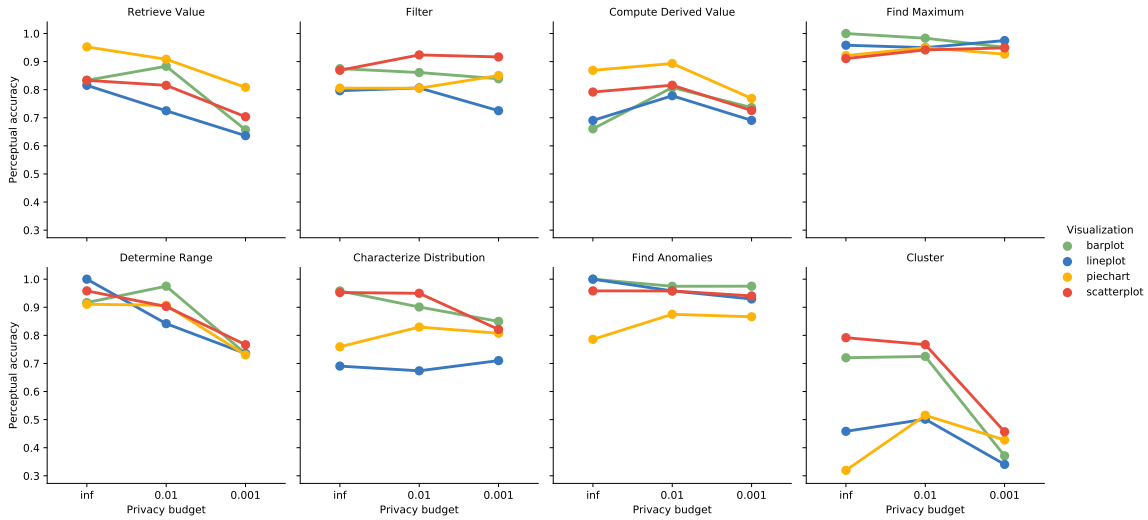


Figure 4.2: Perceptual accuracy for different analysis tasks, visualization types using different privacy budget. As the privacy level gets stricter (less privacy budget), the perceptual accuracy changes for some configurations. The noise added for privacy protection needs influences people's ability to perform visual tasks.

on to the next question after submitting their answer to the current question. The average completion time for the study was around 15 minutes, and we paid each user two dollars.

Validation Low-quality responses are not rare in online crowd-sourcing studies. To have a better sense of the overall quality of a user's responses, we introduced validation questions as recommended in [70]. For each worker, we randomly injected four validation questions, which are exact replicates of the warm-up questions seen before in the instructions. We considered the responses provided by a worker valid only if they answered three out of the four validation questions correctly. Otherwise, all responses from the worker would be discarded since they either failed to understand the task or perhaps made selections randomly.

Data collection Throughout the study, we collect worker responses to the multiple-choice questions and record the time they used to answer each question. At the end of the study, we ask the worker to fill out a simple demographic questionnaire asking about their gender and approximate age. Finally, we ask them to rate the overall difficulty of the question on a ten-point scale and provide short text feedback if they choose.

4.2.8 Data Analysis

We collected responses from a total of 204 MTurk workers and filtered out low-quality data according to the validation criteria described in Section 4.2.7. Among these workers, 176 finished the study and provided valid responses (105 male, 69 female, 2 other). 73% workers are in the age range from 18 to 40 years old; the rest are above the age of 40. On a scale of 1-10 (where 10 is the most difficult), the workers reported an average difficulty score of 4.25 (out of a maximum of 10). This shows our questions have a reasonable level of difficulty and that most people did not find the tasks confusingly difficult or trivially easy.

We analyzed collected data to quantify participants' performance in terms of time and accuracy under different experimental conditions. While the analysis of performance time was a straightforward process, the assessment of accuracy was challenging. Typically, a user's success in performing a task on a non-private visualization is assessed by determining whether the user 1) correctly identifies the visual artifact(s) of interest as requested by the task, and 2) correctly decodes the visualization to retrieve or derive values and draw conclusions. However, a similar assessment of user success cannot be used under private visualization. Due to the injection of noise and consequent perturbation of data, evaluating only the two aspects cannot guarantee user success in visual tasks with respect to the underlying sensitive data. In this work, we evaluate task success in terms of *perceptual accuracy* and *perturbation accuracy*.

4.2.8.1 Dichotomous Assessment of Task Success under DP

Perceptual accuracy To capture the information loss in human perception and cognition while performing visual tasks, we measure the *perceptual accuracy* defined as the rate of "perceptual successes". Here we compare the participant's response with the task answer based on the encoded data, whether or not noise has been added to the visualized data. The response is considered a "perceptual success" if it matches the encoded answer, although

it could be different from the answer based on the true data. For instance, if a user task was to "find the group with largest value" using a private bar chart, we considered the user perceptually accurate only if they successfully found the highest bar in the noisy bar chart. Perceptual accuracy describes a user's ability to perform visual tasks, consistent with the assessment of visual effectiveness in prior work [46, 85].

Perturbation accuracy

Continuing with the example task of finding the group with the maximum count, there are other potential sources of task failure. Even if the user had a perceptual success, there are chances that the highest bar in the private visualization is different from the one in the non-private visualization due to noise injection. In such a case, the user still failed to gain accurate information from the non-private data source. To isolate and measure the information loss from noise injection of the private algorithms for data exploration tasks, we introduce the notion of a "perturbation failure". We compare the results of performing a task on the non-private sensitive data and its privatized counterpart after noise injection. If there is a mismatch, then it is considered a "perturbation failure". *Perturbation accuracy* for a task is defined to be the rate of perturbation success after noise injection.

Perturbation for privacy may dramatically change the overall patterns of data, causing failures for summary tasks. It may also lead to changes in individual values, making tasks related to value retrieval fail. Due to the injection of noise, exact retrieval of data value is not feasible. Taking a heuristic approach, we considered an *error tolerance range* to decide if the distance between the non-private and private answers was acceptable. More specifically, we considered a range of $\pm$ 1K in which estimation errors were tolerated. For instance, for the non-noisy value of 10K, we accepted any answer in the range of [9K-11K] as acceptable. This error tolerance range was based on careful experimentation with our dataset and consideration of the range and distribution of data values. The histograms used in the study have average values of around 10K to 20K, so the tolerance range is approximately 5% to 10% of the data values.

We consider a visual task successful if both the perturbation and perceptual conditions are satisfied, meaning the information needed to perform the specific task is preserved after the perturbation and the user successfully performs the task on the noisy data. We evaluate all the tasks in relation to perceptual accuracy and perturbation accuracy using this dichotomous model.

In the analysis of perceptual effectiveness, we conducted a one-way repeated measures analysis of variance (ANOVA) for each task to test the differences of effectiveness across different visualizations and tasks. The performance time data was not normally distributed, so it was log-transformed to meet the normality assumption. In the analysis of perturbation error, to avoid the influence of perceptual uncertainty, we conduct the task using only the data values without any visualization. The calculation of perturbation accuracy is done off-line and involves no users.

4.2.9 Findings

Task	Perceptual accuracy		
	Better	Worse	ANOVA
Retrieve Value	pie chart	-	$F_{3,44} = 2.61, p < 0.05$
Filter	scatterplot	-	$F_{3,44} = 5.35, p < 0.01$
Compute Derived Value	-	-	$F_{3,44} = 1.19, p > 0.05$
Find Maximum	-	-	$F_{3,44} = 0.90, p > 0.05$
Determine Range	-	-	$F_{3,44} = 0.28, p > 0.05$
Characterize Distribution	-	line chart	$F_{3,44} = 12.14, p < 0.01$
Find Anomalies	-	pie chart	$F_{3,44} = 8.86, p < 0.01$
Cluster	scatterplot	-	$F_{3,44} = 3.74, p < 0.01$

Table 4.1: Response time comparison for each task showing visualization types that are significantly faster or slower than others.

First, we investigate the perceptual effectiveness of the four visualization types and compare the success rate and (log-transformed) response time. Consistent with effectiveness comparisons conducted in an earlier study [85], we found that bar chart and scatterplot are the visualization types that are most accurate and have the best response times, while

Task	Response time		
	Faster	Slower	ANOVA
Retrieve Value	bar chart	line chart	$F_{3,44} = 4.71, p < 0.01$
Filter	scatterplot	-	$F_{3,44} = 6.81, p < 0.01$
Compute Derived Value	-	pie chart	$F_{3,44} = 4.64, p < 0.01$
Find Maximum	-	pie chart	$F_{3,44} = 3.47, p < 0.05$
Determine Range	-	pie chart	$F_{3,44} = 4.50, p < 0.01$
Characterize Distribution	scatterplot, barchart	pie chart, line chart	$F_{3,44} = 15.38, p < 0.01$
Find Anomalies	-	pie chart	$F_{3,44} = 35.57, p < 0.01$
Cluster	scatterplot	pie chart	$F_{3,44} = 22.84, p < 0.01$

Table 4.2: Perceptual accuracy comparison for each task showing visualization types that are significantly better or worse than others.

pie chart has the worst response time but higher accuracy than line chart. Table 4.1 and 4.2 show detailed comparisons with the results of ANOVA tests.

Next, Fig. 4.1 shows the perturbation accuracy trends with varying privacy levels across analysis tasks. For all the tasks, task success rates on perturbed data drop as we move to higher noise levels (i.e., smaller ϵ). This was an expected outcome since the privacy-utility tension is a known phenomenon under differential privacy. While the rates of accuracy decline seem to be almost consistent within the tasks, there are noticeable differences between them. In particular, we found perceptible differences between *summary tasks*, including Filter, Characterize Distribution, Find Anomalies, Find Maximum, and *value tasks* including Retrieve value, Compute Derived Value, and Determine Range. At a similar level of noise, the rates of accuracy loss for value tasks are higher than those of summary tasks. This finding suggests that different tasks might have varying degrees of *noise tolerance*. The task Cluster showed a mixed pattern where the success rate was highly preserved for a lower level of noise but then sharply plummeted as the noise increased.

Furthermore, we want to see if noise injection influences the users' ability to perform visual tasks. Small multiples in Fig. 4.2 show the further breakdown of perceptual accuracy over different privacy levels. We omit the result for performance time because there is

no significant difference in participant response time across different noise levels. It is interesting to see that the average perceptual accuracy drops slightly as we move to stricter privacy levels (i.e., smaller privacy budgets) for many tasks and visualizations. For task Characterize Distribution, the privacy budget has significant impact on the visual accuracy of scatterplot ($F_{2,14} = 2.99, p < 0.1$). For task Determine Range, the privacy budget used influences visual accuracy of bar chart ($F_{2,14} = 4.94, p < 0.1$). For task Cluster, visual perception of both scatterplot ($F_{2,14} = 5.64, p < 0.1$) and bar chart ($F_{2,14} = 7.13, p < 0.1$) are influenced by the noise level. This shows that visual tasks, especially the more complex summary tasks, can become harder for people when the underlying data is noisy. In other words, the information loss from data perturbation could lead to a higher level of uncertainty in visual perception. However, unlike perturbation accuracy, the influence of noise injection isn't always monotonic. Users' ability to accurately perceive information from visualizations could increase or decrease as more noise is added. To better understand this effect, we conduct a second phase of our study in the next section.

4.3 RQ2: Tailoring Noise Injection to Analysis Task

Prior work in data visualization has shown that characteristics of underlying data such as distribution shape impact visualization in perceptible ways [46, 90, 72, 94]. As a simple example, it is easier to find the bin with maximum value in a unimodal histogram with values $\langle 11, 10, 30, 12 \rangle$ than a flat histogram with values $\langle 11, 10, 13, 12 \rangle$. The same level of privacy protection can be achieved by various DP algorithms (e.g.,[49, 35]). However, depending on the specific mechanism chosen, the injection of noise can lead to entirely different data distributions, which are consequently reflected in private visualizations. Drawing on the findings from both fields, we investigated the possibility of tuning the noise injection to result in a private data distribution that will facilitate performing a certain task assuming we have the advanced knowledge of the task at hand and privacy level required.

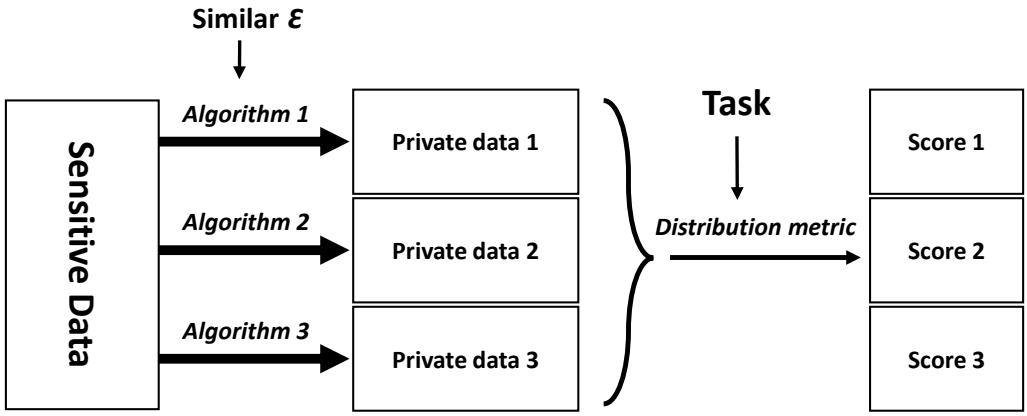
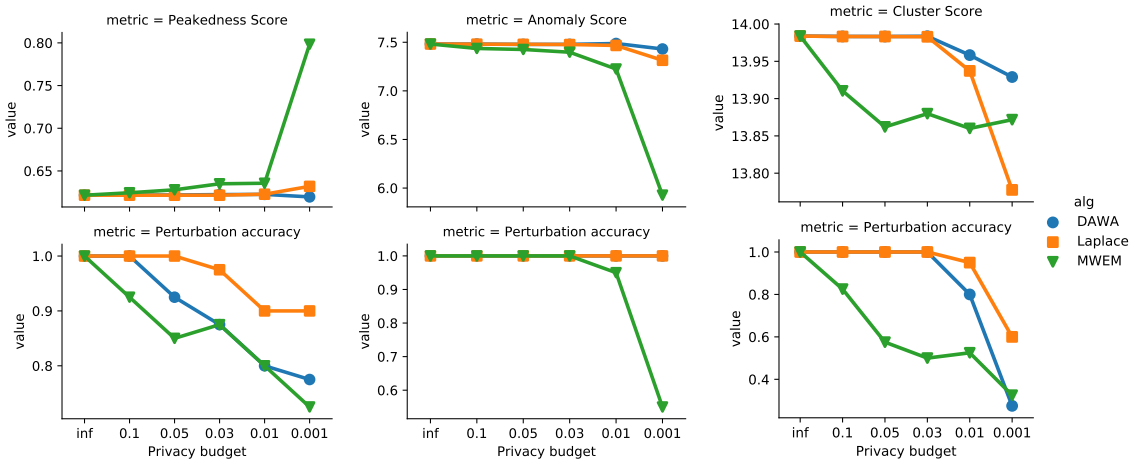


Figure 4.3: This figure shows our proposed model for tuning noise injection to tasks using our suggested distribution metrics. First, sensitive data is privatized using alternative DP algorithms (e.g. Algorithms 1, 2 & 3). All the privatized data meet a certain required level of privacy (similar ϵ). Next, based on the task at hand (e.g., Find the group with maximum value), the related distribution metric (e.g., Peakedness Score) is utilized to calculate a score for each set of privatized data. The privatized dataset with the highest score offers a data distribution shape that will better support the task.

In this work, we suggest a way of tuning the noise injection for summary tasks. Fig. 4.3 provides a schematic of our approach. At the core of our model lie "distribution metrics" that quantify the distribution shape of private data. This enables us to compare the privatized output of several algorithms and select one that would best support a task. In the following section, we provide details of our suggested metrics:

4.3.1 Distribution Metrics

Inspired by prior work on Scagnostics (e.g., [98, 99, 20]), we suggest three *distribution metrics* which quantify the shape of the data distribution. Each metric is designed and corresponds to a specific summary task. The reason for focusing on summary tasks was that the success of these tasks mainly relies on the user's perceptual accuracy which in turn is related to the shape of the data distribution [94, 95]. While for value tasks, there are no consistent relationships between people's ability to read a single data point and the overall data distribution.



(a) Task Find Maximum (b) Task Find Anomalies (c) Task Cluster

Figure 4.4: For the three summary tasks considered, the upper row shows distribution metrics for different algorithms and the lower row shows task success rate from data perturbation at the corresponding privacy level.

Peakedness Score: this metric is designed for the Find Maximum task. For $\vec{x}$, it is calculated as:

$$P = \frac{m_1}{\sum_{i=1}^{n} x_i} + 1 - \frac{m_2}{m_1}$$

where m_1, m_2 are the largest and second largest value of the distribution $\vec{x}$. For higher Peakedness Scores, the max point stands out more and it is easier to perform the task Find Maximum accurately.

Anomaly Score: this metric is designed for the Find Anomaly task. It is calculated as:

$$A = \frac{\max(|x_i - \text{mean}(\vec{x})|)}{\text{std}(\vec{x})}$$

it finds the furthest point from the sample mean and normalizes the distance by the standard deviation of the distribution. The higher the score is, the more likely the point is an outlier and it is easier for a user to detect it visually.

Cluster Score: this metric is designed for the task Cluster. It is calculated using weights from soft clustering which yield soft assignments of data points to clusters. We generate the soft clustering weights from existing clustering results. In this work, we use the Mean-shift [16] clustering algorithm which iteratively moves data points towards the mode. Unlike many other popular cluster algorithms, it does not require a pre-specified number of clusters. More specifically, given data $\vec{x}$, each data point x_i is assigned to a cluster $Clu(j)$ using the hard mean-shift clustering. The soft assignment weight is the likelihood that a data point i belongs to cluster j:

$$L_{i,j} = \frac{1}{\sum_{k=1}^{c} (\frac{x_i - c_j}{x_i - c_k})^2}$$

where c_j refers to the center of the jth cluster.

The Cluster Score of a distribution is the sum of likelihoods for all data points in the assignment, $L = \sum_{i=1}^{c} L_{i,Clu(i)}$ where $Clu(i)$ is the assigned cluster index for each data point i. Similarly, the higher the clustering score, the more likely there will be coherent clusters with clear boundaries.

4.3.2 Preliminary Evaluation

To assess the feasibility of using our suggested model to tune the injection of noise for summary tasks, we investigate the influence of privacy mechanisms on the data distribution. More specifically, we chose three representative differentially private algorithms and empirically compare their output at the same privacy levels over multiple runs.

With metrics to measure the visual difficulty of data distributions, we still need to understand how noise injection influences these metrics and later visual perception. Next, we compare three widely used differentially private algorithms in terms of their impact on different distribution metrics. Besides the Laplace algorithm, we also consider the MWEM

[35] algorithm which iteratively updates the histogram using noisy data estimations and the DAWA [49] algorithm which forms carefully chosen groups before noise injection.

We run these algorithms over 4 input histograms from the census data and measure the metrics of the noisy histogram. The upper row in Fig. 4.4 shows average distribution metric scores over all the input histograms and 10 random trials for each configuration. The lower row shows their perturbation accuracy for the corresponding task.

A good private algorithm should push the distribution towards the easy side of the spectrum as much as possible while at the same time preserving the underlying task answer. In other words, we want algorithms that produce outputs with good distribution metrics which lead to good perceptual accuracy. But we do not want the algorithm to be exaggerating the visual pattern too much and causing a perturbation failure. For example, for task Find Maximum, the Peakedness Score is high using the privacy parameter 0.001. While at the same time, the perturbation accuracy is low, showing that the noise shifted the distribution too much and created an easy-to-perceive but incorrect peak. Thus, in this case, MWEM is not a good algorithm choice.

4.3.3 Findings

For the three summary tasks considered, the MWEM algorithm tends to provide the lowest perturbation accuracy and only provides better perceptual metrics for one task: Find Maximum. For the task Find Anomalies, DAWA and Laplace have comparable perturbation accuracy, and DAWA tends to produce higher perceptual metrics with a lower privacy budget. So it is best to use DAWA as the DP mechanism for this task to gain good end-to-end visual utility, outperforming Laplace by a small margin. For task Find Maximum and Cluster, Laplace has noticeably higher perturbation accuracy than the other two algorithms, making it a wise choice. In real-world data exploration, it is not always clear what the downstream tasks are. Our findings show that despite its simplicity, the Laplace mechanism is a safe general choice for DP algorithms.

4.4 Discussion

In this section, we reflect on our findings and discuss strategies to better support visual data analysis under differential privacy.

The findings of our study show that the rate of perturbation accuracy loss (for the same level of noise) differs between summary and value tasks. In particular, summary tasks seem to be more tolerant of the injection of noise while value tasks are noticeably more sensitive. This finding has an important implication: it enables users to more efficiently manage their assigned privacy budget while analyzing data. For instance, based on the knowledge that the summary task Find Anomalies is highly resistant to high levels of noise, they can spend a smaller fraction of their budget on this task. Considering that the amount of privacy budget assigned to a user has a limit, efficient budget management is very important. Similarly, understanding the higher sensitivity of value tasks to the injection of noise and their higher rate perturbation accuracy loss can benefit users. In this case, such knowledge can inhibit performing queries that would result in futile outcomes. For example, consider the value task Determine Range, with a severe loss of accuracy under high-levels of noise, the user can decide to increase the privacy budget spent on the task to get more reliable answers or avoid the task altogether. In the setting of our study, performing a task like Filter, Find Maximum or Find Anomalies with privacy parameter 0.001 only introduces less than 10% additional error compared to using privacy parameter 0.01 but saves 90% of the privacy budget. However, for a value task such as Retrieve value, the accuracy drop can be as significant as 60%, and it is worth spending more privacy budget to get acceptable accuracy.

Although various differentially private algorithms have been designed to achieve higher query accuracy, it remains unclear which algorithms or noise injection mechanism will better facilitate downstream visual analysis. Newer and complex algorithms like [35, 49] apply specialized techniques to increase the accuracy of target workload queries. However, our initial investigation shows that these techniques can produce visual artifacts that make

the data "harder" when used in visual analysis tasks. For example, the MWEM algorithms can produce plateau-shaped distribution since it updates queries in groups identified by workload queries. Our findings show that, without a specific task, the simplest Laplace mechanism is the safe choice for private visualization.

Our work is the first in the confluence of differential privacy and visual data analysis that enables managing the privacy budget based on analysis tasks and visualization. This might be even more important for exploratory visual data analysis (EVDA). EVDA revolves around the continued formulation and evaluation of questions and hypotheses by the user. Many times, an analysis avenue does not result in any interesting insights and knowledge, and users move on to investigating other aspects of data. Under such conditions, the effective management of the privacy budget is even more critical. In practical data exploration, we do not always know the exact sequence of operations to perform ahead of time. So it can be hard to generate an optimal global allocation of privacy budget. However, with the knowledge about noise tolerance of tasks, we could go with a greedy approach trying to avoid spending too much of the privacy budget at each step in the iterative process.

CHAPTER 5

UNCERTAINTY IN STATIC DIFFERENTIALLY PRIVATE VISUALIZATION

We've investigated the influence of configurations in the private visualization pipeline in Chapter 4. However, the important aspect of uncertainty is missing. In this chapter, we discuss the uncertainty in static differentially private visualizations, using two-dimensional location data as an example. Then we propose a solution and evaluate it through an online user study.

5.1 Challenges

In this section, we present challenges of visualizing uncertain data together with some other main challenges faced in visualizing differentially private location data.

5.1.1 Visualization Under Uncertainty

Directly visualizing differentially private data only plot a single output of a randomized private algorithm. Although the effects of noise are visible, the uncertainty in the output is not presented to the user in a manner that can be properly interpreted. Appropriate visualization of uncertainty is a key challenge in visualizing differentially private data. This is similar to the challenge of visualizing *statistical* uncertainty, in which a practitioner is encouraged to not directly trust data (since there is uncertainty in statistical inference), or forecasts from a computational model like climate simulations [88] (since there is uncertainty in the model's accuracy).

Fig. 5.2 illustrates some of these challenges with the Beijing taxi data. Part (A) shows original data, plotted using the heat map approach described earlier, with cell color mapped

to the counts on a log scale. Dense road networks can be seen in the city, as well as some less-traveled roads in less dense areas, such as those that connect to the airport. The Laplace mechanism ($\epsilon = 0.1$) is used to obtain the noisy version (B) which would be given to a data analyst. Some of the high-frequency structures are preserved, but low-density regions are substantially changed in a random manner. For three selected cells, plot (C) shows the original true values (red triangles), versus noisy versions (blue dots). The fact that cell (1) has a higher frequency than (2) is preserved in the noisy data. But cells (2) and (3) have a *sign error*—their relative ordering is flipped in the data.

For 1d data, uncertainty can be summarized with error bars. Fig. 5.2(C) shows 95% intervals as vertical lines. These are constructed from a noisy data point $\hat{x}_i$ as $[\hat{x}_i + F^{-1}(0.025), \hat{x}_i + F^{-1}(0.975)]$, where F^{-1} is the inverse CDF of the Laplace (yielding intervals of approximately $\hat{x}_i \pm 30$ for this setting of ϵ); by construction, these intervals contain the true value 95% of the time. These error bars could be presented to a user, to be interpreted in a similar manner as confidence intervals from statistical inference; and helpfully, unlike the case of statistical inference where modeling assumptions may not hold, in this setting the confidence intervals are guaranteed to have correct coverage since the noise distribution is known.

But for 2D data, uncertainty visualization is less straightforward due to limitations on space and visual channels in a 2D setting (e.g. (A) or (B)). Researchers have explored methods to represent uncertainty on the same 2D figure with the data, such as summary plots [75], modifying the color to use hue or saturation to encode uncertainty [59], and showing uncertain data out of focus [58]. Alternatively, one can use interactivity. For example, in a linked-displays approach [12], a user could click to select one or a few cells from the (B) map, then be shown the cells' values in a second display (like (C)) with room to show error bars. These approaches deserve further consideration for visualizing private data.

Another approach to the faithful representation of uncertainty is to match the imprecision inherent to visual perception to the imprecision introduced by the privacy mechanism. The proposed principle is that *statistically indistinguishable counts should be visually indistinguishable*. More details will be discussed in Sec. 5.2

Overall, for visualizing uncertainty, we hope to benefit from the fact that the error in estimates is coming from a well-understood process (the privacy mechanism). Yet for some state-of-the-art algorithms, reliable error bounds are hard to establish because these algorithms adapt the noise distribution to the data. While it is possible to release a noisy measure of error, this adds an additional level of uncertainty that must be reconciled.

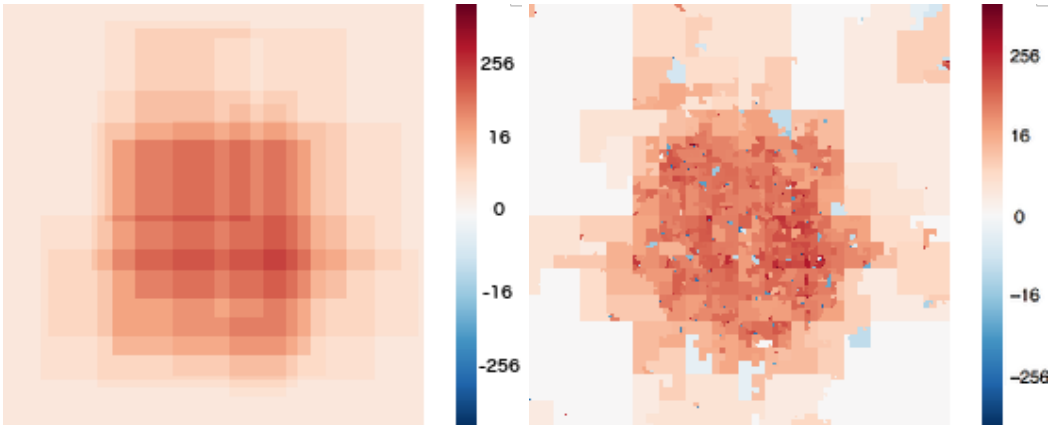
5.1.2 Visual Artifacts

The output of a differentially private algorithm may include visual artifacts which obscure true features or lead to false conclusions. For example, the noise introduced by the algorithm may result in negative counts for grid cells (which are clearly impossible) and can have a significant impact on a visualization if not corrected.

Negative counts can be easily corrected by rounding, but such adjustments sometimes have their own consequences. Simply rounding all negative values to zero boosts the overall sum across the grid cells leading to a biased output which may have its own visual impacts. More sophisticated ways to handle negative values have been proposed [54] and some mechanisms, like the Multiplicative Weights Exponential Mechanism [35], output non-negative counts directly. Issues such as non-negativity can sometimes be ignored when the private output is used to compute query answers, but are likely to become much more important in the context of visualization.

In addition to negative counts, there are other algorithm-specific artifacts that obscure the interpretation of the visualization. For example, the visualization in Fig. 5.1b, produced by the DAWA algorithm, includes large blocks of uniform regions, especially on the periphery of the figure where the density is lower. The algorithm intentionally estimates these

regions uniformly and avoids estimating sub-regions or cells internal to the region. This feature of the algorithm is quite effective in reducing numerical measures of error that are commonly used in the research literature but may mislead the viewer when the results are presented visually. This is especially true for the non-expert viewer unfamiliar with the algorithm's mechanics who may mistake algorithmic artifacts for structure in the data. This may call for re-thinking some of the advanced algorithmic techniques which are currently used to reduce error when measured by standard error metrics.



(a) Output of MWEM (b) Output of DAWA

Figure 5.1: Outputs with equal query-based error

5.1.3 Specifying and Achieving "Visual Utility"

The above discussion shows that effective data visualization is a utility goal that is potentially very different than the utility goals considered to-date in the literature on differential privacy. It is not clear how to make a notion of "visual utility" precise. Here we discuss potential ways to access the visual utility.

Similarity-based Utility The most straightforward way to measure the quality of the private visualization is to see how similar it is to its non-private version. Without any specific use case, we could go with the informal definition based on *perceived visual similarity to the true data*.

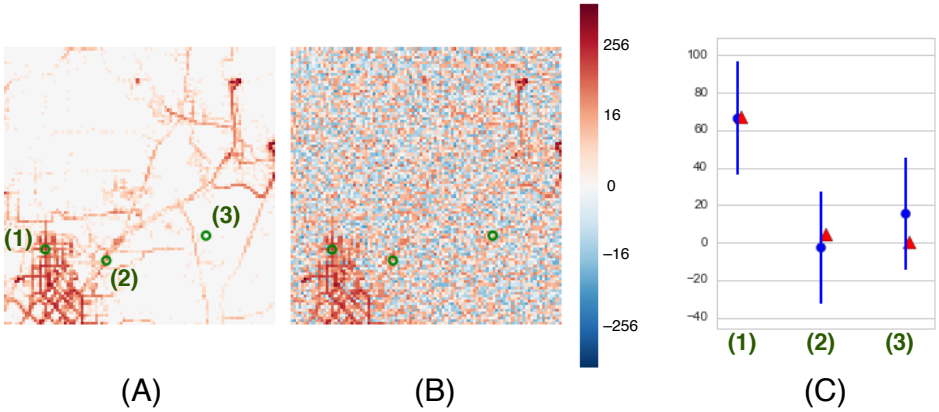


Figure 5.2: Illustration of uncertainty due to the Laplace mechanism, on taxi frequency data from northeast Beijing (Sec. 5.1.1). (A) Original data. (B) Noisy output, which preserves some structures but introduces spurious phenomena. (C) For three selected cells, original data values (red triangles), noisy versions (blue dots), and 95% confidence intervals (vertical lines). Cell (2) has a negative valued output, and the comparison between cells (2) and (3) has a sign error.

Task-based Utility Although visual utility could be generally defined through the similarity between the true and perceived data, a more specific task-based utility may be preferable when there's a well-defined visual task. The utility of visualization would then be determined by comparing the success rate for users to carry out the task on the true and private data.

There are two main obstacles to using the task-based utility. The first one is that the task needs to be known and well-defined. This is not often the case, especially with iterative data exploration. One potential way is to start from some common and basic visual tasks like density comparison, correlation estimation, and search for maximum and minimum values/regions. If some visualization techniques perform well on these common building blocks, we are more confident that they will perform well on some more complex tasks.

Another problem is that a large portion of the perceptual tasks can't be easily automated. This requires the participation of human users which can be expensive and time-consuming. Fortunately, the emergence of online crowd-sourcing platforms has made carrying out user studies a little bit easier.

Query-based Error

For our setting, there are many differentially private algorithms that can be applied to 2D data. Some algorithms produce a noisy histogram targeting a general class of queries (e.g. sums over all rectangles in the 2D domain) [26, 40, 78, 80] and some accept as input a user-specified workload of queries and tailor the output to accuracy for the workload queries [35, 50]. In either case, the error is commonly measured using metrics like L_1 or L_2 error on some set of queries of interest. We refer to this as *query-based error* and want to see if this widely used error measurement captures visual utility well.

With the similarity-based utility, the query-based error is not a reliable measure of visual utility. We show in the following example that two noisy outputs with the query-based equal error may have very different visual utility.

For plotting 2D data, we measure query-based error as the average, per-cell L_2 error of the 256×256 histogram output. This seems like the most natural metric because the user is seeing a colored representation of noisy values in each cell and we are comparing this representation to the true heat map. Fig. 5.1a and Fig. 5.1b are noisy outputs of two different algorithms named MWEM [35] and DAWA [50] with the same input data. We have used *different* epsilons for each algorithm in order to make the query-based error of the algorithms equal (here MWEM uses $\epsilon = 1$, while DAWA uses $\epsilon = 0.0065$). Clearly, these two figures have very different visual properties, demonstrating that similarity-based visual utility is not captured by query-based error.

However, with task-based utility, the answer is mixed. If the task could be decomposed and expressed in terms of a set of linear queries, then the goal of achieving lower query-based error matches well with the goal of achieving better success rate in the task. This is also the case where recent algorithm advancements will prove beneficial to visualization. With a proper input workload designed for the task, the effort of tailoring the output to accurate workload answers will pay off in terms of task-based visual utility.

The problem is that it is usually hard to identify a good workload for a lot of the complicated real-world tasks. The "optimal" set of linear queries might not even exist for some

tasks. Without proper input workload, the query-based error will fail to capture task-based visual utility.

5.2 An Attempt: Plotting under Uncertainty

In this section, we propose a principle to deal with the uncertain nature of differentially private outputs and a solutions to achieve the principle.

As discussed in Section 5.1.1, uncertainty information could be encoded in additional dimensions or channels in a visualization. However, this is not straightforward for 2D data, so we go with the other approach where we match uncertainty in data to the inherent imprecision in visual perception and propose the principle:
Indistinguishability Principle: *Statistically indistinguishable counts should be visually indistinguishable.*

5.2.1 Statistical Indistinguishability

In a 2D heat map of a noisy histogram, if we observe two cells C_1 and C_2 and find out that the noisy counts $y_1 > y_2$, there are two possible reasons: a) their underlying true counts $x_1 > x_2$, or b) the underlying data pairs have a different relationship, but the noise injected leads to the observation. The principle asks us to only visualize cells differently when we are confident that their observed difference comes from the differences in underlying data instead of from noise injected.

With this in mind, we define statistically indistinguishability using the probability of a "Type S error" [33] (or "sign error") which captures the relationship between observed estimations. There's a Type S error if the order of the two variables is flipped after noise injection.

Definition 10 (Statistical indistinguishability) *Let y_i and y_j be noisy measurements of two variables C_i and C_j, y_i and y_j are α-level statistical indistinguishability if*

$$P(sign(x_i - x_j) \neq sign(y_i - y_j)|y_i, y_j) \geq \alpha$$

where $sign(x) = \begin{cases} 0, & \textit{if } x > 0 \\ 1, & \textit{otherwise} \end{cases}$*, x_i and x_j are the true values of C_i and C_j*

A high probability of a Type S error implies that underlying data points are close together, this means the observed difference is more likely to be an effect of noise. So if the probability of a Type S error is higher than some threshold, the observations will be considered statistically similar, and should not be shown distinguishable from each other in the visualization.

For algorithms like the Laplace mechanism, in which independent noise is added to each count, statistical indistinguishability could be estimated by examining the difference of the observed noisy counts.

Claim: If the esitmations y_i and y_j are derived by adding i.i.d Laplace distributed noise to x_i and x_j. Then for a given α and Laplace distribution, there's a global constant D that,

$$\forall i, j \; P(sign(x_i - x_j) \neq sign(y_i - y_j)|y_i, y_j) \geq \alpha \implies |y_i - y_j| < D$$

Proof

We know $y_i = x_i + Lap_i, y_j = x_j + Lap_j$, where Lap_i and Lap_j are i.i.d Laplace vairables. Let the random variable $Z = |Lap_i - Lap_j|$.

$$P(sign(x_i - x_j) \neq sign(y_i - y_j)|y_i, y_j)$$

$$= P(sign(y_i - Lap_i - y_j + Lap_j) \neq sign(y_i - y_j)|y_i, y_j)$$

$$= P(|Lap_j - Lap_i| > |y_i - y_j| \wedge sign(y_i - y_j) \neq sign(Lap_j - Lap_i)|y_i, y_j)$$

$$= P(|Lap_i - Lap_j| > |y_i - y_j| \; |y_i, y_j)$$

$$*P(sign(y_i - y_j) \neq sign(Lap_i - Lap_j)|y_i, y_j)$$

$$= P(Z > |y_i - y_j| \; |y_i, y_j) * \frac{1}{2}$$

$$= \frac{1}{2}(1 - CDF_Z(|y_i - y_j|)) \geq \alpha$$

, then

$$|y_i - y_j| <= CDF_Z^{-1}(1 - 2\alpha)$$

so there's a constant $D = CDF_Z^{-1}(1 - 2\alpha)$ with a given α and Laplace distribution.

■

5.2.2 Achieving the Principle

As indicated in the principle, we want the statistically indistinguishable data to be visually indistinguishable. In the setting of a heat map, this means the difference between colors of the cells should be too small for a human to detect. In practice, this is done by providing different colormaps which map numeric counts of each cell to proper color. The colormap could be either continuous or discrete. For simplicity, we use a discrete colormap which consists of several bins. The range of the noisy counts will be divided into intervals of width D, counts within the same interval will be mapped to the same color. This ensures that counts within the same interval won't be distinguished from each other.

This mechanism is essentially a smoothing effort where we try to smooth out the difference which we believe results from noise. Given a specific DP algorithm, we have a well-understood noise model which helps us detect statistically distinguishable data.

5.3 User Study on Correlation Perception

As discussed in Sec. 5.1.3, without a good general-purpose visual utility measurement, we choose to use the task-based utility on common and basic tasks. In this section, we con-

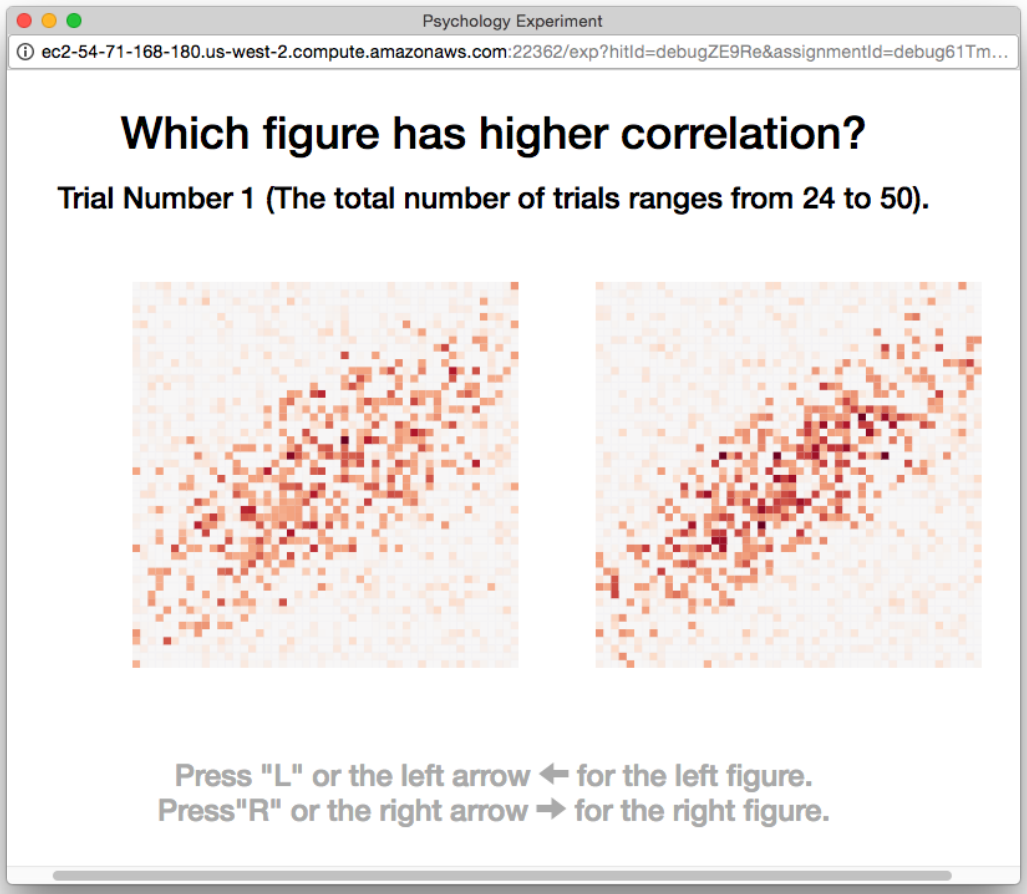
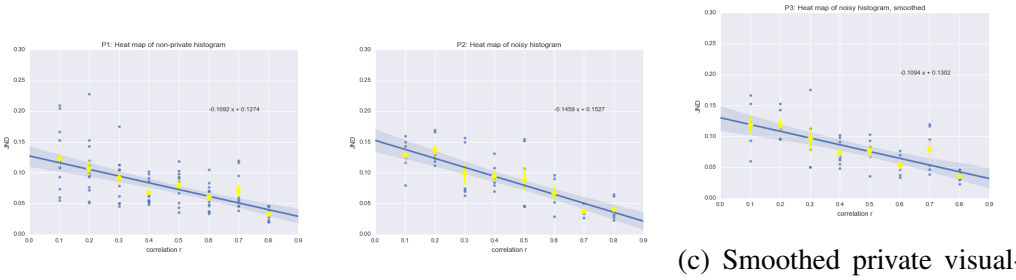


Figure 5.3: An example step of the user study, with the baseline private heatmap

duct multi-factor human-subject experiments to examine the effectiveness of our approach on some specific tasks using Amazon's crowdsourcing platform Mechanical Turk.

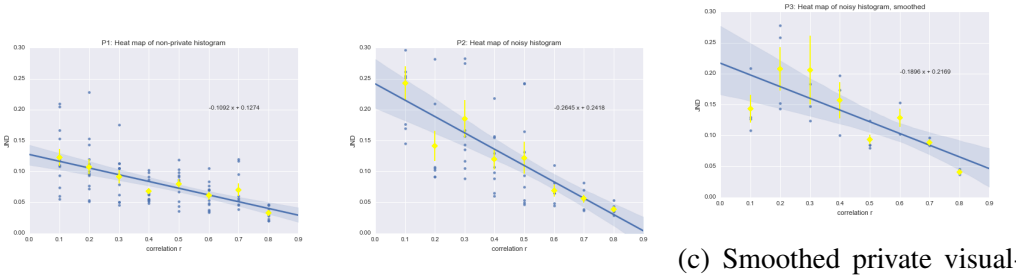
5.3.1 Task

The first thing to consider when measuring task-based utility is to determine a good task. However, real data exploration tasks are highly dependent on specific use cases and even datasets. So we go with the simpler tasks which we believe to be the common building blocks of more complicated tasks. Among the many basic visual tasks, we choose to start with correlation perception for several reasons: a) visualizing correlation is a very important and common task for 2D visualizations since it allows people to understand the



(a) Non-private visualizer (b) Naive private visualizer (c) Smoothed private visualizer

Figure 5.4: JND-r plots, moderate privacy level



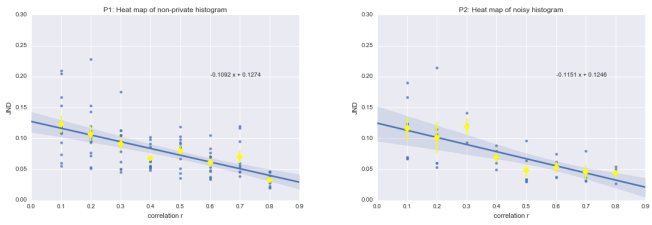
(a) Non-private visualizer (b) Naive private visualizer (c) Smoothed private visualizer

Figure 5.5: JND-r plots, high privacy level

relationship between variable pair; b) the task has been well-studied in previous works [82] [37] with justified experimental designs.

Rensink et al. demonstrated that the perception of positive correlation in scatterplots can be modeled using Weber's law [82], which indicates that the human perception of differences in correlation has a linear relationship.

To model a perceptual process using Weber's law, we need to experimentally determine how much a given stimulus must change before humans can notice with a reliable probability. This amount is called just noticeable difference (JND) [17]. It captures how well stimulus properties can be discriminated and has been widely used in psychology and vision sciences. If one method produces lower JND than the other in visualizing information, it provides better discrimination and helps people detect more subtle changes.



(a) Non-private visualizer (b) Naive private visualizer

Figure 5.6: JND-r plots, low privacy level

The 2D location data we've been using as an example is good for many interesting visual tasks but is too complicated for our correlation perception task. So from now on, we'll use synthetic data in the following experiments.

5.3.2 Experimental Settings

Although we don't continue to use the taxi pickup dataset as the example, we still follow a similar visualization pipeline. First, a dataset with a target correlation r is generated, then the data is binned to generate a histogram, and then the histogram is the input to the visualizers.

Three visualizers are compared in the experiment. The *non-private visualizer* takes the histogram and draws a heat map; the *naive private visualizer* adds independent Laplace noise to the histogram buckets and draws a heat map on the noisy histogram; the *smoothed private visualizer* adds noise in the same way but draws the heat map with the smoothing technique described in Sec. 5.2.2.

We follow the similar experimental design in [82] for each visualization technique in an online crowd-sourcing environment like Harrison et al. did in [37]. For each visualizer at a particular base correlation r, a user is asked to look at a sequence of side-by-side heat maps produced by the visualizer and decide which one has a higher correlation coefficient. Each step in the sequence is called a trial. An example trial in the experiment is shown in Fig. 5.3. We go with the "above" approach, the two plots are generated with correlation r and $r + d$. d is a dynamic distance value controlled by the system. If the user answers

the current trial correctly, the distance will be decreased by 0.01 for the next trial while keeping the base correlation r constant. On the other hand, if the user chooses incorrectly, the distance will be increased by 0.03, making the distinction more obvious. In each trial, the visualization with the base correlation and the variable correlation is randomized, so that the base correlation plot doesn't always appear on the left or on the right.

The staircase procedure stops when either the user's ability to distinguish correlation is believed to have stabilized or it reaches 50 trials. The judgment of stability of user's perception is determined through standard statistical tests. Three equal-sized subgroups are formed in the moving window of the last 24 trials. Stability is reached when there's no significant difference between these three subgroups via an F-test ($F(2, 21); \alpha = 0.1$). After the procedure ends, the average d in these subgroups is used as the JND for the tested base correlation r.

With the linear relationship $JND(r) = k(1/b - r)$, the JND at any correlation could be estimated from a small set of (r, JND) pairs. So in order to compare two visualizations, we sample 8 base correlation values $(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8)$ in the range $[0, 1]$, get the corresponding JND values.

For each visualizer and base correlation, we recruited 10 participants. Each participant will answer a sequence of left/right choices with at least 24 and at most 50 trials. In order to make sure the participants understand the task, after a general instruction, 5 example trials with feedback are provided for each participant. The participant will know if he/she answered the trial correctly and why, and the underlying correlation is also shown in the example trials. Each participant was paid $\$0.3$ for a sequence of answers.

Even with the instruction and example trials, we still got unqualified results. So we did simple post-processing based on some simple heuristics to filter out responses of extremely low quality. The following behaviors are considered "suspicious": a) responding to all trials with the same answer; b) having extremely short reaction time; c) failing to answer very obvious correlation distinctions (e.g. $r = 0.5$ v.s. $r = 1$). These behaviors

either indicate that the participant failed to understand the concept of correlation or the participant made their choice based on random or fixed answers no matter what they saw. The problem of classifying responses according to quality could be an interesting machine learning problem itself. However, it is beyond the scope of this work. So we carried out the post-processing on manually tuned thresholds for these suspicious behaviors.

With the processed data, we were able to fit a linear $JND - r$ relationship for each visualizer. In terms of performance comparison, for both k and b, smaller values denote better performance, with optimal performance as these values approach zero.

5.3.3 Results and Findings

The resulting JND-r plots are shown in Fig. 5.4. The x-axis is the true correlation in the non-private data. Each blue dot indicates a JND value calculated from some participant on a particular base correlation. The blue lines show the linear relationship, with 95% confidence interval of the regression. The yellow diamonds shows the mean JND at each base correlation with error bars reporting the standard error.

As shown in Fig. 5.4a and Fig. 5.4b, the private visualizer has higher JND values, especially in the low correlation region. People's ability to distinguish correlation in true data is harmed by the noise. This is intuitive since adding random noise in histogram counts equals adding random samples which tends to make the data less correlated, thus raises the JND. In conclusion, the naive private visualizer performs worse than the non-private visualizer.

Now compare Fig. 5.4c with the other two plots. The smoothed private visualizer performs better than the naive private visualizer and similar to the non-private visualizer for the privacy level tested. This shows our effort to smooth out the unnecessary distinctions in visualization helps people better distinguish the underlying correlation information. This is because the smoothing "restores" the perturbed correlation back to the true correlation to some extent by removing low confident data distinctions.

Another interesting finding is that people's perception of correlation is more stable in the high correlation region than in the low correlation region. For example, when the correlation is as high as 0.8, the JND is nearly the same for the three visualizers for this privacy level. The reason could be human's ability to "denoise". With independent random noise, the injected noise itself acts as a low correlation background, and the high correlation true data stands out for human eyes. This justifies our method in that we are doing similar things as human eyes, but doing the smoothing in a more educated way, taking advantage of the statistical information of the noise.

Privacy level

As we can see from the previous discussion, the noise injected will introduce perturbation and thus make the data less correlated. Visualizing the perturbed data directly could lead to a bad estimation of the underlying correlation, and our smoothing technique could alleviate the perceptual deviation. In the following discussion, we'll look at an important factor which could have a huge impact on the performance of the private visualizers, the privacy level.

In differential privacy, the level of privacy protection is controlled by the parameter ϵ. A smaller ϵ implies stricter privacy or higher privacy level and a larger ϵ provides a guarantee in a lower privacy level. Although algorithms differ in their ways to inject noise, generally more noise is needed to achieve a higher privacy level.

Researchers in the differential privacy community have been aware of the trade-off between privacy and utility, especially in terms of statistics estimated in a privacy-preserving manner. In order to better understand the trade-off between visual utility and privacy for our correlation task, we conduct additional user studies in different privacy levels. We manually tune the privacy parameter to find a range of "reasonable ϵ's": the minimum ϵ is the largest value which leads to a naive private visualization that makes sense as a correlation plot, and the maximum ϵ is the smallest value which leads to nearly unnoticeable noise in the naive private visualization.

Fig. 5.5 shows results at a high privacy level. The JND increment from Fig. 5.5a to Fig. 5.5b, especially in the low correlation area, shows that noise at this level will result in bad correlation perception if we use the naive private visualizer. Our smoothed private visualizer (Fig. 5.5c) helps to bring the JND down a little bit in the low correlation area, but much higher than that in non-private version. This means with this much noise added for the required high privacy level, the smoothing technique can still help restore part of the true correlation but not to a comparable level with the non-private visualizer.

Fig. 5.6 shows the other end of the privacy spectrum. As shown in the comparison of Fig. 5.6a and Fig. 5.6b, when the privacy level is low enough, the noise injected barely influences the correlation perception for the naive private visualizers. The smoothing technique is not even necessary here, so experiment for the smoothed private visualizer is omitted.

In conclusion, our smoothing technique works best with a moderate privacy level where the smoothed visualizer could restore most of the correlation loss caused by noise injection while the naive visualizer fails to do so. When the privacy level is too high or too low, the smoothing is unnecessary or won't help much. However, a very high or low privacy level is rarely wanted in practice since it is too strict that basically no algorithm could provide reasonable utility or too loose to be satisfiable protection.

5.4 Related work

Without privacy requirements, visualizing under uncertainty has been studied for years. One common solution is trying to present the present uncertainty explicitly. The uncertain statistic can be represented as an overlaying layer encoded in the formats of error bars or summary plots[75], [87] investigated the effectiveness of different glyphs and markers in conveying uncertainty. Another way is to encode uncertainty with some dimensions in the color space. [59] used hue and saturation to indicate uncertain level, [42] proposed mixing white pixels to represent high uncertainty. [58] added a layer with opacity proportional to uncertainty, so the uncertain areas would be shown out of focus. Even though with dif-

ferent encoding strategies, a common first step is to identify proper statistics like standard deviation to measure the uncertain level. And in order to read the visualization, a basic understanding of the statistic is needed.

Despite the absence of a good definition of visual utility, tasks in general data exploration have been studied. [6] described a task topology for the exploratory analysis of spatial and temporal data based on Jacques Bertin's ideas. Tasks are categorized as elementary or synoptic according to the level of data analysis. Elementary tasks include lookup and comparison of individual elements while synoptic tasks try to find a pattern for the fact and relationship in data considered as a whole. The correlation perception task in our user study can be considered as a synoptic task trying to find the correlating pattern through visualization.

Our user study is designed and carried out based on previous works. [82] was the first work to apply classic psychophysical methods originally developed for simple properties to the evaluation of the visual perception of correlation in scatterplots. Rensink et al. not only demonstrated correlation perception in scatterplots follows Weber's law, but also provided a rigorous way to evaluate complex visualizations. Based on their work, [37] replicated the original experiments on an online crowd-sourcing platform and extended the investigation from scatterplots to various visualization techniques for the correlation perception task. Their work showed that perceptual laws could be leveraged to rank different visualizers for the same task. We built our user study on top of their experimental designs and extended the comparison to the heat map of histograms and private-preserving visualizers.

In terms of privacy, we have been considering the private-preserving data publication. Here the trusted data curator has access to the sensitive dataset, executes the private visualizer offline and only publishes the final private visualization. This is only one of the many privacy models. Some other models include interactive model [25][83] where the curator provides a private API which can be queried multiple times, local privacy model [44][23]

where the curator is not trusted and privacy of the individual is protected by randomized responses at the data collection step.

5.4.1 Communicating Uncertainty under DP

Differentially private data, like data generated from many other privacy-preserving techniques, is inherently uncertain. DP mechanisms achieve the privacy guarantee by adding noise to computations on the sensitive data, making any single output a random instance of some underlying distribution. Therefore, dealing with uncertainty is an essential aspect of differentially private data visualization.

There has been a rich line of work on visualizing under uncertainty. One common solution is to present the level of uncertainty explicitly. The uncertainty statistic can be represented as an overlaying layer encoded in the formats of error bars or summary plots [75]. Sanyal et al. [87] investigated the effectiveness of different glyphs and markers in conveying uncertainty. However, error bars can be hard to read in multi-dimensional visualizations. An alternative is to color-code uncertainty information. Maceachren et al. [59] used hue and saturation to indicate uncertainty levels, and Hengl et al. [42] proposed mixing white pixels to represent high uncertainty. A critical first step towards communicating uncertainty is selecting metrics (e.g., standard deviation) to quantify uncertainty. This can be challenging for differentially private output. For simple algorithms like the Laplace mechanism [26], we could easily derive the scale and variability of the noise from the privacy parameters. However, the computation of such uncertainty for complex algorithms can be hard. For example, the DAWA[49] algorithm first applies grouping based on the characteristics of the input data distribution to achieve better accuracy. Thus each data group has different uncertainty levels and carefully designed approaches are needed to calculate these local uncertainty statistics privately.

CHAPTER 6

DASHGUARD: INTERACTIVE PRIVATE DASHBOARD

In the previous chapter, we have discussed the application of differential privacy for static visualizations. However, with static visualizations, users can't go beyond the predefined view to get additional stories or further explore the data. To tackle these limitations, interactive visual dashboards have been widely used in exploratory data analysis and complex data storytelling. Dashboards visually organize, analyze, and display important statistics about the underlying data.

Such applications introduce risks of harm to individuals due to possible breaches of personal privacy and leakage of sensitive information. Prior research in the confluence of privacy and visual data analysis has been mainly focused on the use of syntactic privacy models such as k-anonymity and l-diversity (e.g.,[96, 15, 14]) despite the rising popularity of DP. We still know little about the effective use of DP in the context of visual data analysis, especially in an interactive data exploration setting. In this chapter, we investigate the challenges of privacy-preserving interactions with sensitive data using visual dashboards and propose a solution to those challenges.

6.1 Challenges

A typical interactive dashboard usually includes several small panels showing views of the underlying data using visualization techniques like scatterplots, linecharts, or geographical distributions over maps. The corresponding views and visualization types are usually customizable by the user to meet special data exploration needs. Then the user can interact with the data by zooming in and out interesting regions or drill down by filtering with

certain conditions. Besides interacting with a single visualization view, they can also *link* multiple panels such that zooming and filtering interactions on one view will be reflected on all linked panels. This powerful functionality allows users to compare and explore relationships between different data aspects. In this work, we limit the set of interactions to independent browsing, brushing, and linking (linked filtering) of univariate histograms.

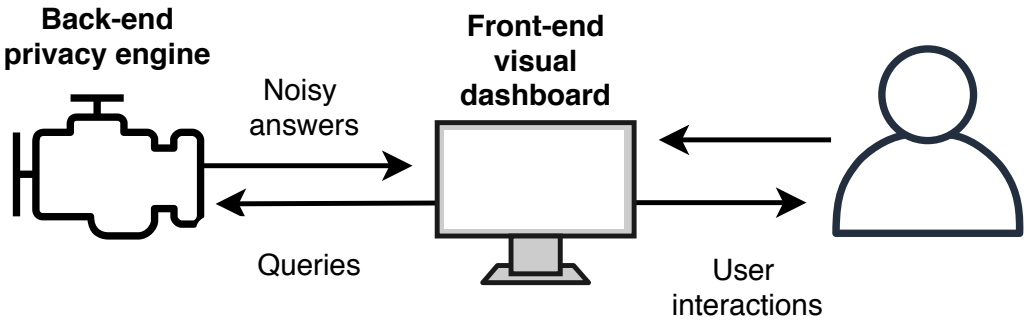


Figure 6.1: Naive private interactive dashboard where every request from the front end will be queried on the sensitive data directly

To perform the exploration in a private-preserving manner, a naive model is described in Fig. 6.1. A user interacts with the front-end visualization dashboard where every interaction request is converted to a query on the underlying data. To protect privacy, every knowledge learned about sensitive data needs to be conducted in a privacy-preserving way. We use Ektelo [106] as the back-end private engine. In the naive model, the privacy engine directly handles all interactions with the data and manages the privacy budget. However, this simple model raises certain challenges in practical deployments.

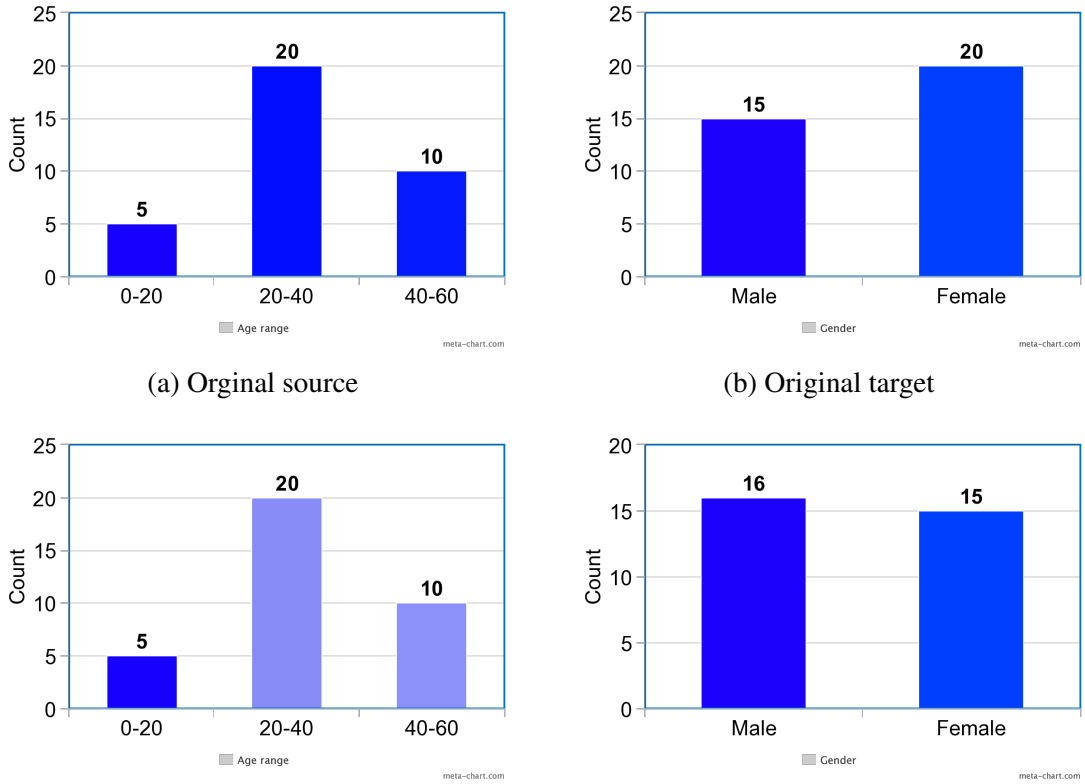
Challenge 1: potential waste of privacy budget Data explorations usually begin with a vague goal and the users decide directions to drill down based on findings in earlier steps. The process often includes constant zooming in and out, trying out different filtering conditions. Thus, queries in the exploratory sequence can be correlated or even duplicated. Directly issuing all those queries in the back-end privacy engines leads to duplicate observations of the same part of the data. As noted earlier, any observation of sensitive data

consumes part of the privacy budget. Thus exploring data using the naive model could lead to a serious waste of privacy budget. When the privacy budget runs out, the user can no longer make any additional observations of the data source.

Since differentially privacy is safe under post-processing, any previous result generated consuming privacy budget can be reused without paying additional budget. Therefore, there are chances that the answer to a certain query could be constructed from earlier results (or even from reusing previous results). As an example, after doing independent browsing of distributions over two attributes *Age Group* and *Gender*, any brushing and linking between those two attributes can be re-constructed using the histogram on the cross-domain of age group and gender.

Challenge 2: inconsistent brushing and linking Differentially private algorithms perturb the underlying data and introduce well-tailored noise to achieve the privacy guarantee. The noise may cause inconsistency within interactions which could confuse the users. When a user performs brushing and linking, the noise could break the inequality constraint between the original data and the filtered data. When a user performs brushing on the source visualization, the linked target should change to the filtered view based on the brushing attribute. Thus, the new target visualization should only contain a subset of data points, leading to strictly lower counts in a histogram. However, with independent noise injected at each step of the interaction sequence, there are chances that the filtered results get higher noisy counts than the original data. We show a simplified example in Fig. 6.2. The left column shows the source visualization on age group before and after brushing and the right column shows the linked attribute gender. In the brushing operation, the right two bars are highlighted so the linked distribution should be gender distribution within the age range 20-60. Using the naive model, a new query is issued on the linked attribute age adding independent noise. In an unlucky case, as shown in the figure, two major inconsistencies occur. Both the sum of male and female and the count of male is larger than the number

in the unfiltered data. Such inconsistencies convey confusing information and may greatly impact the flow of interaction.



(a) Orginal source

(b) Original target

(c) Brushed source

(d) Linked target

Figure 6.2: Example of inconsistent brushing and linking. When brushed on the source attribute (selected bars are shown in a lighter shade), the linked distributed changed and causing inconsistencies: 1) the sum of male and female becomes larger, 2) the filtered count of male is larger than the original count.

Challenge 3: slow response time Computing every query on-the-fly as shown in Fig. 6.1 can be slow and might break the interactivity of the user's workflow, especially when using the more complex and time-consuming mechanisms. Precomputation is a natural solution. However, with the need for privacy protection, any information retrieved from a sensitive data source needs to be properly managed and should consume some privacy budget. Due to the exploratory nature of these interactions, it is very hard to predict the sequence of queries in advance. A naive approach is to generate a synthetic version of the full joint

distribution over all the attributes. However, this is usually not practical since the size of the full joint grows exponentially with respect to the domain size and may go beyond the limit of machines' physical memory.

To address these challenges with interactive private visualization, we propose the framework DashGuard utilizing a smart middle layer between the front-end interface and back-end privacy engine with the following benefits:

1. The framework can automatically "correct" some suboptimal sequences of queries generated by novice users through pruning unnecessary queries and reuse of measurements. Such caching and reusing of measurements improves query accuracy and performance substantially and avoids waste of privacy budget.

2. The inference component removes inconsistency in user interaction which may confuse the user and interrupt the interactive workflow.

3. Through lazy precompute, sufficient statistics are prepared only when necessary. This avoids the heavy time and memory requirement of the full precompute approach and provides faster interaction than the pure online approach where everything is computed on-the-fly.

4. Flexible middle layer that can be plugged in between different user frontend and backend privacy engines with the required interface to support fast and accurate private visual exploration.

6.2 DashGuard

To address the three challenges of the naive model, we propose DashGuard, a private interactive visual dashboard with a smart middle layer that acts as a query proxy of any front-end interactions and back-end queries.

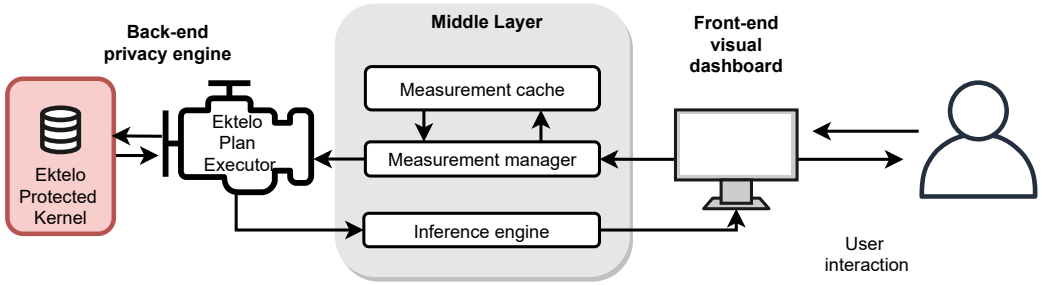


Figure 6.3: Overall structure of DashGuard. The middle layer acts as a proxy of any query to the back-end engine and automatically decides on which measurements to take based on the input query.

6.2.1 Visual Front-end and Interactions

The DashGuard front-end is the visual interface that interacts with end-users. It converts any user's action to queries to the back-end, collects the returned result, and then renders the changes in the visualizations. More specifically, the user could i) create a visualization of a univariate histogram over a certain attribute by adding a new plot to the dashboard, or ii) link two existing visualizations by adding a connecting component of corresponding plots in the dashboard, or iii) perform brush and linking. When a user brushes on one plot, the underlying attribute got filtered and all linked plot gets updated.

We choose to focus on visualizations of univariate histograms since they are the foundation of many more complex visualizations and their static differentially private deployment has been investigated in Chapter 4 (also [107]). And as a popular interaction, brushing and linking connects different views of the underlying data and can be used to discover relationships between data dimensions and attributes. All three types of interactions supported can be well represented as a linear range query over the domain of the underlying data. The overall user interaction trace is the sequence of interaction at each step and is restricted by the total amount of privacy loss budget ϵ granted by the data owner ahead of time.

6.2.2 Backend Privacy Engine

In this work, we choose Ektelo[106] as our backend privacy engine since it provides an executing environment with privacy guarantee and support for achieving better accu-

racy. The back-end engine takes in linear range queries. To answer the queries, the engine makes noisy measurements of the underlying sensitive data using differentially private mechanisms. When there are multiple measurements, inference methods could be used to combine them into a consistent estimate of the sensitive data and answer the incoming query.

6.2.3 Smart Middle Layer

Motivated by the fact that some queries in an interaction sequence can benefit from reusing partial or full information from previous results, the middle layer caches all previous interaction results and only issues new back-end queries when necessary. Before returning the results, it performs inference to resolve potential inconsistencies.

The middle layer has three major components:

Measurement Manager The measurement generates takes direct requests from the front-end, checks the cache, and decides on the back-end query to issue (if necessary). It has three main functionalities:

1. Retrieve previous measurements. If an incoming query has been measured in earlier steps, the manager will not issue a new request to the back-end. It retrieves the value from the measurement cache, returns to the front-end interface, and saves the corresponding privacy budget.

2. Combine previous measurements. If an incoming query hasn't been directly measured before but can be constructed from noisy estimates in the measurement cache, then no new request will be issued to the back-end engine. The measurement manager will retrieve the previous estimate, feed it into the proper inference engine, and then return the query result to the front-end.

3. Issue measurement request to the back-end. The manager only issues a new request to the back-end when necessary.

4. Lazy precompute. To make the interaction fast, each time a new attribute is introduced into the panel, the generator issues back-end query calculating sufficient statistics of any potential interaction involving the new attributes.

5. Manage privacy budget. The manager keeps track of the actual privacy budget consumption.

Measurement cache It keeps track of all the queries issued by the measurement generator together with the corresponding privacy budget and their noisy answer.

Inference engine Noisy results returned from the back-end privacy engine or the measurement cache can have inconsistencies. The inference engine combines those results to generate an "best guess" that meets the consistency constraints and returns it to the from-end for visualization.

6.2.4 Example Use Case

Next, we further illustrate the functionalities of DashGuard by describing a concrete use case

Alice owns census data collected from the whole county. Bob is an analyst who wants to explore the data with a vague goal of understanding the relationship between participants' age, income, and gender. Bob first generates bar chart to visualize histogram over attribute "age", "income" and "gender". DashGuard checks the cached measurements and found no match, so the middle layer issues the queries to the back-end and returns the result to the front-end when finished. After seeing the overall distribution, Bob decides to further explore the relations, so he links "income" with both "age" and "gender". DashGuard would now precompute all sufficient statistics for any possible brushing and linking operations. In particular, it would be the joint distribution over (income, age) and (income, gender). Then Bob brushes on gender to select only the female participants, the linked income distribution changes accordingly. DashGuard makes sure the filtered income frequencies will be

smaller than its full marginals values. Bob does the same brushing for age and has a better understanding of the impacting factors of income in the area.

6.3 Evaluation

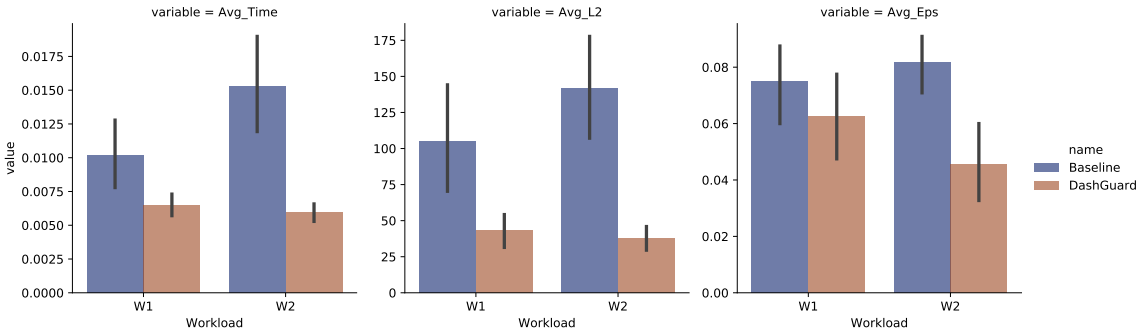


Figure 6.4: Overall comparison of DashGuard vs. the naive baseline shows that using the DashGuard (with bounded inference engine) provides better accuracy, faster responses with lower privacy budget consumption.

Next, we show some initial experimental evaluation results demonstrating the effectiveness of DashGuard by comparing it to the naive approach from the perspective of accuracy, privacy budget consumption, and response time.

6.3.1 Dataset and Workload

IPUMS-CPS[89] is a collection of datasets that harmonizes microdata from the monthly U.S. labor force survey and the Census Current Population Survey (CPS), covering the period 1962 to the present. We used a data extract which is a subset of the Annual Social and Economic (ASEC) data from 2010. We chose this population survey data for two main reasons.

The data extract contains personal survey information on 159,277 individuals, each contributing one row to the dataset. For our experiment, we selected a subset of numerical and categorical attributes: Age, Sex, Race, Marital status, Education status, and Total income.

Step	W1	W2
0	Plot(Income)	Plot(Income)
1	Plot(Sex)	Plot(Sex)
2	Link(Income, Sex)	Link(Income, Sex)
3	Brush(Income, (0,30))	Brush(Income, (0,30))
4	Plot(Race)	Plot(Race)
5	Link(Income, Race)	Link(Income, Race)
6	Brush(Income, (0,30))	Brush(Income, (0,10))
7	Brush(Income, (0,40))	Brush(Income, (0,30))
8		Brush(Income, (0,60))
9		Brush(Income, (0,80))
10		Brush(Income, (0,90))

Table 6.1: Detail of example workload. Three types of actions (Plot, Link and Brush) are supported.

Selected data were organized in a tabular format where each row represented information about an individual.

The workload refers to the sequence of interactions users perform in their exploration process. To evaluate our framework without a full-fledged user study, we adopt the generating process of IDEBench[28]. We use a synthetic query sequence generated using a Markov chain from a pre-defined set of queries and interactions). DashGuard interface supports three types of action from the front-end. Plot generates a visualization for a univariate histogram on the corresponding variable and adds a panel to the dashboard. Interactions stars with users generating plots and populating the dashboard. Then when there are more than two visualizations available, the user could choose to link any two of them. Linked visualizations are connected and their values may change together in future interactions. The third action brush where users could highlight some part of a view and any connected views will be filtered to only show the highlighted data points. In the workload formatting, the range stands for the highlighted bins in the histogram. Table 6.1 shows the steps of two example workloads.

In the experimental evaluation, we execute the example workload using both the naive approach and DashGuard. For each interaction step, we measure the L2 error between the

returned noisy query results and the query result on the true data and record the response time which is the time needed to fulfill the front-end query. Since differentially private algorithms are randomized, we take the average over 5 random trials and report the average for each experimental configuration. For every single step of interaction, the system is allocated a privacy budget $\epsilon = 0.1$. The naive model always consumes the budget to get a new measurement, however, DashGuard may decide to rebuild the result from its measurement cache and save the budget for later.

Figure 6.4 reports the average response time, L2 error, and budget consumption average over steps for the different workload. Here for DashGuard, we use the bounded inference engine. As shown in the figure, DashGuard outperforms the naive baseline approach in terms of accuracy and efficiency on both workloads. And since it avoids unnecessary ϵ consumption, the accumulated privacy loss budget usage is also lower than the baseline.

6.3.2 Per-step Breakdown

To better understand how DashGuard provides the benefits, we show a broken-down comparison of each metric in Figure 6.5. First, it shows how duplicate queries like step 3 and step 6 in W1 (or step 3 and 7 in W2) can be answered with cached results in DashGuard while still recomputed in the naive model. Linking operations like steps 3 and 5 in both workloads are fast and do not consume any budget in the baseline model since no actual computation is happening. At the link step, DashGuard performs lazy precompute and prepares sufficient measurement for any future queries. Although this causes additional time and budget consumption at the linking step, the overall average is greatly improved especially for an interaction-heavy workload.

6.3.3 Comparing Inference Engines

Figure 6.6 shows the accuracy and efficiency trade-off between variations of DashGuard using different inference engines. Models with prefix DG stands for DashGuard variations and the postfix indicates the inference engine used. As demonstrated earlier, all variants

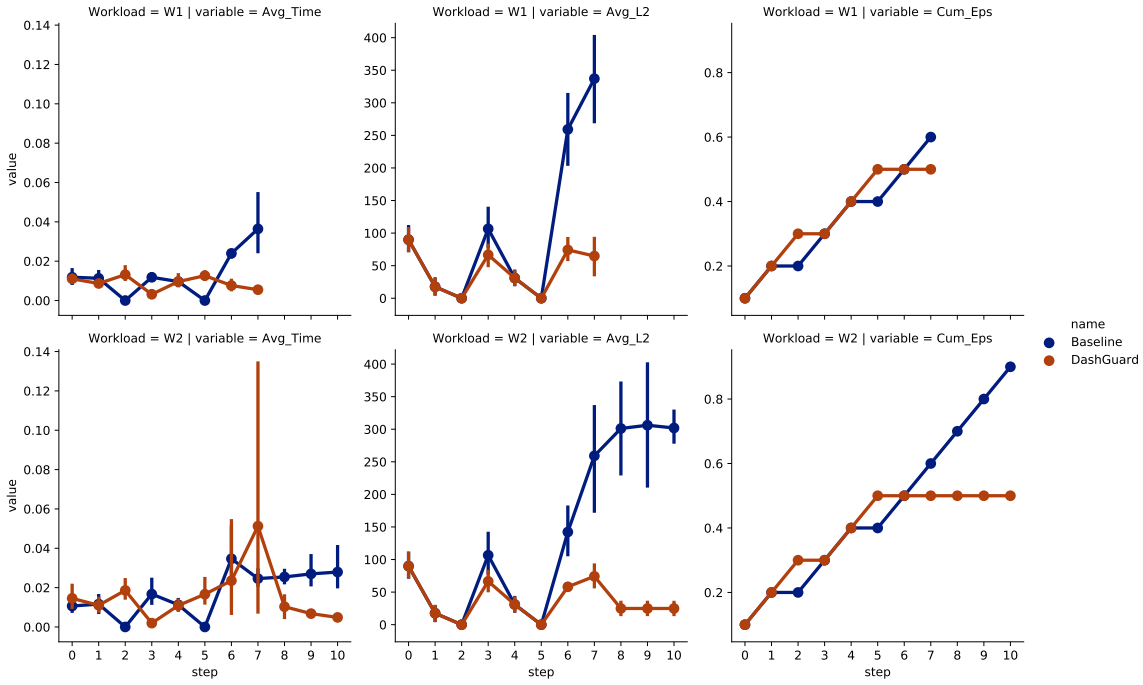
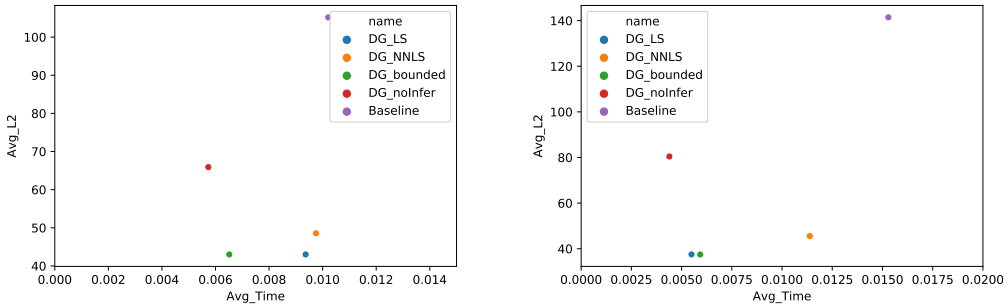


Figure 6.5: Per-step accuracy, response time and cumulative budget consumption comparison.

are in a better position in the accuracy-time space compared to the baseline. DashGuard without any inference provides the fastest response time but suboptimal accuracy. Using the bounded optimization inference engine, DashGuard achieves the best accuracy with slight runtime overhead for both workloads. The standard least-squares engine provides comparable accuracy and slower runtime compared to the bounded optimization engine. When enforcing the non-negative constraint, both accuracy and performance decreased compared to LS. Overall, we choose to use the bounded optimization engine as the default inference engine but leave the flexibility to change.

6.4 Related Work

There has been a whole line of research on general visual dashboards. However, much less attention has been allocated to making the dashboard private. Overlook [92] is a synopsis-based visual dashboard, meaning a one-shot private data summary (i.e., the synopsis) is generated before later interactions. With synopsis-based approaches, budget



(a) for workload 1

(b) for workload 2

Figure 6.6: Accuracy and efficiency trade-off between variations of DashGuard (marked with prefix DG) for two example workloads.

management is easier since any queries issued in the interactions perform on the synopsis generated with pre-defined ϵ. However, usually no single version of the data summary works for all types of interactions. Without a good assumption about the later interactive queries, it is hard to guarantee the quality of those queries. The construction of such synopsis usually requires solving complex and expensive optimizations which can get very time-consuming. Besides, Overlook is only considering interactions within plots but not across plots like brushing and linking. PSI [32] has a visual interface to communicate the impact of noise injection using different privacy budgets ϵ. It is designed for data sharing rather than interactive data exploratory. VISDPT [41] provides an interface to privately view two-dimensional user trajectories in a static manner.

6.5 Conclusion

We describe the design and implementation of DashGuard, a private interactive visual analytic dashboard that makes use of a smart middle layer between the frontend visual interface and backend privacy engine. The middle layer keeps track of all measurements taken and automatically decides on good measurements to issue to the back-end engine, conduct lazy precompute when necessary. After collecting measurements, it conducts inference to resolve inconsistencies. We demonstrate the effectiveness of DashGuard through

empirical comparisons on the census data for accuracy, budget consumption, and response time.

CHAPTER 7

CONCLUSION

In this thesis, we work towards making practical differentially private mechanism design and deployment easier. We address two main questions: 1) how can we aid programmers in developing private programs with high utility? and 2) how can we deploy differentially private algorithms to both static and interactive visual analysis systems which are widely used across sensitive domains such as health care and civic decision making?

We introduce ϵKTELO, a programming framework, and system which can be used to author programs for a variety of statistical tasks that involve answering counting queries. In ϵKTELO, programs are described as compositions of reusable modules and automatically satisfy differential privacy. We demonstrate the use of ϵKTELO by designing new algorithms offering state-of-the-art accuracy and runtime.

Then we move on to deploying the mechanisms into real-life applications. Specifically. we focus on visual analytics systems and take deep looks at both static and interactive visualization. We start by investigating the challenges of deploying differentially private algorithms in visualization tasks. Specifically, we conduct a study to better understand the relationship between noise introduced for privacy protection, visual analysis tasks, visualization, and accuracy in static visualizations. We also look at the challenges of dealing with the inherent uncertainty in private visualizations and propose a solution where only differences that could be confidently distinguished will be visualized.

Next, with interactive visual analytics, we show how the direct deployment of differentially private algorithms causes both efficiency and accuracy issues and propose a solution using a middle layer proxy that delegates and processes any front-end queries. Then we

evaluate it against the naive model of the private interactive dashboard, demonstrating improvements in terms of both efficiency and accuracy and privacy budget consumption.

BIBLIOGRAPHY

[1] 2018 differential privacy synthetic data challenge.

[2] Onthemap. `https://onthemap.ces.census.gov/`, 2010.

[3] Ács, Gergely, Castelluccia, Claude, and Chen, Rui. Differentially private histogram publishing through lossy compression. In *ICDM* (2012), pp. 1–10.

[4] Albarghouthi, Aws, and Hsu, Justin. Synthesizing coupling proofs of differential privacy. *Proc. ACM Program. Lang.*, POPL (Dec. 2017).

[5] Amar, Robert, Eagan, James, and Stasko, John. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* (2005), IEEE, pp. 111–117.

[6] Andrienko, Natalia, and Andrienko, Gennady. *Exploratory analysis of spatial and temporal data: a systematic approach*. Springer Science & Business Media, 2006.

[7] Archambault, Daniel, and Hurley, Neil. Visualization of trends in subscriber attributes of communities on mobile telecommunications networks. *Social Network Analysis and Mining 4*, 1 (2014), 205.

[8] Barak, Boaz, Chaudhuri, Kamalika, Dwork, Cynthia, Kale, Satyen, McSherry, Frank, and Talwar, Kunal. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *PODS* (2007), pp. 273–282.

[9] Barthe, Gilles, Farina, Gian Pietro, Gaboardi, Marco, Arias, Emilio Jesus Gallego, Gordon, Andy, Hsu, Justin, and Strub, Pierre-Yves. Differentially private bayesian programming. In *CCS* (2016), pp. 68–79.

[10] Bhattacharjee, Kaustav, Chen, Min, and Dasgupta, Aritra. Privacy-preserving data visualization: Reflections on the state of the art and research opportunities. *STAR 39*, 3 (2020).

[11] Brickell, Justin, and Shmatikov, Vitaly. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), ACM, pp. 70–78.

[12] Buja, Andreas, Cook, Dianne, and Swayne, Deborah F. Interactive high-dimensional data visualization. *Journal of computational and graphical statistics 5*, 1 (1996), 78–99.

[13] Byrd, Richard H, Lu, Peihuang, Nocedal, Jorge, and Zhu, Ciyou. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing 16*, 5 (1995), 1190–1208.

[14] Chou, Jia-Kai, Bryan, Chris, and Ma, Kwan-Liu. Privacy preserving visualization for social network data with ontology information. In *2017 IEEE Pacific Visualization Symposium (PacificVis)* (2017), IEEE, pp. 11–20.

[15] Chou, Jia-Kai, Wang, Yang, and Ma, Kwan-Liu. Privacy preserving visualization: A study on event sequence data. In *Computer Graphics Forum* (2018), Wiley Online Library.

[16] Comaniciu, Dorin, and Meer, Peter. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence 24*, 5 (2002), 603–619.

[17] Coren, S, Ward, LM, and Enns, JT. Sensation and perception. 1999. *Hartcourt Brace, Forth Worth*.

[18] Cormode, Graham. Personal privacy vs population privacy: Learning to attack anonymization. In *KDD* (2011).

[19] Cormode, Graham, Procopiuc, Magda, Shen, Entong, Srivastava, Divesh, and Yu, Ting. Differentially private spatial decompositions. In *ICDE* (2012), pp. 20–31.

[20] Dang, Tuan Nhon, Anand, Anushka, and Wilkinson, Leland. Timeseer: Scagnostics for high-dimensional time series. *IEEE Transactions on Visualization and Computer Graphics 19*, 3 (2012), 470–483.

[21] Dasgupta, Aritra, Chen, Min, and Kosara, Robert. Measuring privacy and utility in privacy-preserving visualization. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 35–47.

[22] Dasgupta, Aritra, and Kosara, Robert. Adaptive privacy-preserving visualization using parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2241–2248.

[23] Dewri, Rinku. Local differential perturbations: Location privacy under approximate knowledge attackers. *IEEE Transactions on Mobile Computing 12*, 12 (2013), 2360–2372.

[24] Dinur, I., and Nissim, K. Revealing information while preserving privacy. In *PODS* (2003), pp. 202–210.

[25] Dwork, Cynthia. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation* (2008), Springer, pp. 1–19.

[26] Dwork, Cynthia, Nissim, Frank McSherry Kobbi, and Smith, Adam. Calibrating noise to sensitivity in private data analysis. In *TCC* (2006), pp. 265–284.

[27] Ebadi, Hamid, and Sands, David. Featherweight pinq. *JPC 7*, 2 (2017).

[28] Eichmann, Philipp, Zgraggen, Emanuel, Binnig, Carsten, and Kraska, Tim. Idebench: A benchmark for interactive data exploration. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (2020), pp. 1555–1569.

[29] Erlingsson, Úlfar, Pihur, Vasyl, and Korolova, Aleksandra. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS* (2014).

[30] Fong, David Chin-Lung, and Saunders, Michael. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput. 33*, 5 (Oct. 2011), 2950–2971.

[31] Gaboardi, Marco, Haeberlen, Andreas, Hsu, Justin, Narayan, Arjun, and Pierce, Benjamin C. Linear dependent types for differential privacy. In *POPL* (2013), pp. 357–370.

[32] Gaboardi, Marco, Honaker, James, King, Gary, Murtagh, Jack, Nissim, Kobbi, Ullman, Jonathan, and Vadhan, Salil. Psi ({\Psi}): a private data sharing interface. *arXiv preprint arXiv:1609.04340* (2016).

[33] Gelman, Andrew, and Tuerlinckx, Francis. Type s error rates for classical and bayesian single and multiple comparison procedures. *Computational Statistics 15*, 3 (2000), 373–390.

[34] Haney, Samuel, Machanavajjhala, Ashwin, Abowd, John, Graham, Matthew, Kutzbach, Mark, and Vilhuber, Lars. Utility cost of formal privacy for releasing national employer-employee statistics. In *SIGMOD* (2017).

[35] Hardt, Moritz, Ligett, Katrina, and McSherry, Frank. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems* (2012), pp. 2339–2347.

[36] Hardt, Moritz, Ligett, Katrina, and McSherry, Frank. A simple and practical algorithm for differentially private data release. In *NIPS* (2012).

[37] Harrison, Lane, Yang, Fumeng, Franconeri, Steven, and Chang, Remco. Ranking visualizations of correlation using weber's law. *IEEE transactions on visualization and computer graphics 20*, 12 (2014), 1943–1952.

[38] Hay, Michael, Machanavajjhala, Ashwin, Miklau, Gerome, Chen, Yan, and Zhang, Dan. Principled evaluation of differentially private algorithms using dpbench. In *SIGMOD* (2016).

[39] Hay, Michael, Machanavajjhala, Ashwin, Miklau, Gerome, Chen, Yan, and Zhang, Dan. Principled evaluation of differentially private algorithms using dpbench. In *Proceedings of the 2016 International Conference on Management of Data* (2016), ACM, pp. 139–154.

[40] Hay, Michael, Rastogi, Vibhor, Miklau, Gerome, and Suciu, Dan. Boosting the accuracy of differentially private histograms through consistency. *PVLDB* (2010).

[41] He, Xi, Raval, Nisarg, and Machanavajjhala, Ashwin. A demonstration of visdpt: Visual exploration of differentially private trajectories. *Proceedings of the VLDB Endowment 9*, 13 (2016), 1489–1492.

[42] Hengl, Tomislav, and Toomanian, Norair. Maps are not what they seem: representing uncertainty in soil-property maps. In *Proc. Accuracy* (2006), pp. 805–813.

[43] Huang, Zhiqi, McKenna, Ryan, Bissias, George, Miklau, Gerome, Hay, Michael, and Machanavajjhala, Ashwin. Psyndb: accurate and accessible private data generation. *Proceedings of the VLDB Endowment 12*, 12 (2019), 1918–1921.

[44] Kairouz, Peter, Oh, Sewoong, and Viswanath, Pramod. Extremal mechanisms for local differential privacy. In *Advances in neural information processing systems* (2014), pp. 2879–2887.

[45] Kellaris, Georgios, Papadopoulos, Stavros, and Papadias, Dimitris. Differentially private histograms for range-sum queries: A modular approach. *arXiv* (2015).

[46] Kim, Younghoon, and Heer, Jeffrey. Assessing effects of task and data distribution on the effectiveness of visual encodings. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 157–167.

[47] Kotsogiannis, Ios, Machanavajjhala, Ashwin, Hay, Michael, and Miklau, Gerome. Pythia: Data dependent differentially private algorithm selection. In *SIGMOD* (2017).

[48] Lee, Jaewoo, Wang, Yue, and Kifer, Daniel. Maximum likelihood postprocessing for differential privacy under consistency constraints. In *KDD* (2015).

[49] Li, Chao, Hay, Michael, and Miklau, Gerome. A data- and workload-aware algorithm for range queries under differential privacy. *PVLDB* (2014).

[50] Li, Chao, Hay, Michael, Miklau, Gerome, and Wang, Yue. A data-and workload-aware algorithm for range queries under differential privacy. *Proceedings of the VLDB Endowment 7*, 5 (2014), 341–352.

[51] Li, Chao, Hay, Michael, Rastogi, Vibhor, Miklau, Gerome, and McGregor, Andrew. Optimizing linear counting queries under differential privacy. In *PODS* (2010), pp. 123–134.

[52] Li, Chao, and Miklau, Gerome. An adaptive mechanism for accurate query answering under differential privacy. *PVLDB 5*, 6 (2012), 514–525.

[53] Li, Chao, Miklau, Gerome, Hay, Michael, McGregor, Andrew, and Rastogi, Vibhor. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB Journal* (2015), 1–25.

[54] Li, Chao, Miklau, Gerome, Hay, Michael, McGregor, Andrew, and Rastogi, Vibhor. The matrix mechanism: optimizing linear counting queries under differential privacy. *VLDB Journal* (2015).

[55] Li, Ninghui, Li, Tiancheng, and Venkatasubramanian, Suresh. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering* (2007), IEEE, pp. 106–115.

[56] Li, Tiancheng, and Li, Ninghui. On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), ACM, pp. 517–526.

[57] Liu, Jingcheng, and Talwar, Kunal. Private selection from private candidates. *CoRR abs/1811.07971* (2018).

[58] MacEachren, Alan M. Visualizing uncertain information. *Cartographic Perspectives*, 13 (1992), 10–19.

[59] MacEachren, Alan M, Robinson, Anthony, Hopper, Susan, Gardner, Steven, Murray, Robert, Gahegan, Mark, and Hetzler, Elisabeth. Visualizing geospatial information uncertainty: What we know and what we need to know. *Cartography and Geographic Information Science 32*, 3 (2005), 139–160.

[60] Machanavajjhala, Ashwin, Kifer, Daniel, Gehrke, Johannes, and Venkitasubramaniam, Muthuramakrishnan. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD) 1*, 1 (2007), 3–es.

[61] McKenna, Ryan, Miklau, Gerome, Hay, Michael, and Machanavajjhala, Ashwin. Optimizing error of high-dimensional statistical queries under differential privacy. *Proceedings of the VLDB Endowment 11*, 10 (2018), 1206–1219.

[62] McKenna, Ryan, Miklau, Gerome, Hay, Michael, and Machanavajjhala, Ashwin. Optimizing error of high-dimensional statistical queries under differential privacy. *PVLDB 11*, 10 (2018).

[63] McKenna, Ryan, Sheldon, Daniel, and Miklau, Gerome. Graphical-model based estimation and inference for differential privacy. In *ICML* (2019).

[64] McSherry, Frank D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD* (2009), pp. 19–30.

[65] McSherry, Frank D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* (2009), ACM, pp. 19–30.

[66] Mironov, Ilya. On significance of the least significant bits for differential privacy. In *CCS* (2012).

[67] Narayanan, A., and Shmatikov, V. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (2008), pp. 111–125.

[68] Narayanan, A., and Shmatikov, V. De-anonymizing social networks. In *Security and Privacy* (2009).

[69] Oksanen, Juha, Bergman, Cecilia, Sainio, Jani, and Westerholm, Jan. Methods for deriving and calibrating privacy-preserving heat maps from mobile sports tracking application data. *Journal of Transport Geography 48* (2015), 135–144.

[70] Olson, Judith S, and Kellogg, Wendy A. *Ways of Knowing in HCI*, vol. 2. Springer, 2014.

[71] Orr, Laurel, Balazinska, Magdalena, and Suciu, Dan. Probabilistic database summarization for interactive data exploration. *Proc. VLDB Endow. 10*, 10 (June 2017), 1154–1165.

[72] Pandey, Anshul Vikram, Krause, Josua, Felix, Cristian, Boy, Jeremy, and Bertini, Enrico. Towards understanding human similarity perception in the analysis of large sets of scatter plots. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016), pp. 3659–3669.

[73] Park, Yongjoo, Tajik, Ahmad Shahab, Cafarella, Michael, and Mozafari, Barzan. Database learning: Toward a database that becomes smarter every time. In *ACM Conference on Management of Data (SIGMOD)* (2017), pp. 587–602.

[74] Penrose, R. On best approximate solutions of linear matrix equations. *Mathematical Proceedings of the Cambridge Philosophical Society 52*, 1 (1956), 17–19.

[75] Potter, Kristin, Kniss, Joe, Riesenfeld, Richard, and Johnson, Chris R. Visualizing summary statistics and uncertainty. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 823–832.

[76] Proserpio, Davide, Goldberg, Sharon, and McSherry, Frank. A workflow for differentially-private graph synthesis. In *Workshop on online social networks* (2012).

[77] Proserpio, Davide, Goldberg, Sharon, and McSherry, Frank. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proc. VLDB Endow. 7*, 8 (Apr. 2014), 637–648.

[78] Qardaji, Wahbeh, Yang, Weining, and Li, Ninghui. Differentially private grids for geospatial data. In *ICDE* (2013), IEEE, pp. 757–768.

[79] Qardaji, Wahbeh, Yang, Weining, and Li, Ninghui. Understanding hierarchical methods for differentially private histograms. *PVLDB* (2013).

[80] Qardaji, Wahbeh, Yang, Weining, and Li, Ninghui. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment 6*, 14 (2013), 1954–1965.

[81] Qardaji, Wahbeh, Yang, Weining, and Li, Ninghui. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (2014), ACM, pp. 1435–1446.

[82] Rensink, Ronald A, and Baldridge, Gideon. The perception of correlation in scatterplots. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 1203–1210.

[83] Roth, Aaron, and Roughgarden, Tim. Interactive privacy via the median mechanism. In *Proceedings of the forty-second ACM symposium on Theory of computing* (2010), ACM, pp. 765–774.

[84] Roy, Indrajit, Setty, Srinath T. V., Kilzer, Ann, Shmatikov, Vitaly, and Witchel, Emmett. Airavat: Security and privacy for mapreduce. In *NSDI* (2010).

[85] Saket, Bahador, Endert, Alex, and Demiralp, Cagatay. Task-based effectiveness of basic visualizations. *IEEE transactions on visualization and computer graphics* (2018).

[86] Samarati, Pierangela, and Sweeney, Latanya. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.

[87] Sanyal, Jibonananda, Zhang, Song, Bhattacharya, Gargi, Amburn, Phil, and Moorhead, Robert. A user study to compare four uncertainty visualization methods for 1d and 2d datasets. *IEEE transactions on visualization and computer graphics 15*, 6 (2009), 1209–1218.

[88] Sanyal, Jibonananda, Zhang, Song, Dyer, Jamie, Mercer, Andrew, Amburn, Philip, and Moorhead, Robert J. Noodles: A tool for visualization of numerical weather model ensemble uncertainty. *Visualization and Computer Graphics, IEEE Transactions on 16*, 6 (2010), 1421–1430.

[89] Sarah Flood, Miriam King, Renae Rodgers Steven Ruggles, and Warren, J. Robert. Integrated public use microdata series, current population survey: Version 6.0 [dataset]. https://doi.org/10.18128/D030.V6.0, 2018.

[90] Sarikaya, Alper, and Gleicher, Michael. Scatterplots: Tasks, data, and designs. *IEEE transactions on visualization and computer graphics 24*, 1 (2017), 402–412.

[91] Sherman, Jack, and Morrison, Winifred J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Statist. 21*, 1 (03 1950), 124–127.

[92] Thaker, Pratiksha, Budiu, Mihai, Gopalan, Parikshit, Wieder, Udi, and Zaharia, Matei. Overlook: Differentially private exploratory visualization for big data. *arXiv preprint arXiv:2006.12018* (2020).

[93] Vaidya, J., Shafiq, B., Basu, A., and Hong, Y. Differentially private naive bayes classification. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)* (Nov. 2013), vol. 1, pp. 571–576.

[94] Veras, Rafael, and Collins, Christopher. Discriminability tests for visualization effectiveness and scalability. *IEEE transactions on visualization and computer graphics 26*, 1 (2019), 749–758.

[95] Visengeriyeva, Larysa. Advancing data curation with metadata and statistical relational learning.

[96] Wang, Xumeng, Chen, Wei, Chou, Jia-Kai, Bryan, Chris, Guan, Huihua, Chen, Wenlong, Pan, Rusheng, and Ma, Kwan-Liu. Graphprotector: A visual interface for employing and assessing multiple privacy preserving graph algorithms. *IEEE transactions on visualization and computer graphics 25*, 1 (2019), 193–203.

[97] Wang, Xumeng, Chou, Jia-Kai, Chen, Wei, Guan, Huihua, Chen, Wenlong, Lao, Tianyi, and Ma, Kwan-Liu. A utility-aware visual approach for anonymizing multi-attribute tabular data. *IEEE transactions on visualization and computer graphics 24*, 1 (2018), 351–360.

[98] Wilkinson, Leland, Anand, Anushka, and Grossman, Robert. Graph-theoretic scagnostics. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* (2005), IEEE, pp. 157–164.

[99] Wilkinson, Leland, and Wills, Graham. Scagnostics distributions. *Journal of Computational and Graphical Statistics 17*, 2 (2008), 473–491.

[100] Williams, Oliver, and McSherry, Frank. Probabilistic Inference and Differential Privacy. *NIPS* (2010), 2451–2459.

[101] Xiao, Xiaokui, Wang, Guozhang, and Gehrke, Johannes. Differential privacy via wavelet transforms. In *ICDE* (2010), pp. 225–236.

[102] Yeh, I-Cheng, and hui Lien, Che. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications* (2009), 2473–2480.

[103] Zhang, Dan, Hay, Michael, Miklau, Gerome, and O'Connor, Brendan. Challenges of visualizing differentially private data.

[104] Zhang, Dan, McKenna, Ryan, Kotsogiannis, Ios, Bissias, George, Hay, Michael, Machanavajjhala, Ashwin, and Miklau, Gerome. εktelo: A framework for defining differentially private computations. *ACM Transactions on Database Systems (TODS) 45*, 1 (2020), 1–44.

[105] Zhang, Dan, McKenna, Ryan, Kotsogiannis, Ios, Hay, Michael, Machanavajjhala, Ashwin, and Miklau, Gerome. Ektelo: A framework for defining differentially-private computations. In *Proceedings of the 2018 International Conference on Management of Data* (New York, NY, USA, 2018), SIGMOD '18, ACM, pp. 115–130.

[106] Zhang, Dan, McKenna, Ryan, Kotsogiannis, Ios, Hay, Michael, Machanavajjhala, Ashwin, and Miklau, Gerome. Ektelo: A framework for defining differentially-private computations. In *Proceedings of the 2018 International Conference on Management of Data* (2018), pp. 115–130.

[107] Zhang, Dan, Sarvghad, Ali, and Miklau, Gerome. Investigating visual analysis of differentially private data. *IEEE Transactions on Visualization and Computer Graphics* (2020).

[108] Zhang, Danfeng, and Kifer, Daniel. Lightdp: Towards automating differential privacy proofs. In *POPL* (2017), pp. 888–901.

[109] Zhang, Jun, Cormode, Graham, Procopiuc, Cecilia M., Srivastava, Divesh, and Xiao, Xiaokui. PrivBayes: Private data release via Bayesian networks. *TODS 42* (2017).

[110] Zhang, Jun, Xiao, Xioakui, and Xie, Xing. Privtree: A differentially private algorithm for hierarchical decompositions. In *SIGMOD* (2016).

[111] Zhang, Xiaojian, Chen, Rui, Xu, Jianliang, Meng, Xiaofeng, and Xie, Yingtao. Towards accurate histogram publication under differential privacy. In *SDM* (2014).