WINNING THE NIST CONTEST: A SCALABLE AND GENERAL APPROACH TO DIFFERENTIALLY PRIVATE SYNTHETIC DATA

RYAN MCKENNA, GEROME MIKLAU, AND DANIEL SHELDON

College of Information & Computer Sciences, The University of Massachusets, Amherst, MA 10002
e-mail address: rmckenna@cs.umass.edu

College of Information & Computer Sciences, The University of Massachusets, Amherst, MA 10002
e-mail address: miklau@cs.umass.edu

College of Information & Computer Sciences, The University of Massachusets, Amherst, MA 10002
e-mail address: sheldon@cs.umass.edu

Abstract. We propose a general approach for differentially private synthetic data generation, that consists of three steps: (1) **select** a collection of low-dimensional marginals, (2) **measure** those marginals with a noise addition mechanism, and (3) **generate** synthetic data that preserves the measured marginals well. Central to this approach is `Private-PGM` [42], a post-processing method that is used to estimate a high-dimensional data distribution from noisy measurements of its marginals. We present two mechanisms, `NIST-MST` and `MST`, that are instances of this general approach. `NIST-MST` was the winning mechanism in the 2018 NIST differential privacy synthetic data competition, and `MST` is a new mechanism that can work in more general settings, while still performing comparably to `NIST-MST`. We believe our general approach should be of broad interest, and can be adopted in future mechanisms for synthetic data generation.

1. Introduction

Data sharing within the modern enterprise is extremely constrained by privacy concerns. Privacy-preserving synthetic data is an appealing solution: it allows existing analytics workflows and machine learning methods to be used while the original data remains protected. But recent research has shown that unless a formal privacy standard is adopted, synthetic data can violate privacy in subtle ways [18,25]. Differential privacy offers such a formalism, and the problem of differentially private synthetic data generation has therefore received considerable research attention in recent years [3, 6, 9, 13, 14, 26, 31, 32, 39, 40, 52–55, 59, 60, 66, 68, 70, 71].

In 2018, the National Institute of Standards and Technology (NIST) highlighted the importance of this problem by organizing the *Differential Privacy Synthetic Data Competition* [56]. This competition was the first of its kind for the privacy research community, and it encouraged privacy researchers and practitioners to develop novel practical mechanisms for this task. The competition consisted of three rounds of increasing complexity. In this paper we describe `NIST-MST`, the winning entry in the third and final round of the competition. Our algorithm is an instance of a general template for differentially private synthetic data generation that we believe will simplify design of future mechanisms for synthetic data.

Our approach to differentially private synthetic data generation consists of three high-level steps, as show in Figure 1: (1) query selection, (2) query measurement and (3) synthetic

Key words and phrases: differential privacy, synthetic data, graphical models.

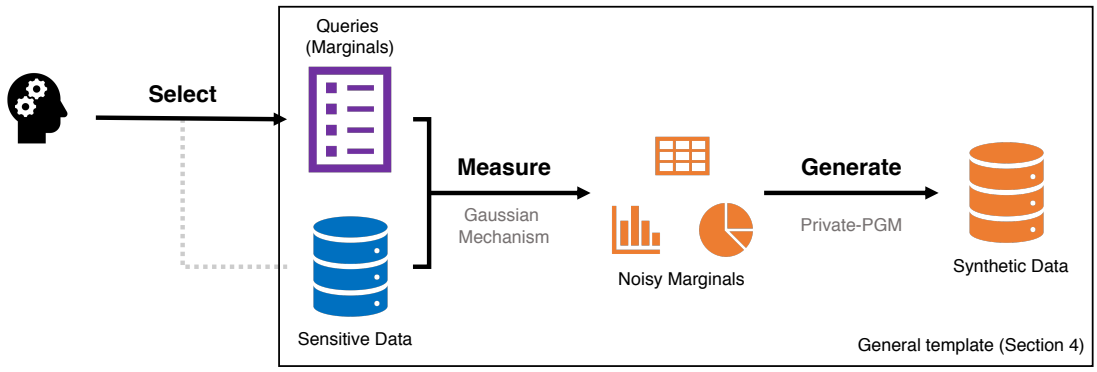

Figure 1. A general template for differentially private synthetic data generation. First, a collection of marginal queries is **selected**, either manually (e.g., by a domain expert) or automatically by an algorithm. Second, the Gaussian mechanism is used to **measure** those marginals while preserving differential privacy. Finally, `Private-PGM` is used to post-process the noisy marginals and **generate** a synthetic dataset that respects them.

data generation. For step (1), there are various ways to approach query selection; a domain expert familiar with the data and its use cases can specify the set of queries, or they can be automatically determined by an algorithm. The selected queries are important because they will ultimately determine the statistics for which the synthetic data preserves accuracy. For step (2), after the queries are fixed, they are measured privately with a noise-addition mechanism, in our case, with the Gaussian mechanism. In step (3), the noisy measurements are processed through `Private-PGM` [42], a post-processing method that can estimate a high-dimensional data distribution from noisy measurements and generate synthetic data.

This approach is similar in spirit to the widely studied select-measure-reconstruct paradigm for linear query answering under differential privacy [4, 16, 17, 29, 35–38, 38, 41, 46–48, 57, 58, 61–65, 67, 69]. However, the output is now synthetic data, rather than query answers. In addition, most existing methods from this paradigm suffer from the curse of dimensionality, and have trouble scaling to high-dimensional domains. Our approach is simple and modular but there are three main technical challenges to using it in practice. These are (1) identifying what statistics to measure about the dataset, (2) generating synthetic data that effectively preserves the measured statistics, and (3) overcoming the challenges of high-dimensional domains. Fortunately, `Private-PGM` solves problem (2) and (3) above, as long as the measured statistics only depend on the data through its low-dimensional marginals. This allows the mechanism designer to focus on problem (1), and frees them from the burden of figuring out *how* to generate synthetic data with differential privacy, allowing them instead to focus on *what* statistics to measure, based on what they want the synthetic data to preserve. Thus, we believe this approach to differentially privacy synthetic data, using `Private-PGM`, will be broadly applicable.

In Figure 1, there is a dashed gray line connecting the select step with the sensitive data. This indicates that query selection may or may not depend on the sensitive data, but if it does, it must be via a differentially private mechanism. The rules of the NIST competition permitted the use of a public provisional dataset which the `NIST-MST` algorithm uses for query selection. The effectiveness of query selection relies on the similarity of the public data to the private data being synthesized. Since high quality provisional data may not

always be available, we propose a variant of the algorithm, called `MST`, that does not require public data and instead uses a portion of the privacy budget to select measurements. The novelty of this algorithm is that it uses the data (privately) to select marginals to measure that support efficient synthesis in step (3). This extension leads to an algorithm that can be applied in a wider variety of settings. We show experimentally that, without the advantage of provisional data, it nevertheless performs comparably to `NIST-MST`.

This paper is organized as follows. In Section 2, we set up notation, state assumptions, and summarize relevant background in differential privacy. In Section 3, we summarize the important aspects of the competition. In Section 4, we describe the general template from Figure 1 in greater detail. In Section 5, we present `NIST-MST`, the winning mechanism from the competition, by building on the general template. In Section 6, we present `MST`, a novel mechanism inspired by `NIST-MST` that does not rely on the existence of public provisional data. We conclude with a simple experimental evaluation and discussion of results.

2. Background

The algorithms described in this paper take as input a dataset, assumed to be a single table, and generate a synthetic dataset satisfying (ϵ, δ)-differential privacy. Below we provide the relevant background and notation on datasets, marginals, and differential privacy.

2.1. Data.

The input is a dataset D consisting of m records, each containing potentially sensitive information about one individual. Each record has d attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$, and the domain of possible values for an attribute A_i is denoted by Ω_i. We assume Ω_i is finite and has size $|\Omega_i| = n_i$. The full domain of possible values is thus $\Omega = \Omega_1 \times \cdots \times \Omega_d$ which has size $\prod_i n_i = n$. We use $\mathcal{D}$ to denote the set of all possible datasets, which is equal to $\mathcal{D} = \cup_{m=0}^{\infty} \Omega^m$.

2.2. Marginals.

A marginal is a key statistic that captures low-dimensional structure in a high-dimensional data distribution. We will explain (in Section 3) that the evaluation metrics of the contest can be defined in terms of marginals computed on the dataset. In addition, our algorithms will privately measure selected marginals and use the resulting noisy measurements to construct synthetic data. More precisely, a marginal, for a set of attributes C, is a table that counts the number of occurrences of each combination of possible values for attributes C.

Definition 1 (Marginal)**.** *Let $C \subseteq \mathcal{A}$ be a subset of attributes, $\Omega_C = \prod_{i \in C} \Omega_i$, and $n_C = |\Omega_C|$. The marginal on C is a vector $\mu \in \mathbb{R}^{n_C}$, indexed by domain elements $t \in \Omega_C$, such that each entry is a count, i.e., $\mu_t = \sum_{x \in D} \mathbb{1}[x_C = t]$. We let $M_C : \mathcal{D} \to \mathbb{R}^{n_C}$ denote the function that computes the marginal on C, i.e., $\mu = M_C(D)$.*

2.3. **Differential privacy.** Differential privacy [20, 21] protects individuals by bounding the impact any one individual can have on the output of an algorithm. This is formalized using the notion of neighboring datasets. Two datasets $D, D' \in \mathcal{D}$ are neighbors (denoted $D \sim D'$) if D' can be obtained from D by adding or removing a single record.

Definition 2 (Differential Privacy [20])**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies (ϵ, δ)-differential privacy (DP) if for any neighboring datasets $D \sim D' \in \mathcal{D}$, and any subset of possible outputs $S \subseteq \mathcal{R}$,*

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(\epsilon) \Pr[\mathcal{M}(D') \in S] + \delta.$$

This definition requires that, on any two neighboring input databases, the difference in the output distributions of the randomized algorithm $\mathcal{M}$ is bounded by e^ϵ, except with a small failure probability δ. This failure probability δ is usually assumed to be cryptographically small; in the contest it was set to $\delta \approx 2 \cdot 10^{-12}$. The algorithms in this paper achieve differential privacy by repeated application of the Gaussian mechanism and the Exponential Mechanism, defined below:

Definition 3 (Gaussian Mechanism)**.** *Let $f : \mathcal{D} \to \mathbb{R}^p$ be a vector-valued function of the input data. The Gaussian Mechanism adds i.i.d. Gaussian noise with scale σ to $f(D)$:*

$$\mathcal{M}(D) = f(D) + \mathcal{N}(0, \sigma^2 \mathbf{I}).$$

Definition 4 (Exponential Mechanism)**.** *Let $q : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$ be quality score function and ϵ be a parameter. Then the exponential mechanism outputs a candidate $r \in \mathcal{R}$ according to the following distribution:*

$$\Pr[\mathcal{M}(D) = r] \propto \exp\Big(\epsilon \cdot q(D, r)\Big)$$

To accurately analyze the privacy of multiple invocations of the Gaussian/Exponential mechanisms (i.e., to derive the (ϵ, δ) parameters) we use the tools of Rényi Differential Privacy (RDP), a variant of differential privacy so named because it uses the Rényi divergence in the bound on a mechanism's output distributions for neighboring inputs.

Definition 5 (Rényi Differential Privacy [45])**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies (α, γ)-Rényi differential privacy (RDP) for $\alpha \geq 1$ and $\gamma \geq 0$, if for any neighboring datasets $D \sim D' \in \mathcal{D}$, we have:*

$$D_\alpha(\mathcal{M}(D) \,||\, \mathcal{M}(D')) \leq \gamma,$$

where $D_\alpha(\cdot \,||\, \cdot)$ is the Rényi divergence of order α between two probability distributions.

To analyze the privacy of the mechanisms above under Rényi-DP, we define the sensitivity of a vector-valued query as follows:

Definition 6 (Sensitivity)**.** *Let $f : \mathcal{D} \to \mathbb{R}^p$ be a vector-valued function of the input data. The L_2 sensitivity of f is $\Delta_f = \max_{D \sim D'} \|f(D) - f(D')\|_2$.*

It is easy to verify that the L_2 sensitivity of any marginal function M_C is 1, regardless of the attributes in C. This is because one individual can only contribute a count of 1 to a single cell of the output vector. A single invocation of the Gaussian Mechanism satisfies Rényi-DP with parameters determined by the noise scale σ and the sensitivity Δ_f of the function. Similarly, a single invocation of the Exponential Mechanism also satisfies Rényi-DP with parameters determined by ϵ and Δ_q.

Proposition 1 (Rényi-DP of the Gaussian Mechanism [22, 45]). *The Gaussian Mechanism applied to the function $f : \mathcal{D} \to \mathbb{R}^p$ satisfies $\left(\alpha, \alpha\frac{\Delta_f^2}{2\sigma^2}\right)$-RDP for all $\alpha \geq 1$.*

Proposition 2 (Rényi-DP of the Exponential Mechanism [12, 44]). *The Exponential Mechanism applied to the quality score function $q : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$ satisfies $(2\epsilon\Delta, 0)$-DP and $(\alpha, \alpha\frac{(2\epsilon\Delta)^2}{8})$-RDP for all $\alpha \geq 1$, where $\Delta = \max_{r\in\mathcal{R}} \Delta_{q(\cdot,r)}$ is the maximum sensitivity of q.*

Note that any mechanism that is $(\alpha, \alpha\rho)$-RDP for all $\alpha \geq 1$ is also ρ-zCDP [11] and vice-versa. We rely on the following propositions to reason about multiple adaptive invocations of RDP mechanisms, and the translation between Rényi-DP and (ϵ, δ)-DP.

Proposition 3 (Adaptive Composition of RDP Mechanisms [45]). *Let $\mathcal{M}_1 : \mathcal{D} \to \mathcal{R}_1$ be (α, γ_1)-RDP and $\mathcal{M}_2 : \mathcal{D} \times \mathcal{R}_1 \to \mathcal{R}_2$ be (α, γ_2)-RDP. Then the mechanism $\mathcal{M} = \mathcal{M}_2(D, \mathcal{M}_1(D))$ is $(\alpha, \gamma_1 + \gamma_2)$-RDP.*

Proposition 4 (RDP to DP [45]). *If a mechanism $\mathcal{M}$ satisfies (α, γ)-Rényi differential privacy, it also satisfies $\left(\gamma + \frac{\log(1/\delta)}{\alpha-1}, \delta\right)$-differential privacy for all $\delta \in (0, 1]$.*

3. Competition setup

In this section we will summarize the format of the competition and the different components of the challenge problem. The competition consisted of three rounds of increasing complexity, but our focus is on the third round, which built on the previous two rounds.

3.1. Competition Format.

Competitors were given approximately one month to design their differentially private synthetic data mechanism. The competition organizers provided contestants with a precise problem specification along with a "competitor pack", which included a provisional dataset to test and develop mechanisms, a file that contained domain information for each attribute in the dataset, a script to evaluate the quality of the synthetic data according to their custom scoring criteria, and a baseline mechanism. Each of these components will be described in detail in the subsequent sections. During the one-month competition period, competitors could submit the synthetic data produced by their mechanism to be scored and attain a spot on the provisional leaderboard. This was a good way to gauge how well other competitors were doing, although it was not an authoritative source as the submissions had not been vetted to ensure they satisfied differential privacy (e.g., someone could submit the true data and get a perfect score on the provisional leaderboard).

At the end of the competition period, competitors had to submit their source code along with a document describing the solution and a proof of privacy. A team of experts unknown to the competitors checked the final submitted algorithms to ensure that they satisfied (ϵ, δ)-differential privacy. This was done by checking the written description of the algorithm as well as the source code, to ensure there were no privacy violations or mistakes. After the mechanism was verified to be differentially private, its utility was evaluated on the final dataset (different from the provisional dataset) and performance results were added to a final leaderboard that would determine the ranking of solutions.

In the subsequent subsections, we will specify the details of each component of the problem — i.e., the dataset, the domain, and the evaluation criteria.

3.2. **Dataset.** Algorithms were designed for and evaluated on data from the 1940 U.S. decennial census. The provisional dataset contained data for one state (Colorado) and the final holdout data was for a different state (unknown at the time of the competition). We remark that the provisional dataset was treated as public information. Therefore, any analysis and insights derived from it were not considered to violate privacy. However, solutions that used information from the provisional dataset too aggressively risked over-fitting, and scoring poorly on the final dataset. The provisional dataset contained 98 attributes and about 661 thousand records. All attributes were discrete, taking on values from the domain $\Omega_i = \{0, \ldots, n_i - 1\}$. The value of n_i for each attribute i was provided in a separate specs file. The values of n_i ranged from 2 (for binary attributes like `SEX`) to $10,000,000$ (for numerical attributes like `INCWAGE`). The total number of possible database rows (i.e., the full domain size) was about $|\Omega| = 5 \times 10^{205}$. We provide a full breakdown of the domain in Table 4.

3.3. **Evaluation metrics.** The utility of the synthetic data was measured by how well it preserved key statistics in the ground truth data with respect to three main criteria, enumerated below. We state below the statistics that need to be preserved to score well, but not the exact formula for calculating score. For the precise information, please refer to the official challenge problem statement [56]. Note that the scores for each evaluation metric were normalized to the same range, and averaged across the three metrics (with equal weights). Algorithms were evaluated at three privacy levels, with $\epsilon = 0.3, 1.0, 8.0$ and $\delta \approx 2 \cdot 10^{-12}$, and these scores were averaged to obtain the final score. Computational efficiency was not taken into consideration; several of the solutions (including `NIST-MST`) required up to 10 minutes or more to run.

(1) **3-way Marginals.** The synthetic data was evaluated by comparing its marginals with the marginals of the true data for 100 random triples of attributes, unknown to competitors at submission time. Therefore, a synthetic dataset $\tilde{D}$ scores well on this metric (in expectation) if $M_C(D) \approx M_C(\tilde{D})$ *for all* triples C. There are a total of $\binom{98}{3} = 152096$ possible triples, so this evaluation criteria requires the synthetic data to preserve a large number of marginals to consistently score well.

(2) **High-order conjunctions.** The synthetic data was evaluated by looking at how well it preserved high-order conjunctions. Probabilistically, a high-order conjunction for a set of attributes $C \subseteq \mathcal{A}$ assumes the form $\Pr[\bigwedge_{i \in C}[t_i \in S_i] \mid t \sim D]$, where $S_i \subseteq \Omega_i$ is a subset of the domain for attribute i. This quantity can be expressed in terms of the marginal $\mu = M_C(D)$ via $\sum_{t \in S} \mu_t$, where S is the Cartesian product of S_i's, i.e., $S = \prod_{i \in C} S_i$ and t is a tuple restricted to the attributes in the set C. Synthetic data was evaluated on 300 random high-order conjunctions, where C is generated with a simple random sample of the attributes $\mathcal{A}$ with selection probability 0.1, and S_i is a random subset of Ω_i. There are a total of $2^{98} \approx 10^{29}$ possible choices for C, and the expected size of C is $0.1 \cdot 98 \approx 10$. Even without accounting for the variability in S_i, it is clear that the number of statistics that need to be preserved is enormous.

(3) **Income inequality and gender wage gap.** The synthetic data was evaluated by how well it preserved statistics relating to income inequality and gender wage gap, broken down by city. While the precise details of this metric can be found in the official problem statement, to score well, it suffices for the synthetic data to be accurate with respect to the marginal on (`SEX,CITY,INCWAGE`). Unlike metrics (1) and (2), above, this metric is relatively easy to score well on, because it just requires preserving one marginal well.

```
1  from private_pgm import FactoredInference
2  from scipy.sparse import identity
3  from numpy.random import normal
4
5  data = load_NIST()
6  queries = [("SEX","LABFORCE"), ("LABFORCE","SCHOOL")]
7  measurement_log = []
8  for c in queries:
9      M_c = data.project(c).datavector()
10     y_c = M_c + normal(loc=0, scale=50, size=M_c.size)
11     measurement_log.append( (identity(M_c.size), y_c, 50, c) )
12 engine = FactoredInference(data.domain)
13 model = engine.estimate(measurements)
14 synth = model.synthetic_data()
```

Figure 2. A demonstration of how to generate synthetic data with `Private-PGM` using real `Python` code. In this case, the selected marginals are (`SEX,LABFORCE`) and (`LABFORCE,SCHOOL`). In Lines 9-12, these marginals are measured with Gaussian noise to protect privacy. In Lines 14-16, `Private-PGM` takes these noisy measurements as input, estimates a model, and generates synthetic data. The `Private-PGM` library provides a straightforward interface that allows users to quickly write end-to-end code to generate synthetic data; different statistics can be preserved by changing Line 6.

4. Overview of Measurement and Inference with Private-PGM

In this section, we elaborate on the general template for a mechanism outlined in Figure 1. Recall there are three high-level steps:

(1) **Select.** Select a collection of marginals to measure.
(2) **Measure.** Use the Gaussian mechanism to measure each marginal in the collection.
(3) **Generate.** Use `Private-PGM` to estimate a data distribution from the noisy measurements and generate synthetic data that preserves the measured marginals well.

In this section, we describe the latter two steps, which form the core of the mechanism. In the next section, we will describe the full mechanism `NIST-MST` in detail, including the *select* step and many other details relating specifically to the NIST contest and dataset. Figure 2 shows how simple and modular this framework for synthetic data generation is. The open source `Private-PGM` library[1] provides a simple interface to the key routines so that an end-to-end synthetic data generation mechanism can be written with very little code, allowing the modeler to focus on tailoring the procedure to the workload and domain. Under the hood, `Private-PGM` has thousands of lines of code, but it exposes a simple interface that is easy to use. In this example, there are only two selected marginal queries: (`SEX,LABFORCE`) and (`LABFORCE,SCHOOL`), but the code can be readily modified (Line 6) to accommodate other marginal queries. In the rest of the section, we describe this general approach in more detail, and give some insight into the steps described in this code snippet.

4.1. Measuring Marginals with the Gaussian Mechanism.

Algorithm 1 shows the method for measuring marginals. Given a collection of attribute subsets $\mathcal{C}$, it computes the marginal for each $C \in \mathcal{C}$ and adds i.i.d. Gaussian noise to preserve privacy. It also accepts a weight w_C for each attribute subset, which represents the relative importance of

[1] https://github.com/ryan112358/private-pgm/

Algorithm 1: Measure Marginals

Input: D (sensitive dataset), $\mathcal{C}$ (a collection of attribute subsets), w_C (weights for each $C \in \mathcal{C}$), σ (noise scale)

Output: log (a list of noisy measurements together with metadata)

(1) Normalize weights, $w_C \leftarrow w_C / \sqrt{\sum_C w_C^2}$.

(2) For $C \in \mathcal{C}$:

(3) Calculate noisy marginal, $\tilde{\mu} = w_C M_C(D) + \mathcal{N}(0, \sigma^2 I)$

(4) Append 4-tuple $(w_C I,\ \tilde{\mu},\ \sigma,\ C)$ to measurement log

that marginal. It collects all of these noisy measurements into a *measurement log*, which will be passed to `Private-PGM` for post-processing. The measurement log records, for each marginal defined by a subset of attributes, the noisy marginal query answers together with information about the weight assigned to the marginal and the magnitude of noise used to measure it. It is easy to verify the privacy properties of Algorithm 1, as it is a direct application of the Gaussian mechanism on a sensitivity-1 quantity.[2]

Theorem 1. *Algorithm 1 satisfies* $(\alpha, \frac{\alpha}{2\sigma^2})$*-RDP for all* $\alpha \geq 1$.

4.2. `Private-PGM`**: Inference and Synthetic Data Generation.** `Private-PGM` is a general-purpose post-processing tool to infer a data distribution given noisy measurements [42]. It is compatible with measurements from a wide variety of mechanisms for discrete data, and can often improve utility at no cost to privacy. Because it infers a representation of a full data distribution, it produces query answers that are *consistent* with one another, even if the noisy measurements are inconsistent. It uses a compact representation of the data distribution to avoid exponential complexity in many cases, though the size of the representation will depend on the measurements, as we describe below.

The high-level idea of `Private-PGM` is to solve an optimization problem to find a data distribution that would produce measurements close to the ones that were observed. It applies to cases when private measurements depend on the data through marginals. For example, suppose the measurements are of the form

$$y_C = Q_C M_C(D) + \xi$$

for all attributes sets C in some collection $\mathcal{C}$, where $Q_C \in \mathbb{R}^{p_C \times n_C}$ is a linear transformation applied to the marginal prior to release and $\xi \in \mathbb{R}^{p_C}$ is zero-centered noise (e.g., Laplace or Gaussian) with known standard deviation. The measurements taken in Algorithm 1 represent the common case where Q_C is just the identity matrix, so that we observe the noisy marginals directly. However, the ability to measure arbitrary linear transformations of marginals is a nice feature that is useful for some types of measurements that occur in practice.[3] Examples of this include hierarchical measurements for answering range queries [29, 35, 47], and optimized measurements for answering general linear query workloads [36, 41].

Given these measurements, `Private-PGM` infers a data distribution P that best explains the measurements by solving the optimization problem

$$\underset{P}{\operatorname{argmin}} \sum_{C \in \mathcal{C}} \|Q_C M_C(P) - y_C\|_2^2 \,. \tag{4.1}$$

[2]The weights are explicitly normalized so that the collection of marginals has sensitivity 1.

[3]In fact, Q_C can be replaced with an arbitrary non-linear differentiable transformation, and `Private-PGM` will accept that as input as well.

The objective of this optimization problem is the negative log-likelihood of the noisy measurements under the Gaussian release mechanism, so Equation (4.1) can be seen as a maximum likelihood estimator. We have abused notation by allowing M_C to operate on a data distribution rather than a dataset; the correct interpretation is to substitute the probability vector P for the contingency table representation D of the dataset for computing the marginal. An obvious issue with the optimization problem in Equation (4.1) is that the dimension of the decision variable P is equal to the domain size n, which is exponential in the number of attributes, so we cannot usually solve this problem directly. The key observation of `Private-PGM` is

Fact 1. *Equation (4.1) has an optimum of the form P_θ, where P_θ is a graphical model with one factor for each set $C \in \mathcal{C}$ of attributes for which the mechanism measured a marginal.*

This allows us to solve the much lower-dimensional optimization problem

$$\underset{\theta}{\operatorname{argmin}} \sum_{C \in \mathcal{C}} \|Q_C M_C(P_\theta) - y_C\|_2^2 , \tag{4.2}$$

with no loss in solution quality. The decision variable θ is the parameter vector of the graphical model, and has dimension equal to the total length of the set of measured marginals. A simple proximal algorithm is given in [42] that solves this optimization problem using only repeated calls to a routine to perform marginal inference in a discrete graphical model — i.e., computing $M_C(P_\theta)$ for all $C \in \mathcal{C}$ and various different θ. The procedure is efficient whenever marginal inference in the graphical model is efficient. Belief propagation is the standard way to perform marginal inference in practice, as it efficiently computes $M_C(P_\theta)$ directly in terms of θ without ever explicitly materializing the full joint distribution P_θ [33].

Remark 1 (Scalability of `Private-PGM`)**.** `Private-PGM` is able to scale to very high-dimensional domains. The main factors that influence it's scalability are (1) the total size of the parameter vector θ and (2) the structure of the set $\mathcal{C}$. The size of θ is the same as the size of all of the relevant marginals combined, and that must not be too large. The size of each marginal depends directly on $|\Omega_i|$, the number of possible values for each attribute. Furthermore, the set $\mathcal{C}$ is important because it corresponds to the structure of the graphical model, and belief propagation is most efficient for tree-structured models. The scalability of belief propagation and `Private-PGM` for non tree-structured models depends on a quantity known as the tree width, which is a measure of how "tree-like" the model is [33].

Remark 2 (Lack of modeling assumptions)**.** It is easy to misconstrue the meaning of the graphical model representation. The inferred distribution P_θ will satisfy conditional independence properties dictated by the structure of the model. However, *no approximation is made when solving the optimization problem in Equation (4.1)*, and *the independence properties do not arise from assumptions made by the modeler about the structure of the data distribution.* By Fact 1, there is an optimum to Equation (4.1) that is a graphical model, and hence satisfies these conditional independence properties. Moreover, the graphical model solution P_θ can be shown to have maximum entropy among all optima of Equation (4.1) [42].

Once P_θ is estimated, `Private-PGM` can be used for multiple purposes: reducing error on measured marginals, estimating unmeasured marginals, and even generating synthetic tabular data. These use cases are explained in greater detail below:

Reducing error on measured marginals. First, `Private-PGM` improves utility by combining all sources of measured information into a single cohesive estimate for the data distribution. When the measurements are inconsistent with each other, `Private-PGM` resolves these inconsistencies in a principled manner, reducing variance and boosting utility. One achieves this by using $\bar{y}_C = Q_C M_C(P_\theta)$ in place of the noisy observation y_C. The estimated marginal $\bar{y}_C$ will typically have smaller variance than y_C and will often have lower overall error as well, therefore offering immediate utility improvements at no cost to privacy. Example 1 demonstrates this idea more concretely in a toy setting.

Example 1 (Boosting utility on measured marginals)**.** *We draw* 1000 *tuples from the actual contest dataset and measure two of their marginals, (`SEX,LABFORCE`) and (`LABFORCE,SCHOOL`), using the Gaussian mechanism with* $\sigma = 50$*. Tables (a–c) below show the true marginals, the noisy marginals, and the marginals estimated by `Private-PGM`. One can easily verify that the noisy marginals are not consistent: the (`SEX,LABFORCE`) marginal implies the total number of people with `LABFORCE=N` is* $124.549 + 318.029 = 442.578$*, while the (`LABFORCE,SCHOOL`) marginal implies the same that number is* $287.215 + 171.134 = 458.349$*. These are two different estimates for the same quantity, which is a consistency problem. In contrast, the `Private-PGM` estimated marginals are consistent: they both agree that the total number is* 436.873*. Additionally, `Private-PGM` better estimates the true marginals than the noisy marginals do: the* L_1 *distances are about* 213 *and* 272 *for `Private-PGM`, while they are about* 251 *and* 295 *for the noisy marginals, which is a significant boost in utility.*

SEX	LABFORCE	count
M	—	156
M	N	65
M	Y	316
F	—	158
F	N	282
F	Y	23

LABFORCE	SCHOOL	count
—	N	159
—	Y	155
N	N	288
N	Y	59
Y	N	336
Y	Y	3

(a) True marginals

SEX	LABFORCE	count
M	—	132.428
M	N	124.549
M	Y	244.365
F	—	173.633
F	N	318.029
F	Y	−21.358

LABFORCE	SCHOOL	count
—	N	116.021
—	Y	186.826
N	N	287.215
N	Y	171.134
Y	N	278.498
Y	Y	−46.497

(b) Noisy marginals

SEX	LABFORCE	count
M	—	124.829
M	N	121.696
M	Y	254.636
F	—	166.034
F	N	315.177
F	Y	0

LABFORCE	SCHOOL	count
—	N	110.029
—	Y	180.834
N	N	276.477
N	Y	160.396
Y	N	254.636
Y	Y	0

(c) `Private-PGM` marginals

Estimating unmeasured marginals. Second, `Private-PGM` can be used to answer new queries that were never measured directly by using P_θ in place of the true data D. This allows us to estimate new marginals without spending the privacy budget, saving a precious resource. Example 2 demonstrates this idea in a toy setting.

Example 2 (Estimating new marginals)**.** *Building on Example 1, recall that we measured the marginals on (`SEX,LABFORCE`) and (`LABFORCE,SCHOOL`). We can use `Private-PGM` to estimate the marginal on (`SEX,SCHOOL`), even though we never measured it and it can not be directly inferred from the other marginals that were measured. As shown below in Table (b), the provided estimate is reasonable, given that we never measured it, and we added significant noise to the marginals we did measure. We reiterate that we obtained this estimate "for free", without spending additional privacy budget. Additionally, `Private-PGM` can estimate*

the marginal on (SEX,LABFORCE,SCHOOL), which is shown in Table (d). This is pretty close to the true 3-way marginal, shown in Table (c). In fact, the normalized L_1 error is only 0.135. *While there may be other equally good estimates for the 3-way marginal (according to the loss function in Equation (4.1)), the estimate provided by Private-PGM has maximum entropy among all of them. In this case, Private-PGM was fairly accurate because SEX and SCHOOL are (approximately) conditionally independent given LABFORCE in the true data.*

SEX	SCHOOL	count
M	N	423
M	Y	114
F	N	360
F	Y	103

(a) True 2-way marginal

SEX	SCHOOL	count
M	N	378.873
M	Y	122.289
F	N	262.269
F	Y	218.942

(b) Estimated 2-way marginal

SEX	LABFORCE	SCHOOL	count
M	—	N	74
M	—	Y	82
M	N	N	36
M	N	Y	29
M	Y	N	313
M	Y	Y	3
F	—	N	85
F	—	Y	73
F	N	N	252
F	N	Y	30
F	Y	N	23
F	Y	Y	0

(c) True 3-way marginal

SEX	LABFORCE	SCHOOL	count
M	—	N	47.221
M	—	Y	77.608
M	N	N	77.016
M	N	Y	44.68
M	Y	N	254.636
M	Y	Y	0.000
F	—	N	62.808
F	—	Y	103.226
F	N	N	199.461
F	N	Y	115.716
F	Y	N	0.000
F	Y	Y	0.000

(d) Estimated 3-way marginal

Generating synthetic data. Third, Private-PGM can be used to generate synthetic data $\bar{D}$ in tabular format. $\bar{D}$ can be used in place of P_θ and will generally give similar results. They will not give exactly the same results because $\bar{D}$ has integer-valued marginals while P_θ has real-valued marginals, so some additional rounding error is unavoidable. One can obtain the synthetic data in multiple ways; a simple and natural approach would be to sample records from P_θ to form a synthetic dataset. This naive approach would introduce sampling error which is undesirable. Private-PGM uses an alternative approach to reduce error from additional sources of randomness. The details of this procedure are available in the open-source implementation of Private-PGM, and are summarized in the supplementary material. In Example 3, we give an intuitive idea of how this procedure works, and illustrate why it is preferable to the sampling approach.

Example 3 (Generating synthetic data)**.** *Building on Examples 1 and 2, we calculate the LABFORCE marginal from the Private-PGM model in Table (a) below, which has fractional counts. We also use Private-PGM to generate synthetic data and show the same marginal in Table (b), which has integer counts. These two marginals almost exactly match, because Private-PGM tries to preserve the model marginals as closely as possible when generating synthetic data. However, synthetic data obtained by i.i.d sampling will not match the model marginals as closely due to the randomness in sampling, as shown in Table (c).*

LABFORCE	count
—	290.863
N	436.873
Y	254.636

(a) Private-PGM model marginal

LABFORCE	count
—	291
N	437
Y	254

(b) Private-PGM synthetic data marginal

LABFORCE	count
—	262
N	468
Y	252

(c) Sampled synthetic data marginal

Algorithm 2: `NIST-MST`

(1)	**Calibrate Noise**	Derive noise scale σ from target privacy parameters (ϵ, δ)	Equation (5.1)
(2)	**Encode Domain**	Use public information to encode attribute domains	
(3)	**Transform Data**	Transform data using insights from provisional data	Algorithm 3
(4)	**Compress Domain**	Use data to reduce domain:	
	Measure	Measure all one-way marginals	Algorithm 1
	Compress	Remove domain elements failing threshold test	Algorithm 4
(5)	**Select Marginals**	Select a subset of 2- and 3-way marginals	Algorithm 5
(6)	**Measure Marginals**	Measure selected marginals	Algorithm 1
(7)	**Synthesize data**	Synthesize records using `Private-PGM`:	
	Estimate	Estimate distribution from Step 4 and 6 measurements	Equation (4.2)
	Generate	Generate synthetic records	Algorithm 8
(8)	**Reverse**	Reverse the transformation made in Step 3	Algorithm 9

5. Algorithm Description

In this section we describe `NIST-MST`, which takes the basic mechanism template outlined in the previous section, and applies it to the setting of the NIST competition. `NIST-MST` simply invokes the Gaussian mechanism to measure a carefully chosen subset of 1, 2, and 3-way marginals. Then the resulting noisy measurements are post-processed using `Private-PGM` to obtain synthetic data that is most consistent with those marginals. `NIST-MST` does not follow the template from the previous section exactly, as there are two rounds of measurements, and an additional domain compression step developed specifically to deal with some of the challenges around the dataset used in the competition. The high-level steps of `NIST-MST` are stated in Algorithm 2, and a detailed description of each step will be provided in this section, with motivations and intuitions for the various design choices.

Step 1: Calibrate Noise. In this step, σ is calibrated to ensure the whole algorithm satisfies (ϵ, δ)-differential privacy. Note that only steps (4) and (6) in Algorithm 2 use the sensitive data, and these are both invocations of Algorithm 1, which is $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP (Theorem 1). The data transformations made in steps (3) and (4) do not affect the privacy analysis of Algorithm 1 since one individual can still only affect each marginal by at most one. Hence `NIST-MST` is $(\alpha, \frac{\alpha}{\sigma^2})$-RDP by two-fold adaptive composition (Proposition 3). Moreover, by Proposition 4, `NIST-MST` is $\left(\frac{\alpha}{\sigma^2} + \frac{\log(1/\delta)}{\alpha-1}, \delta\right)$-DP for all $\alpha \geq 1$.

For a fixed α, it is easy to determine σ by solving the equation $\frac{\alpha}{\sigma^2} + \frac{\log(1/\delta)}{\alpha-1} = \epsilon$ for σ. The best value of σ can be obtained by minimizing over all α. In the contest, this computation was done by invoking the `moments accountant` [2], which minimizes over $\alpha = 1, \ldots, 512$. However, this minimization can actually be done in closed form [72], leading to the following equation for σ:

$$\sigma = \frac{\sqrt{\log(1/\delta)} + \sqrt{\log(1/\delta) + \epsilon}}{\epsilon} \tag{5.1}$$

Theorem 2 (Privacy of `NIST-MST`). *Algorithm 2 is (ϵ, δ)-differentially private.*

Proof. From the analysis above, we know that `NIST-MST` is $\left(\frac{\alpha}{\sigma^2} + \frac{\log(1/\delta)}{\alpha-1}, \delta\right)$-DP for all $\alpha \geq 1$. By plugging in $\alpha = 1 + \sigma\sqrt{\log(1/\delta)}$ and $\sigma = \frac{\sqrt{\log(1/\delta)}+\sqrt{\log(1/\delta)+\epsilon}}{\epsilon}$ and simplifying, we see that $\frac{\alpha}{\sigma^2} + \frac{\log(1/\delta)}{\alpha-1} = \epsilon$, and hence `NIST-MST` is (ϵ, δ)-DP as desired. The algebraic

1. SPLIT	2/2	21. NCHLT5	7/7	41. SIZEPL	31/19	61. SUPDIST	631/631	81. ENUMDIST	3021/3021
2. SLREC	3/2	22. RACE	7/7	42. EMPSTATD	35/15	62. METAREA	657/334	82. CITYPOP	3225/3225
3. SEX	3/2	23. WKSWORK2	7/7	43. WKSWORK1	53/53	63. PRESGL	816/816	83. URBPOP	3225/3225
4. SCHOOL	3/2	24. VET1940	9/4	44. SEA	54/54	64. BPL	901/163	84. METAREAD	6561/378
5. URBAN	3/3	25. UCLASSWK	9/8	45. OCCSCORE	81/81	65. MBPL	901/164	85. MTONGUED	9602/489
6. FARM	3/3	26. VETPER	9/8	46. AGEMARR	90/89	66. FBPL	901/165	86. MIGMET5	10000/379
7. OWNERSHP	3/3	27. HISPRULE	9/9	47. MIGRATE5D	91/15	67. IND1950	998/162	87. MIGCOUNTY	10000/385
8. RESPONDT	3/3	28. HRSWORK2	9/9	48. MTONGUE	97/92	68. MIGSEA5	998/510	88. ERSCOR50	10000/1002
9. SPANNAME	3/3	29. CLASSWKR	10/4	49. SEI	97/97	69. OCC	999/231	89. EDSCOR50	10000/1002
10. LABFORCE	3/3	30. INCNONWG	10/4	50. CLASSWKRD	99/18	70. EDUCD	1000/44	90. NPBOSS50	10000/1002
11. VETWWI	3/3	31. SAMEPLAC	10/4	51. HRSWORK1	99/99	71. HIGRADED	1000/69	91. CITY	10000/1164
12. SSENROLL	3/3	32. VETSTAT	10/4	52. VETSTATD	100/10	72. GQTYPED	1000/92	92. MIGCITY5	10000/1164
13. METRO	4/4	33. VETCHILD	10/5	53. GQFUNDS	100/13	73. IND	1000/136	93. RENT	10000/10000
14. EMPSTAT	4/4	34. MIGTYPE5	10/6	54. EDUC	100/13	74. UIND	1000/136	94. MBPLD	90021/537
15. HISPAN	5/5	35. SAMESEA5	10/6	55. AGEMONTH	100/15	75. MIGPLAC5	1000/199	95. FBPLD	90021/539
16. CITIZEN	5/5	36. MIGRATE5	10/7	56. HIGRADE	100/25	76. UOCC	1000/231	96. BPLD	90022/536
17. WARD	6/6	37. GQTYPE	10/10	57. CHBORN	100/62	77. UOCC95	1000/279	97. VALUEH	10000000/5003
18. NATIVITY	6/6	38. MARRNO	10/10	58. AGE	109/109	78. OCC1950	1000/283	98. INCWAGE_A	—/52
19. MARST	7/6	39. OWNERSHPD	21/8	59. HISPAND	481/55	79. DURUNEMP	1000/1000	99. INCWAGE_B	—/8
20. GQ	7/7	40. FAMSIZE	22/22	60. RACED	621/238	80. COUNTY	1251/385	INCWAGE	10000000/—

Table 4. Domain information for the census dataset used in the third round of the competition. Table specifies attribute names, and number of possible values for that attribute according to (1) the provided specs file and (2) the specs file combined with IPUMS documentation.

manipulation is routine but messy; it can easily be verified with `sympy` (see Figure 4 in the supplement). □

Remark 3 (Noise Calibration)**.** It is well known that calibrating σ via an RDP analysis does not give the smallest possible value required to achieve (ϵ, δ)-DP, and an analytic calibration gives strictly better results [7], at least for a *single* invocation of the Gaussian mechanism. However, at the time of the competition, adaptive composition of *two* Gaussian mechanisms was needed, and RDP was chosen because of its clean and well-understood guarantees. If using the analytic Gaussian mechanism, advanced composition would be necessary to reason about the privacy of two-fold adaptive composition [21]. Since these are somewhat loose bounds, the benefit of the analytic calibration would be lost. However, since the time of the competition, much progress has been made on understanding the behavior of the Gaussian mechanism under composition and it is now known that the analytic Gaussian mechanism can be used to calibrate noise for multiple (adaptive) invocations of the Gaussian mechanism [19, 51]. This would give a smaller value of σ than the one shown in Equation (5.1), typically offering an improvement of 10 to 20 percent.

Step 2: Encode Domain. Before running `NIST-MST`, it is necessary to know the data domain Ω. The supplied competitor pack came with a "SPECS" file that contained some domain information. Specifically, it supplied a single positive integer n_i for each attribute i, and the domain for that attribute was assumed to be $\Omega_i^{\text{SPECS}} = \{0, \ldots, n_i - 1\}$. However, because the data is derived from a census source, the domain is very thoroughly documented on the Integrated Public Use Microdata Series (IPUMS) website [1]. IPUMS offers a much finer grained view of the data domain, specifying the exact set of possible values for most attributes. We use Ω_i^{IPUMS} to denote the domain of possible values for attribute i according to IPUMS. Example 4 demonstrates the benefit of using the finer grained IPUMS domain information.

Algorithm 3: Transform data

Input: D (sensitive dataset)

Output: D (transformed sensitive dataset)

(1) Replace `VALUEH` attribute in D using transformation:

(2) Split `INCWAGE` attribute in D into two attributes $\texttt{INCWAGE}_\texttt{A}$ and $\texttt{INCWAGE}_\texttt{B}$ using transformation:[4]

$$\texttt{VALUEH} = \begin{cases} \lfloor \frac{\texttt{VALUEH}}{5} \rfloor & \texttt{VALUEH} \leq 25{,}000 \\ 5001 & \texttt{VALUEH} = 9{,}999{,}998 \\ 5002 & \texttt{VALUEH} = 9{,}999{,}999 \\ 5000 & \text{otherwise} \end{cases}$$

$$\texttt{INCWAGE}_\texttt{A} = \begin{cases} \lfloor \frac{\texttt{INCWAGE}}{100} \rfloor & \texttt{INCWAGE} \leq 5000 \\ 50 & \texttt{INCWAGE} < 9{,}999{,}998 \\ 51 & \texttt{INCWAGE} = 9{,}999{,}998 \end{cases}$$

$$\texttt{INCWAGE}_\texttt{B} = \begin{cases} 0 & \texttt{INCWAGE} \equiv 0 \mod 100 \\ 1 & \texttt{INCWAGE} \equiv 0 \mod 20 \\ 2 & \texttt{INCWAGE} \equiv 0 \mod 50 \\ 3 & \texttt{INCWAGE} \equiv 0 \mod 25 \\ 4 & \texttt{INCWAGE} \equiv 0 \mod 10 \\ 5 & \texttt{INCWAGE} \equiv 0 \mod 5 \\ 6 & \texttt{INCWAGE} \equiv 0 \mod 2 \\ 7 & \text{otherwise} \end{cases}$$

Example 4 (SPECS vs. IPUMS)**.** *From the specs file and the IPUMS website, we see the domain for the* `EDUC` *attribute is* $\Omega_i^{SPECS} = \{0, 1, \ldots, 99\}$ *and* $\Omega_i^{IPUMS} = \{0, 1, \ldots, 11, 99\}$*. Note that* 99 *is a special code that typically corresponds to missing data. While both sources agree that* 99 *is the largest possible value, the IPUMS documentation suggests that values in the range* $12, \ldots, 98$ *are not possible. Using the finer granularity domain from IPUMS reduces the number of possible values for* `EDUC` *from* 100 *to* 13*. This has two important ramifications. First, it will make* `Private-PGM` *more efficient in later steps, since the scalability of that tool depends directly on the domain sizes of the attributes. Second, it will prevent* `NIST-MST` *from inadvertently introducing out-of-domain tuples to the synthetic data which could otherwise occur by adding positive noise to zero counts.*

Often Ω_i^{IPUMS} is a subset of Ω_i^{SPECS}, although this is not always the case. In some cases, IPUMS documents a certain value as being possible that never appeared in the provisional dataset or the supplied specs file. To account for this `NIST-MST` uses the intersection of the two domains, i.e., $\Omega_i = \Omega_i^{\text{SPECS}} \cap \Omega_i^{\text{IPUMS}}$. Table 4 enumerates the attributes in the dataset along with the domain size provided in the specs file, and the compressed domain size derived by `NIST-MST`.

Step 3: Transform Data. In addition to the general domain encoding outlined above, `NIST-MST` gave special attention to two of the attributes with the largest domain: `INCWAGE` and `VALUEH`. Both of these attributes started out with 10 million possible values, and the IPUMS documentation provided limited information on these attributes. Therefore, `NIST-MST` leveraged the provisional dataset to try to identify a domain that captured all or most of the observed values for these attributes. Algorithm 3 shows how these attributes are transformed to reduce the domain size. The intuition behind this pre-processing procedure is to compress the domain, while ensuring the compressed domain still covers all or most of the values observed in the provisional dataset. For example, other than the special codes of 9,999,998 and 9,999,999, 99.2% of records have `VALUEH` that is a multiple of 5 and less

[4]Each condition in the piecewise definition of $\texttt{INCWAGE}_\texttt{A}$ and $\texttt{INCWAGE}_\texttt{B}$ should be interpreted as an "else if" statement rather than an "if" statement, as clearly multiple conditions can be true at the same time

Algorithm 4: Domain compression

Input: log (list of noisy measurements), D (sensitive dataset), Ω (domain)
Output: D (transformed sensitive dataset), Ω (transformed domain)

(1) For each measurement $(\_\_, \tilde{\mu}, \sigma, \{i\})$ in log

(2) Replace values for attribute i in dataset D using transformation:

$$t \leftarrow \begin{cases} t & \tilde{\mu}_t \geq 3\sigma \\ \varnothing & \text{otherwise} \end{cases}$$

(3) Modify domain accordingly, $\Omega_i \leftarrow \{t \mid \tilde{\mu}_t \geq 3\sigma\} \cup \{\varnothing\}$

than or equal to 25,000.[5] This allows us to compress the domain of `VALUEH` to a much more manageable size of 5003 while still covering about 99.7% of the observed values.

`NIST-MST` uses a similar approach to handle `INCWAGE`. Over 99.95% of records in the provisional dataset had an `INCWAGE` value of either 9,999,998 or something in the range $[0, 5000]$. For that reason, it is reasonably safe to truncate values above 5000 without introducing too much bias. This transformation reduces the domain size of `INCWAGE` to 5002, but `NIST-MST` takes things one step further. There are clear periodic patterns in the `INCWAGE` marginal, as the most common values are all multiples of 100. Multiples of 20, 50, and 25 are also common. To exploit this observation, `NIST-MST` splits up `INCWAGE` into two attributes: $\texttt{INCWAGE}_\texttt{A}$ and $\texttt{INCWAGE}_\texttt{B}$, and never measures `INCWAGE` directly, but only indirectly through these two derived attributes. $\texttt{INCWAGE}_\texttt{A}$ is meant to capture the coarse-grained income by discretizing it into width 100 bins, whereas $\texttt{INCWAGE}_\texttt{B}$ is meant to capture the periodicity in the last two digits. These two derived attributes have smaller domains of size 52 and 8, respectively. The exact formulas are given in Algorithm 3.

Step 4: Compress Domain. In step (1), `NIST-MST` was able to greatly reduce the domain size by incorporating information from IPUMS. However, even after this domain encoding, some of the attributes in the data remain fairly sparse. For example, only 17.4% percent of counts in the `VALUEH` marginal exceed 100. In this step, we answer all 1-way marginals, i.e., we pass $\mathcal{C} = \{\{i\} \mid i \in \mathcal{A}\}$ into Algorithm 1. Every marginal is assigned an equal weight of 1, with the exception of $\texttt{INCWAGE}_\texttt{A}$, which is assigned a weight of 2.

After obtaining noisy 1-way marginals, Algorithm 4 is called, which searches for domain elements for which the noisy count fell below the threshold of 3σ. These domain elements were merged into a single "other" domain element, denoted $\varnothing$. Later steps of `NIST-MST` operate over the resulting transformed dataset and domain. This step has two main benefits, similar to the ones from the domain encoding step. First, it improves scalability of `Private-PGM` in later steps by reducing the domain size of the attributes. Second, it ensures that the tuples generated by `NIST-MST` (probably) have attribute values that actually occurred in the dataset.

Step 5: Select Measurements. The next step of `NIST-MST` is to identify a collection of 2- and 3-way marginals that will later be measured. This is one of the most crucial components of `NIST-MST`, because the marginals selected in this step will ultimately determine the marginals that will be preserved in the generated synthetic data. It is important to note that this step selects measurements without using the sensitive dataset, although it does rely heavily on the provisional dataset. This algorithm does take the privacy budget ϵ as

[5] 9,999,998 and 9,999,999 are special codes corresponding to missing values of N/A values.

Algorithm 5: Marginal selection algorithm

Input: $\hat{D}$ (the provisional dataset), ϵ (privacy budget)
Output: $\mathcal{C}$ (a collection of attribute subsets), w (weights for each $C \in \mathcal{C}$)

(1) Construct a complete graph G where vertices are attributes in the dataset, and edge (i, j) is weighted according to the mutual information between attribute i and attribute j in the dataset $\hat{D}$. Add 100 to the edge weights for (`SEX,CITY`), (`SEX,INCWAGE`$_\text{A}$) and (`CITY,INCWAGE`$_\text{A}$).

(2) Identify the maximum spanning tree (MST) of the graph, and for each edge in the tree, add the attribute pair to $\mathcal{C}$. Also add (`SEX,CITY`), (`SEX,INCWAGE`$_\text{A}$), (`CITY,INCWAGE`$_\text{A}$), and (`SEX,CITY,INCWAGE`$_\text{A}$) to $\mathcal{C}$ if they are not already included.

(3) For each pair of adjacent edges $(i, j), (i, k)$ in the MST, compute the marginals $M_{ij}(\hat{D})$, $M_{ik}(\hat{D})$, and $M_{ijk}(\hat{D})$. Use `Private-PGM` to estimate $\tilde{M}_{ijk}$ from M_{ij} and M_{ik}, and record the error in the estimate $E_{ijk} = \left\| M_{ijk} - \tilde{M}_{ijk} \right\|_1$.

(4) For each attribute i, construct a complete graph consisting of nodes that are neighbors of i in the MST, and each edge (j, k) in the new graph is assigned a weight of E_{ijk}. Remove edges whose weight is below a threshold of 0.1, and compute the maximum spanning tree of the resulting graph. For each edge (j, k) in the new MST, add the (j, k) and (i, j, k) marginals to $\mathcal{C}$.

(5) Remove attribute subsets whose marginal is too large, i.e., $C \in \mathcal{C}$ such that $\prod_{i \in C} n_i \geq 10^6$.

(6) Assign weights to selected attribute subsets using formula:

$$w_C \propto \begin{cases} 8 & C = (\texttt{SEX,CITY,INCWAGE}_\text{A}), \epsilon \leq 0.3 \\ 4 & C = (\texttt{SEX,CITY,INCWAGE}_\text{A}), \epsilon \geq 4.0 \\ 6 & C = (\texttt{SEX,CITY,INCWAGE}_\text{A}), 0.3 < \epsilon < 4.0 \\ 2 & C \in \{(\texttt{SEX,CITY}), (\texttt{SEX,INCWAGE}_\text{A}), (\texttt{CITY,INCWAGE}_\text{A})\} \\ 1 & \text{otherwise} \end{cases}$$

input, but it does not "consume" it — it only uses it to determine weights to assign to each selected marginal.

Algorithm 5 shows how `NIST-MST` selects 2- and 3-way marginals for measurement. This algorithm is inspired by a similar approach used by two other mechanisms for differentially private synthetic data [14,66]. It combines one principled step, which is to find the maximum spanning tree (MST) on the graph where edge weights correspond to mutual information between two attributes, with some additional heuristics to ensure that certain important attribute pairs are selected, and more heuristics to select some triples while keeping the graph tree-like. A reader familiar with graphical models with recognize the MST step as the famous Chow-Liu algorithm for structure learning in a graphical model [15]. Intuitively, highly correlated marginals should be measured because attributes that are independent can trivially be preserved without direct measurement.

Figure 3 shows the marginals selected by this algorithm in graphical format. Each edge in the graph represents a pair of attributes whose marginal will be measured by `NIST-MST`, and each triangle in the graph represents a triple of attributes whose marginal will be measured by `NIST-MST`. The structure of this measurement graph also corresponds to the structure of the graphical model used by `Private-PGM`. The tree-like structure of the graph will ultimately allow `Private-PGM` to run efficiently. The green subgraph corresponds to the (`SEX,CITY,INCWAGE`$_\text{A}$) clique, the black edges form a maximum spanning tree of the underlying correlation graph, and the dotted red edges are additional edges that enhance the expressive capacity of the model while retaining the tree-like structure.

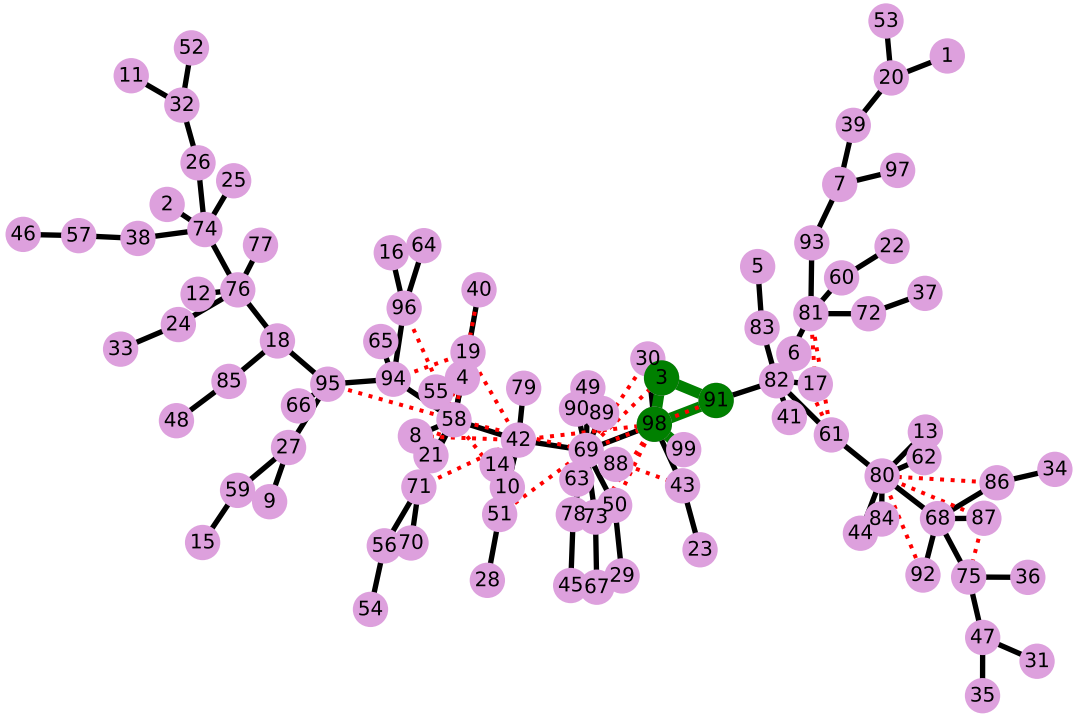

Figure 3. A graphical depiction of the 2- and 3-way marginals selected by `NIST-MST`. Nodes correspond to attributes in the dataset, and edges correspond to marginals selected by `NIST-MST`. Nodes are labeled by the 2-digit code for the attribute given in Table 4. Black edges form a *maximum spanning tree* of the underlying correlation graph. Green nodes and edges correspond to the special (`SEX, CITY, INCWAGE`$_{\texttt{A}}$) attributes. Dotted red edges identify the extra marginals chosen to improve expressive capacity of the model while maintaining tractability of `Private-PGM`. All dotted red edges form a triangle, and for each of those `NIST-MST` also included the 3-way marginal corresponding to the three nodes that make up the triangle in the list of selected marginals.

Step 6: Measure Marginals. The next step of `NIST-MST` is to measure the marginals selected in the previous step with Algorithm 1. The result is a collection of noisy measurements contained within a measurement log, and suitable for post-processing with `Private-PGM`.

Step 7: Synthesize data. The next step of `NIST-MST` is to combine the measurement logs from Steps 4 and 6 and pass them to `Private-PGM`, which returns a synthetic dataset whose marginals approximately match those in the measurement log. Because the measurements in Steps 4 and 6 were made on the uncompressed and compressed domains, respectively, the measurements from Step 4 had to be re-expressed over the compressed domain.

Step 8: Reverse Transformation. The final step of `NIST-MST` is to reverse the transformations made in Steps 4 and 3, to bring the data back to the original domain. For Step 4, this requires evenly distributing any instances of $\varnothing$ among the original domain elements mapped to it. For Step 3, this requires modifying `VALUEH` and combining `INCWAGE`$_{\texttt{A}}$ and `INCWAGE`$_{\texttt{B}}$ back into a single attribute. The details are given in Algorithm 9 of the appendix.

Algorithm 6: Differentially private measurement selection

Input: D (sensitive dataset), log (measurements of 1-way marginals), ρ (privacy parameter), $\mathcal{C}$ (initial set of (i,j) pairs to measure; empty by default)
Output: $\mathcal{C}$ (final set of (i,j) pairs to measure)

(1) Use `Private-PGM` to estimate all 2-way marginals $\bar{M}_{ij}$ from log
(2) Compute L_1 error between estimated 2-way marginal and actual 2-way marginal for all i,j: $q_{ij}(D) = \left\|M_{ij}(D) - \bar{M}_{ij}\right\|_1$ (this is a sensitivity 1 quantity)
(3) Let $G = (\mathcal{A}, \mathcal{C})$ be the graph where attributes are vertices and edges are pairs of attributes
(4) Let r be the number of connected components in G [6]
(5) Let $\epsilon = \sqrt{\frac{8\rho}{r-1}}$
(6) Repeat $r-1$ times
(7) Let S be the set of all attribute pairs (i,j), where i and j are in different connected components of G
(8) Select attribute pair (i,j) by running the `exponential mechanism` with quality score function q_{ij} on set S and privacy parameter ϵ.
(9) Add attribute pair (i,j) to $\mathcal{C}$

6. Extensions

One limitation with `NIST-MST` is that it is highly tailored to the setting of the NIST competition, and crucially relies on the existence of a public provisional dataset that can be used to select marginals. In more general settings, we will not always have access to a provisional dataset that follows a similar distribution as the sensitive data. For that reason, we propose `MST`, a general purpose mechanism that is inspired by the `NIST-MST` mechanism, but doesn't rely on the existence of provisional data. The basic mechanism is the same as `NIST-MST` outlined in Algorithm 2, with a couple minor exceptions. First, the preprocessing transformations and corresponding reverse transformations are not done — those were specific to the U.S. Census dataset used in the competition and not generally applicable beyond that setting. Second, the measurement selection step, which previously relied on a provisional dataset to select correlated marginals, is replaced by a differentially-private version that uses the sensitive dataset. `MST` devotes $\frac{1}{3}$ of the RDP budget towards measurement selection, and uses the remaining $\frac{2}{3}$ of the RDP budget for measuring the marginals. Privacy of `MST` follows by adaptive composition Proposition 3. For completeness, this calculation is given in the appendix.

The measurement selection algorithm is shown in Algorithm 6. Just like Algorithm 5, this algorithm tries to find a collection of attribute pairs that form a maximum spanning tree of an underlying correlation graph. However, as it uses the sensitive dataset, it must do this in a differentially private way. To achieve this, we first use a low-sensitivity approximation of the mutual information for assigning edge weights. We assume that we already measured all 1-way marginals, so we can get reasonable estimates of 2-way marginals by invoking `Private-PGM`.[7] The edge weights are then computed as the L_1 distance between the true 2-way marginal and the estimated one (a sensitivity 1 quantity). After computing the edge weights, Algorithm 6 can be seen as a differentially private version of Kruskal's algorithm [34] for computing a maximum spanning tree. It consists of $d-1$ steps (the number of edges in a

[6] If $\mathcal{C}$ is empty, this is just the number of attributes d.

[7] In this simple case, `Private-PGM` estimates 2-way marginals under an independence assumption, which could alternatively be achieved by multiplying the (noisy) one-way marginals together.

spanning tree), and in each step, it adds a highly weighted edge that connects two different connected components. In Kruskal's algorithm, the highest weighted edge is chosen, but this would not be differentially private. We instead invoke the `exponential mechanism` to select a highly weighted edge in a differentially private way. In principle, we could apply any private selection algorithm here, including report-noisy-max [21] and the recently developed permute-and-flip mechanism [43]. While permute-and-flip is known to dominate the exponential mechanism under ϵ-DP [43], the exponential mechanism enjoys a tighter privacy analysis under Rényi-DP [12].

The result of this algorithm is a collection of attribute pairs that will be measured by `MST`. Algorithm 6 has an optional argument, $\mathcal{C}$, which is an initial set of attribute pairs to measure. If this is supplied, the algorithm will always include those in the result, and then constructs a maximum spanning tree around them. This enables some extra flexibility that may be beneficial in certain settings where some marginals are more important than others, and need to be preserved even if they are not the most highly correlated. For many applications, $\mathcal{C}$ can just be empty. In the context of the competition, this feature is useful because the marginals relating `SEX`, `CITY`, and $\texttt{INCWAGE}_\texttt{A}$ are very important (since their accuracy determines $\frac{1}{3}$ of the final score).

Theorem 3 (Privacy of Algorithm 6)**.** *Algorithm 6 is $(\alpha, \alpha\rho)$-RDP for all $\alpha \geq 1$.*

Proof. Step 4*a* is $(\alpha, \alpha\frac{1}{8}\epsilon^2)$-RDP by Proposition 2. Substituting $\epsilon = \sqrt{\frac{8\rho}{r-1}}$, we see that it is equivalent to $(\alpha, \alpha\frac{\rho}{r-1})$-RDP. It is called $r-1$ times, so the entire mechanism is $(\alpha, \alpha\rho)$-RDP by Proposition 3. □

7. Experiments

In this section we discuss the experimental evaluation carried out by the contest organizers, and how `NIST-MST` compared to the submissions from other teams. We offer our own insights into the numbers and explanations for the differences between mechanisms.

Evaluations were carried out on U.S. Census data for two different states: Arizona and Vermont. These datasets had 293,999 and 211,228 records, respectively. Scores were calculated separately for each of the three evaluation metrics described in Section 3.3. The final score was calculated by averaging the scores for each metric, each value of ϵ, and each of the two datasets. In Table 5 we show the score breakdown by metric (averaged over Arizona and Vermont) for the top five submitted algorithms. We also evaluated our new mechanism (`MST`) and included it as an extra row (highlighted), even though it was not evaluated by the contest organizers at the time of the competition. This was possible because the final evaluation datasets were released after the competition, and the script used to evaluate the synthetic data was provided as part of the competitor pack.

`MST` was described in Section 6. We instantiate it with $\mathcal{C} = \{(\texttt{SEX},\texttt{CITY}), (\texttt{SEX},\texttt{INCWAGE}_\texttt{A}), (\texttt{CITY},\texttt{INCWAGE}_\texttt{A})\}$, and add $(\texttt{SEX},\texttt{CITY},\texttt{INCWAGE}_\texttt{A})$ to the returned result as well. These marginals are weighted using the same formula as `NIST-MST` (see Algorithm 5).

Among the contest submissions, `NIST-MST` consistently performed the best, for most metrics and values of ϵ. Compared to DPSyn, it did a much better job at answering 3-way marginals and high order conjunctions, but performed slightly worse at handling the income inequality metric. Compared to every other mechanism, `NIST-MST` did better on every metric.

ϵ	Team	3-way marginals	High order conjunctions	Income inequality	Overall
0.3	RMcKenna (NIST-MST)	0.12	0.17	0.10	0.13
0.3	MST	**0.11**	**0.16**	0.10	**0.12**
0.3	DPSyn	0.15	0.31	**0.07**	0.18
0.3	PrivBayes	0.19	0.29	0.18	0.22
0.3	Gardn999	0.21	0.32	0.25	0.26
0.3	UCLANESL	0.57	0.72	0.22	0.50
1.0	RMcKenna (NIST-MST)	**0.09**	**0.15**	**0.04**	**0.09**
1.0	MST	**0.09**	**0.15**	0.05	0.10
1.0	DPSyn	0.11	0.23	0.05	0.13
1.0	PrivBayes	0.17	0.26	0.09	0.17
1.0	Gardn999	0.18	0.28	0.22	0.23
1.0	UCLANESL	0.42	0.53	0.28	0.41
8.0	RMcKenna (NIST-MST)	**0.07**	**0.14**	0.04	**0.08**
8.0	MST	0.08	**0.14**	0.05	0.09
8.0	DPSyn	0.09	0.20	**0.02**	0.10
8.0	PrivBayes	0.13	0.23	0.09	0.15
8.0	Gardn999	0.17	0.26	0.24	0.22
8.0	UCLANESL	0.35	0.41	0.25	0.34

Table 5. Evaluation of NIST-MST and other mechanisms from competing teams, broken down by the three scoring metrics: 3-way marginals, high-order conjunctions, and income inequality. MST is also shown for comparison, even though that mechanism was not submitted during the competition. If it was submitted instead of NIST-MST, it would have placed first overall.

Generally speaking, NIST-MST and DPSyn (and to a lesser extent PrivBayes) seemed to be the only mechanisms that scored well on the income inequality metric, which is surprising given that it was the simplest metric and only required preserving one 3-way marginal accurately. This raises an important point: mechanisms that understood the evaluation criteria well, and designed their mechanisms around it, generally performed better than mechanisms that just tried to generate good synthetic data without thinking about how its utility would be evaluated. NIST-MST and DPSyn did a good job of designing their mechanism for the task at hand, which was an important contributing factor for why they outperformed the other solutions.

While NIST-MST relied heavily on the provisional dataset for measurement selection, the more general variant MST still performs well, without explicitly relying on the provisional data. In fact, MST would have won first place if it was submitted instead of NIST-MST at the time of the competition; it was only slightly worse than NIST-MST and still better than the other submissions. The difference in performance between MST and NIST-MST was at most 0.01 for every metric and privacy budget evaluated.

At $\epsilon = 0.3$, MST even achieved a smaller overall score than NIST-MST. NIST-MST measures both 2- and 3-way marginals, while MST only measures 2-way marginals. Since the privacy budget is relatively small, it makes sense that fewer measurements would work better here.

A promising direction for future work is to adaptively select the number of marginals to measure based on the privacy budget and the amount of data available.

For a much more comprehensive evaluation of these mechanisms, we refer the reader to [10], which goes well beyond the metrics used in the competition to evaluate these mechanisms. This work is also a useful resource that summarizes each mechanism at a high level. Their results generally show that `NIST-MST` was the best performing mechanism for many of the additional evaluation metrics not used as part of the official scoring criteria. This suggests that `NIST-MST` produces the most generally useful synthetic data among the submitted mechanisms.

8. Related Work

Differentially private synthetic data has been an active area of research for several years. One of the earliest mechanisms proposed for this task was the "small database mechanism" [21], which instantiates the exponential mechanism over a set of small databases to select one that is statistically similar to the true data. Unfortunately, this mechanism is not able to run in practical settings, as it requires enumerating all possible datasets of a fixed sizes, resulting in a combinatorial explosion even for small dataset sizes (especially if the domain size is large).

In the competition, the top four submissions all followed the same basic template outlined in Figure 1: they selected and measured a collection of marginals, and then used those to estimate synthetic data. Moreover this approach has been applied more generally in the literature [9, 14, 66, 71]. One key difference that sets our approach apart is `Private-PGM`. While [14, 66] do leverage graphical models to generate synthetic data, their approach is limited in how it makes use of the noisy measurements: they more or less treat the noisy marginals as the true marginals (with some lightweight post-processing), whereas `Private-PGM` makes use of all available measurements to resolve inconsistencies and boost utility in a principled manner. An alternative method for resolving inconsistencies and generating synthetic data from noisy marginals is proposed in [71]. This method does not construct an intermediate representation of the data distribution as `Private-PGM` does. As a result, their consistency resolution step only ensures *local consistency* (i.e., that all marginals internally agree on the common marginals), and does not satisfy the stronger notion of *global consistency* (i.e., that there is a data distribution that has all stated marginals), as `Private-PGM` does.

Another popular class of approaches for differentially private synthetic data is based on generative adversarial networks (GANs) [27]. Several differentially private GANs have been proposed for the purpose of generating synthetic data [3, 8, 23, 32, 52–54, 59, 70]. In fact, two teams in the NIST competition adopted a GAN-based approach (UCLANESL in Table 5, and one other team that did not place in the top five), however, their scores were not generally competitive with the other approaches that used the marginal-based framework. GANs are notoriously hard to train in practice [49] and when differential privacy constraints are enforced, it is even more difficult. It typically requires running an algorithm like DP-SGD [2] to train, and if it fails to converge (which is common) the privacy budget used for training is essentially wasted. Setting the right hyper-parameters is also a major challenge for this approach.

Another important and related problem is how to evaluate the quality of synthetic data [5, 10, 30, 50]. Beyond the metrics used in the NIST competition, one alternative is

pMSE, which is a general measure of distributional similarity [50]. Another alternative measure is machine learning efficacy, or how well the synthetic data supports machine learning applications [30]. A number of other measures for evaluating synthetic data can be found in [10]. We believe that there may be no universal answer to this question: it should ultimately depend on the data and its use cases. In general, it would be nice to have a mechanism that can automatically adapt to an analyst-provided *workload*, and generate synthetic data that provides high utility on the queries and tasks in that workload. Several workload-adaptive mechanisms exist, but they are generally restricted to settings where the full high-dimensional histogram can be explicitly materialized in vector form, and are thus unable to scale to high-dimensional domains [21, 24, 28, 36, 41]. When combined with `Private-PGM`, the scalability (and utility) of some of these mechanisms can be improved, however [42].

9. Conclusions

In this paper, we described `NIST-MST`, the winning mechanism from the NIST differential privacy synthetic data competition, and `MST` a new mechanism inspired by `NIST-MST` that works almost as well, without relying on public provisional data. While these mechanisms are state-of-the-art, the problem of differentially private synthetic data is far from solved. Nevertheless, we believe our basic framework centered around `Private-PGM` can serve as a core component of new mechanisms for this task. `Private-PGM` allows the mechanism designer to focus on *what to measure*, rather than *how to post-process* those measurements to get synthetic data while extracting the most utility from them. In fact, in the final round of the recently completed follow-up challenge, the NIST 2020 Temporal Map Challenge, *both* the first and second place teams used `Private-PGM` for post-processing, with each team developing novel techniques for measurement selection.

References

[1] Ipums usa. https://usa.ipums.org/usa/.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[3] N. C. Abay, Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and L. Sweeney. Privacy preserving synthetic data release using deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 510–526. Springer, 2018.

[4] G. Ács, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In *ICDM*, pages 1–10, 2012.

[5] C. Arnold and M. Neunhoeffer. Really useful synthetic data–a framework to evaluate the quality of differentially private synthetic data. *arXiv preprint arXiv:2004.07740*, 2020.

[6] H. J. Asghar, M. Ding, T. Rakotoarivelo, S. Mrabet, and M. A. Kaafar. Differentially private release of high-dimensional datasets using the gaussian copula. *arXiv preprint arXiv:1902.01499*, 2019.

[7] B. Balle and Y.-X. Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403, 2018.

[8] B. K. Beaulieu-Jones, Z. S. Wu, C. Williams, R. Lee, S. P. Bhavnani, J. B. Byrd, and C. S. Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122, 2019.

[9] V. Bindschaedler, R. Shokri, and C. A. Gunter. Plausible deniability for privacy-preserving data synthesis. *Proceedings of the VLDB Endowment*, 10(5), 2017.

[10] C. M. Bowen and J. Snoke. Comparative study of differentially private synthetic data algorithms from the nist pscr differential privacy synthetic data challenge. *Journal of Privacy and Confidentiality*, 11(1), 2021.

[11] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[12] M. Cesar and R. Rogers. Bounding, concentrating, and truncating: Unifying privacy loss composition for data analytics. In *Algorithmic Learning Theory*, pages 421–457. PMLR, 2021.

[13] A.-S. Charest. How can we analyze differentially-private synthetic datasets? *Journal of Privacy and Confidentiality*, 2(2), 2011.

[14] R. Chen, Q. Xiao, Y. Zhang, and J. Xu. Differentially private high-dimensional data publication via sampling-based inference. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 129–138. ACM, 2015.

[15] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

[16] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *Data engineering (ICDE), 2012 IEEE 28th international conference on*, pages 20–31. IEEE, 2012.

[17] B. Ding, M. Winslett, J. Han, and Z. Li. Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 217–228. ACM, 2011.

[18] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, 2003.

[19] J. Dong, A. Roth, and W. Su. Gaussian differential privacy. *Journal of the Royal Statistical Society*, 2021.

[20] C. Dwork, F. M. K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[21] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Found. and Trends in Theoretical Computer Science, 2014.

[22] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 521–532. IEEE, 2018.

[23] L. Frigerio, A. S. de Oliveira, L. Gomez, and P. Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 151–164. Springer, 2019.
[24] M. Gaboardi, E. J. G. Arias, J. Hsu, A. Roth, and Z. S. Wu. Dual query: Practical private query release for high dimensional data. In *International Conference on Machine Learning*, pages 1170–1178. PMLR, 2014.
[25] S. Garfinkel, J. M. Abowd, and C. Martindale. Understanding database reconstruction attacks on public data. *Communications of the ACM*, 62(3):46–53, 2019.
[26] C. Ge, S. Mohapatra, X. He, and I. F. Ilyas. Kamino: Constraint-aware differentially private data synthesis. *Proceedings of the VLDB Endowment*, 14(3), 2020.
[27] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.
[28] M. Hardt, K. Ligett, and F. Mcsherry. A simple and practical algorithm for differentially private data release. *Advances in Neural Information Processing Systems*, 25:2339–2347, 2012.
[29] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1-2):1021–1032, 2010.
[30] M. Hittmeir, A. Ekelhart, and R. Mayer. On the utility of synthetic data: an empirical evaluation on machine learning tasks. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–6, 2019.
[31] Z. Huang, R. McKenna, G. Bissias, G. Miklau, M. Hay, and A. Machanavajjhala. Psyndb: accurate and accessible private data generation. *Proceedings of the VLDB Endowment (Demo)*, 12(12):1918–1921, 2019.
[32] J. Jordon, J. Yoon, and M. Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2018.
[33] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
[34] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.
[35] C. Li, M. Hay, G. Miklau, and Y. Wang. A data-and workload-aware algorithm for range queries under differential privacy. *PVLDB*, 7(5):341–352, 2014.
[36] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 123–134. ACM, 2010.
[37] C. Li and G. Miklau. An adaptive mechanism for accurate query answering under differential privacy. *PVLDB*, 5(6):514–525, 2012.
[38] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB Journal*, 24(6):757–781, 2015.
[39] H. Li, L. Xiong, and X. Jiang. Differentially private synthesization of multi-dimensional data using copula functions. In *Advances in database technology: proceedings. International Conference on Extending Database Technology*, volume 2014, page 475. NIH Public Access, 2014.
[40] F. Liu. Model-based differentially private data synthesis. *arXiv preprint arXiv:1606.08052*, 2016.
[41] R. McKenna, G. Miklau, M. Hay, and A. Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Proceedings of the VLDB Endowment*, 11(10):1206–1219, 2018.
[42] R. McKenna, D. Sheldon, and G. Miklau. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pages 4435–4444, 2019.
[43] R. McKenna and D. R. Sheldon. Permute-and-flip: A new mechanism for differentially private selection. *Advances in Neural Information Processing Systems*, 33, 2020.
[44] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, 2007.
[45] I. Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.
[46] W. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *Intl. Conference on Data Engineering (ICDE)*, pages 757–768. IEEE, 2013.
[47] W. Qardaji, W. Yang, and N. Li. Understanding hierarchical methods for differentially private histograms. *PVLDB*, 6(14):1954–1965, 2013.

[48] W. Qardaji, W. Yang, and N. Li. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1435–1446. ACM, 2014.
[49] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NIPS*, 2016.
[50] J. Snoke, G. M. Raab, B. Nowok, C. Dibben, A. Slavkovic, et al. General and specific utility measures for synthetic data. *Journal of the Royal Statistical Society Series A*, 181(3):663–688, 2018.
[51] D. M. Sommer, S. Meiser, and E. Mohammadi. Privacy loss classes: The central limit theorem in differential privacy. *Proceedings on privacy enhancing technologies*, 2019(2):245–269, 2019.
[52] U. Tantipongpipat, C. Waites, D. Boob, A. A. Siva, and R. Cummings. Differentially private mixed-type data generation for unsupervised learning. *arXiv preprint arXiv:1912.03250*, 2019.
[53] A. Torfi, E. A. Fox, and C. K. Reddy. Differentially private synthetic medical data generation using convolutional gans. *arXiv preprint arXiv:2012.11774*, 2020.
[54] R. Torkzadehmahani, P. Kairouz, and B. Paten. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.
[55] G. Vietri, G. Tian, M. Bun, T. Steinke, and S. Wu. New oracle-efficient algorithms for private synthetic data release. In *International Conference on Machine Learning*, pages 9765–9774. PMLR, 2020.
[56] www.nist.gov. 2018 differential privacy synthetic data challenge. https://www.nist.gov/ctl/pscr/open-innovation-prize-challenges/past-prize-challenges/2018-differential-privacy-synthetic, 2018.
[57] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1200–1214, 2011.
[58] Y. Xiao, L. Xiong, L. Fan, S. Goryczka, and H. Li. DPCube: Differentially private histogram release through multidimensional partitioning. *Transactions of Data Privacy*, 7(3), 2014.
[59] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.
[60] C. Xu, J. Ren, Y. Zhang, Z. Qin, and K. Ren. Dppro: Differentially private high-dimensional data release via random projection. *IEEE Transactions on Information Forensics and Security*, 12(12):3081–3093, 2017.
[61] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu. Differentially private histogram publication. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 32–43, 2012.
[62] J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu, and M. Winslett. Differentially private histogram publication. *The VLDB Journal*, pages 1–26, 2013.
[63] G. Yaroslavtsev, G. Cormode, C. M. Procopiuc, and D. Srivastava. Accurate and efficient private release of datacubes and contingency tables. In *ICDE*, 2013.
[64] G. Yuan, Y. Yang, Z. Zhang, and Z. Hao. Convex optimization for linear query processing under approximate differential privacy. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2005–2014. ACM, 2016.
[65] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: optimizing batch queries under differential privacy. *PVLDB*, 5(11):1352–1363, 2012.
[66] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):1–41, 2017.
[67] J. Zhang, X. Xiao, and X. Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *SIGMOD*, 2016.
[68] W. Zhang, J. Zhao, F. Wei, and Y. Chen. Differentially private high-dimensional data publication via markov network. *EAI Endorsed Transactions on Security and Safety*, 6(19), 2019.
[69] X. Zhang, R. Chen, J. Xu, X. Meng, and Y. Xie. Towards accurate histogram publication under differential privacy. In *SDM*, 2014.
[70] X. Zhang, S. Ji, and T. Wang. Differentially private releasing via deep generative model (technical report). *arXiv preprint arXiv:1801.01594*, 2018.
[71] Z. Zhang, T. Wang, N. Li, J. Honorio, M. Backes, S. He, J. Chen, and Y. Zhang. Privsyn: Differentially private data synthesis. *arXiv preprint arXiv:2012.15128*, 2020.
[72] J. Zhao, T. Wang, T. Bai, K.-Y. Lam, Z. Xu, S. Shi, X. Ren, X. Yang, Y. Liu, and H. Yu. Reviewing and improving the gaussian mechanism for differential privacy. *arXiv preprint arXiv:1911.12060*, 2019.

Appendix A. Supplementary Material

Noise Calibration for MST. We begin by calculating our total RDP budget from (ϵ, δ) by invoking Proposition 4. In particular, we find the largest value of ρ such that $(\alpha, \alpha\rho)$-RDP implies (ϵ, δ)-DP by Proposition 4. We accomplish this numerically. We will now divide our RDP budget "ρ" equally among the three steps (4) and (5) and (6) of `MST`. To achieve $\frac{\rho}{3}$-RDP in steps (4) and (6), it suffices to set $\sigma = \sqrt{\frac{3}{2\rho}}$ by Proposition 1. To achieve $\frac{\rho}{3}$-RDP in step (5), we simply call Algorithm 6 with privacy parameter $\frac{\rho}{3}$. The correctness of this

2/3/2021 jpc-noise calibration

In [1]:
```
from sympy import *

α,ε,δ,σ = symbols("alpha epsilon delta sigma")
expr = α/σ**2 + log(1/δ)/(α - 1)
expr = expr.subs(α, 1 + σ*sqrt(log(1/δ)))
expr = expr.subs(σ,(sqrt(log(1/δ))+sqrt(log(1/δ)+ε))/ε)
expr.simplify()
```

Out[1]: ϵ

In [ ]:

Figure 4. `sympy` code to augment the proof of Theorem 2

Additional Experiment on MST vs. NIST-MST. In Section 7, we evaluated `MST` in the context of the challenge test suite and evaluation metrics, and found that it performed comparably to `NIST-MST`. In this section, we compare the quality of the marginals selected from public data (Algorithm 5) and from the sensitive data (Algorithm 6) using various privacy budgets. Specifically, we logged the marginals selected by `MST`, and compared them to the publicly chosen marginals used in `NIST-MST` (ignoring the extra 2- and 3-way marginals selected in step 4) using the mutual information criteria described in Algorithm 5. We also show the scores that would be achieved by the best marginals (i.e., the true maximum spanning tree), as well as the scores that would be achieved by a random spanning tree. As shown in the table below, both the publicly chosen marginals and the privately chosen marginals nearly match the score acheived by the best marginals, for both the Arizona and Vermont datasets. The publicly chosen marginals are slightly better at $\epsilon = 0.3$ and $\epsilon = 1.0$, and slightly worse at $\epsilon = 8.0$. Both variants are much better than the random baseline.

Selected Marginals	Arizona	Vermont
Best	73.64	63.32
Public (Algorithm 5)	72.81	62.79
$\epsilon = 0.3$ (Algorithm 6)	70.53	61.15
$\epsilon = 1.0$ (Algorithm 6)	72.27	62.35
$\epsilon = 8.0$ (Algorithm 6)	73.05	63.09
Random	8.55	6.95

Algorithm 7: Synthetic column

Input: μ (vector of fractional counts), n (total number of records to generate)
Output: column (synthetic column of data)

(1) Generate $\lfloor \mu_t \rfloor$ items with value t and add to column for each t in domain
(2) Calculate remainders, $p_t = \mu_t - \lfloor \mu_t \rfloor$
(3) Sample $n - \sum_t \lfloor \mu_t \rfloor$ items (without replacement) from distribution proportional to p_t, and add to column
(4) Shuffle values in column

Algorithm 8: Synthetic data generation

Input: graphical model
Output: dataset (synthetic dataset)

(1) Initialize the set of processed attributes to the empty set
(2) For each attribute i
(3) Let C be the set of all neighbors of i in the graphical model, intersected with the set of processed attributes
(4) Group data by C, and for each group in C
(5) Calculate μ from the graphical model, the vector of fractional counts for every possible value of attribute i, for the given group of other attributes
(6) Generate synthetic column for this group using Algorithm 7
(7) Add this partial column to the grouped rows in the dataset
(8) Add i to the set of processed attributes

Algorithm 9: Reverse transformation of Algorithm 3

Input: D (sensitive dataset)
Output: D (transformed sensitive dataset)

(1) Compute DIGITS from $\texttt{INCWAGE}_\texttt{B}$ using Algorithm 10
(2) Replace VALUEH and INCWAGE attributes in D using transformations:

$$\texttt{VALUEH} = \begin{cases} 5 \cdot \texttt{VALUEH} & \texttt{VALUEH} \leq 5000 \\ 9{,}999{,}998 & \texttt{VALUEH} = 5001 \\ 9{,}999{,}999 & \texttt{VALUEH} = 5002 \end{cases} \qquad \texttt{INCWAGE} = \begin{cases} 100 \cdot \texttt{INCWAGE}_\texttt{A} + \texttt{DIGITS} & \texttt{INCWAGE}_\texttt{A} \leq 50 \\ 9{,}999{,}998 & \texttt{INCWAGE}_\texttt{A} = 51 \end{cases}$$

Algorithm 10: Convert $\texttt{INCWAGE}_\texttt{B}$ to DIGITS

Input: k (a value for $\texttt{INCWAGE}_\texttt{B}$)
Output: l (a value for DIGITS)

(1) Let $L = \{0, \ldots, 99\}$
(2) Let $m = [100, 20, 50, 25, 10, 5, 2, 1]$
(3) *For* $i = 0, \ldots, k$
(4) Let $L_i = \{l \in L \mid l \equiv 0 \mod m_i\}$
(5) Let $L = L \setminus L_i$
(6) Sample l uniformly from L_k