Journal of Privacy and Confidentiality
Vol. 10 (2) 2020

Submitted Sept 2018
Published June 2020

DIFFERENTIALLY PRIVATE RELEASE OF HIGH-DIMENSIONAL DATASETS USING THE GAUSSIAN COPULA

HASSAN JAMEEL ASGHAR, M. DING, T. RAKOTOARIVELO, S. MRABET, AND D. KAAFAR

Macquarie University & Data61, CSIRO
e-mail address: hassan.asghar@mq.edu.au

Data61, CSIRO
e-mail address: ming.ding@data61.csiro.au

Data61, CSIRO
e-mail address: thierry.rakotoarivelo@data61.csiro.au

Data61, CSIRO
e-mail address: sirine.mrabet@data61.csiro.au

Macquarie University & Data61, CSIRO
e-mail address: dali.kaafar@mq.edu.au

Abstract. We propose a generic mechanism to efficiently release differentially private synthetic versions of high-dimensional datasets with high utility. The core technique in our mechanism is the use of copulas, which are functions representing dependencies among random variables with a multivariate distribution. Specifically, we use the Gaussian copula to define dependencies of attributes in the input dataset, whose rows are modelled as samples from an unknown multivariate distribution, and then sample synthetic records through this copula. Despite the inherently numerical nature of Gaussian correlations we construct a method that is applicable to both numerical and categorical attributes alike. Our mechanism is efficient in that it only takes time proportional to the square of the number of attributes in the dataset. We propose a differentially private way of constructing the Gaussian copula without compromising computational efficiency. Through experiments on three real-world datasets, we show that we can obtain highly accurate answers to the set of all one-way marginal, and two-and three-way positive conjunction queries, with 99% of the query answers having absolute (fractional) error rates between 0.01 to 3%. Furthermore, for a majority of two-way and three-way queries, we outperform independent noise addition through the well-known Laplace and Gaussian mechanisms. In terms of computational time we demonstrate that our mechanism can output synthetic datasets in around 6 minutes 47 seconds on average with an input dataset of about 200 binary attributes and more than 32,000 rows, and about 2 hours 30 mins to execute a much larger dataset of about 700 binary attributes and more than 5 million rows. To further demonstrate scalability, we ran the mechanism on larger (artificial) datasets with 1,000 and 2,000 binary attributes (and 5 million rows) obtaining synthetic outputs in approximately 6 and 19 hours, respectively. These are highly feasible times for synthetic datasets, which are one-off releases.

Key words and phrases: differential privacy; synthetic data; high dimensional; copula.

JPrivacy Confidentiality
www.journalprivacyconfidentiality.org
DOI:10.29012/jpc.686

1. Introduction

There is an ever increasing demand to release and share datasets owing to its potential benefits over controlled access. For instance, once data is released, data custodian(s) need not worry about access controls and continual support. From a usability perspective, data release is more convenient for users (expert analysts and novices alike) as compared to access through a restricted interface. Despite its appeal, sharing datasets, especially when they contain sensitive information about individuals, has privacy implications which have been well documented. Current practice, therefore, suggests privacy-preserving release of datasets. Ad hoc techniques such as de-identification, which mainly rely on properties of datasets and assumptions on what background information is available, have failed to guarantee privacy [Narayanan and Shmatikov, 2008, Ohm, 2009]. Part of the reason for the failure is the lack of a robust definition of privacy underpinning these techniques.

This gave rise to the definition of differential privacy [Dwork et al., 2006b]. Differential privacy ties the privacy property to the process or algorithm (instead of the dataset) and, informally, requires that any output of the algorithm be almost equally likely even if any individual's data is added or removed from the dataset. A series of algorithms have since been proposed to release differentially private datasets, often termed as synthetic datasets. Simultaneously, there are results indicating that producing synthetic datasets which accurately answer a large number of queries is computationally infeasible, i.e., taking time exponential in the number of attributes (dimension of the dataset) [Ullman and Vadhan, 2011]. This is a serious roadblock as real-world datasets are often high-dimensional. However, infeasibility results from Ullman and Vadhan [2011] and Ullman [2013] are generic, targeting *provable* utility for *any* input data distribution. It may well be the case that a large number of real-world datasets follow constrained distributions which would make it computationally easier to output differentially private synthetic datasets that can accurately answer a larger number of queries. Several recent works indicate the plausibility of this approach claiming good utility in practice [Zhang et al., 2014, Li et al., 2014]. Ours is a continuation of this line of work.

In this paper, we present a *generic* mechanism to *efficiently* generate differentially private synthetic versions of high dimensional datasets with *good utility*. By efficiency, we mean that our mechanism can output a synthetic dataset in time $O(m^2n)$, where m is the total number of attributes in the dataset and n the total number of rows. Recall that impossibility results [Ullman and Vadhan, 2011] suggest that algorithms for accurately answering a large number of queries are (roughly) expected to run in time $\text{poly}(2^m, n)$. Thus, our method is scalable for high dimensional datasets, i.e., having a large m. In terms of utility, our generated synthetic dataset is designed to give well approximated answers to all one-way marginal and two-way positive conjunction queries. One-way marginal queries return the number of rows in the dataset exhibiting a given value x or its negation $\overline{x}$ (not x) under any attribute X. Similarly, two-way positive conjunction queries return the number of rows that exhibit any pair of values (x, y) under an attribute pair (X, Y). This forms a subset of all two-way margins; the full set also includes negations of values, e.g., rows that satisfy $(x, \overline{y})$.[1]

[1] The answers to all two-way marginals can also be obtained although accuracy degrades by a factor of two as compared to the accuracy of answers to one-way marginals and two-way positive conjunctions. See Section 3.3.1 for the reason behind this.

While this may seem like a small subset of queries, there are results showing that even algorithms that generate synthetic datasets to (accurately) answer all two-way marginal queries are expected to be computationally inefficient [Ullman and Vadhan, 2011, Ullman, 2013]. Furthermore, we show that our mechanism provides good utility for other queries as well by evaluating the answers to 3-way positive conjunction queries on the synthetic output.

The key to our method is the use of copulas [Nelsen, 2006] to generate synthetic datasets. Informally, a copula is a function that maps the marginal distributions to the joint distribution of a multivariate distribution, thus defining dependence among random variables. Modelling the rows of the input dataset as samples of the (unknown) multivariate distribution, we can use copulas to define dependence among attributes (modelled as univariate random variables), and finally use it to sample rows from the target distribution and generate synthetic datasets. Specifically, we use the Gaussian copula [Nelsen, 2006, p. 23], which defines the dependence between one-way margins through a covariance matrix. The underlying assumption is that the relationship between different attributes of the input dataset is completely characterised by pairwise covariances. While this may not be true in practice, it still preserves the correlation of highly correlated attributes in the synthetic output. Our main reason for using the Gaussian copula is its efficiency, as its run-time is proportional to the square of the number of attributes. We remark that our use of the Gaussian copula *does not* require the data attributes to follow a Gaussian distribution. Only their dependencies are assumed to be captured by the Gaussian copula.

Importantly, as claimed, our method is generic. This is important since an input dataset is expected to be a mixture of numerical (ordinal) and categorical (non-ordinal) attributes meaning that we cannot use a unified measure of correlation between attributes. A common technique to work around this is to create an artificial order on the categorical attributes [Iyengar, 2002]. However, as we show later, the resulting correlations are inherently artificial and exhibit drastically different results if a new arbitrary order is induced. Our approach is to convert the dataset into its binary format (see Section 2.2) and then use a single correlation measure, Pearson's product-moment correlation, for all binary attributes. This eliminates the need for creating an artificial order on the categorical attributes. Our method is thus more generic than another Copula-based method proposed by Li et al. [2014], which only handles attributes with large (discrete) domains and induces artificial order on categorical attributes.[2]

We experimentally evaluate our method on three real-world datasets: the publicly available Adult dataset [Lichman, 2013] containing US census information, a subset of the social security payments dataset provided to us by the Department of Social Services (DSS), a department of the Government of Australia,[3], and a hospital ratings dataset extracted from a national patient survey in the United States, which we call the Hospital dataset. The Adult dataset consists of 14 attributes (194 in binary) and more than 32,000 rows, the DSS dataset has 27 attributes (674 in binary) and more than 5,000,000 rows, and the Hospital dataset contains 9 attributes (1,201 in binary) and 10,000 rows. The generation of synthetic datasets took around 6 minutes 47 seconds (on average) for the Adult dataset, around two and a half hours on average for the DSS dataset, and 46 minutes on average for the Hospital dataset. To further check the scalability of our method, we ran it on two artificial datasets each with more than 5 million rows and 1,000 and 2,000 binary attributes, resulting in

[2]See Section 5 for a further discussion on the differences between the two works.

[3]`https://www.dss.gov.au/`.

run-times of approximately 6 and 19 hours, respectively. Since synthetic data release is a one-off endeavour, these times are highly feasible. In terms of utility, we show that 99% of the one-way marginal queries have an absolute error of less than 300 on Adult, around 400 for DSS, and less than 150 for the Hospital dataset (cf. Table 2), where absolute error is defined as the absolute difference between true and differentially private answers computed from the synthetic dataset. In terms of two-way positive conjunction queries, we again see that 99% of the queries have an absolute error of less than 500 for Adult, less than 250 for DSS and only around 15 for the Hospital dataset. Furthermore, for most of the two-way queries we considerably outperform the Laplace [Dwork et al., 2006b] and Gaussian [Dwork et al., 2006a] mechanisms which add independent Laplace and Gaussian noise, respectively, to each of the query answers. Note that a further advantage of our method is that unlike these two mechanisms, we generate a synthetic dataset. We further expand our utility analysis to include three-way queries and show that our method again calibrates noise considerably better than the aforementioned Laplace and Gaussian mechanisms with 99% of the queries having absolute error of less than 400 for Adult, less than 200 for DSS, and only around 50 for the Hospital dataset. Our utility analysis is thorough; we factor in the possibility that real-world datasets may have significantly high numbers of uncorrelated attributes which implies that a synthetic data generation algorithm might produce accurate answers by chance (for the case of two-way or higher order marginals). Thus we separate results for highly correlated and uncorrelated attributes to show the accuracy of our mechanism. Perhaps one drawback of our work is the lack of a theoretical accuracy bound; but this is in line with many recent works [Zhang et al., 2014, Cormode et al., 2012, Li et al., 2014] which, like us, promise utility in practice.

The rest of the paper is organized as follows. We give a brief background on differential privacy and copulas in Section 2. We describe our mechanism in Section 3. Section 4 contains our experimental utility and performance analysis. In Section 5, we present a detailed comparison between our work and the work most related to ours [Li et al., 2014]. We present related work in Section 6, and conclude in Section 7.

2. Background concepts

2.1. **Notations.** We denote the original dataset by D, which is modelled as a multiset of n rows from the domain $\mathcal{X} = A_1 \times \cdots \times A_m$, where each A_j represents an attribute, for a total of m attributes. We assume the number of rows n is *publicly known.* We denote the set of all n-row databases as $\mathcal{X}^n$. Thus, $D \in \mathcal{X}^n$. The ith row of D is denoted as $(X_1^{(i)}, X_2^{(i)}, \ldots, X_m^{(i)})$, where $X_j^{(i)} \in A_j$. In the sequel, where discussing a generic row, we will drop the superscript for brevity. The notation $\overline{\mathbb{R}}$ represents the real number line $[-\infty, \infty]$, and $\mathbb{I}$ denotes the interval of real numbers $[0, 1]$. The indicator function $I\{P\}$ evaluates to 1 if the predicate P is true, and 0 otherwise.

2.2. **Dummy Coding.** A key feature of our method is to convert the original dataset into its binary format through dummy coding, where each attribute value of $A_j \in D$ is assigned a new binary variable. For instance, consider the "country" attribute having attribute values `USA`, `France` and `Australia`. Converted to binary we will have three columns (binary attributes) one for each of the three values. For any row, a value of 1 corresponding to any of these binary columns indicates that the individual is from that particular country. Thus, we have a total of $d = \sum_{i=1}^{m} |A_i|$ binary attributes in the binary version D_{B} of the dataset D. In case of continuous valued attributes, this is done by first binning the values into discrete bins. Thus for a continuous valued attribute A, $|A|$ indicates the number of bins. The ith row in D_{B} is denoted as $(X_1^{(i)}, X_2^{(i)}, \ldots, X_d^{(i)})$. Note that while the mapping from D to D_{B} is unique, the converse is not always true.[4] Note further that this way of representing datasets is also known as histogram representation [Dwork and Roth, 2014, §2.3].

2.3. **Overview of Differential Privacy.**

Two databases $D, D' \in \mathcal{X}^n$ are called neighboring databases, denoted $D \sim D'$, if they differ in only one row.

Definition 1 (Differential privacy [Dwork et al., 2006b, Dwork and Roth, 2014])**.** A randomized algorithm (mechanism) $\mathcal{M} : \mathcal{X}^n \to R$ is (ϵ, δ)-differentially private if for every $S \subseteq R$, and for all neighbouring databases $D, D' \in \mathcal{X}^n$, the following holds

$$\mathbb{P}(\mathcal{M}(D) \in S) \leq e^{\epsilon}\mathbb{P}(\mathcal{M}(D') \in S) + \delta.$$

If $\delta = 0$, then $\mathcal{M}$ is ϵ-differentially private.

The mechanism $\mathcal{M}$ might also take some auxiliary inputs such as a query or a set of queries (to be defined shortly). The parameter δ is required to be a negligible function of n [Dwork and Roth, 2014, §2.3, p. 18], [Vadhan, 2017, §1.6, p. 9].[5] The parameter ϵ on the other hand should be small but not arbitrarily small. We may think of $\epsilon \leq 0.01$, $\epsilon \leq 0.1$ or $\epsilon \leq 1$ [Dwork et al., 2011, §1], [Dwork and Roth, 2014, §3.5.2, p. 52]. An important property of differential privacy is that it composes [Dwork and Roth, 2014].

Theorem 1 (Basic composition)**.** *If $\mathcal{M}_1, \ldots, \mathcal{M}_k$ are each (ϵ, δ)-differentially private then $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_k)$ is $(k\epsilon, k\delta)$-differentially private.* □

The above is sometimes referred to as (basic) *sequential* composition, as opposed to parallel composition, defined next.

Theorem 2 (Parallel composition [McSherry, 2009])**.** *Let $\mathcal{M}_i$ each provide (ϵ, δ)-differential privacy. Let $\mathcal{X}_i$ be arbitrary disjoint subsets of the domain $\mathcal{X}$. The sequence of $\mathcal{M}_i(D \cap \mathcal{X}_i)$ provides (ϵ, δ)-differential privacy, where $D \in \mathcal{X}^n$.* □

A more advanced form of composition, which we shall use in this paper, only depletes the "privacy budget" by a factor of $\approx \sqrt{k}$. See Dwork and Roth [2014][§3.5.2] for a precise definition of adaptive composition.

Theorem 3 ((Advanced) adaptive composition [Dwork et al., 2010, Dwork and Roth, 2014, Gaboardi et al., 2014])**.** *Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$ be a sequence of mechanisms where each can take as input the output of a previous mechanism, and let each be ϵ'-differentially private. Then $\mathcal{M}(D) = (\mathcal{M}_1(D), \ldots, \mathcal{M}_k(D)$ is ϵ-differentially private for $\epsilon = k\epsilon'$, and (ϵ, δ)-differentially private for*

$$\epsilon = \sqrt{2k \ln \frac{1}{\delta}}\epsilon' + k\epsilon'(e^{\epsilon'} - 1),$$

for any $\delta \in (0, 1)$. Furthermore, if the mechanisms $\mathcal{M}_i$ are each (ϵ', δ') differentially private, then M is $(\epsilon, k\delta' + \delta)$ differentially private with the same ϵ as above. □

[4]For instance, in D_{B}, we may have two binary attributes set to 1, which correspond to two different values of the same attribute in the original dataset D, e.g., `USA` and `France`.

[5]A function f in n is negligible, if for all $c \in \mathbb{N}$, there exists an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, it holds that $f(n) < n^{-c}$.

Another key feature of differential privacy is that it is immune to post-processing.

Theorem 4 (Post-processing [Dwork and Roth, 2014])**.** *If $\mathcal{M} : \mathcal{X}^n \to R$ is (ϵ, δ)-differentially private and $f : R \to R'$ is any randomized function, then $f \circ \mathcal{M} : \mathcal{X}^n \to R'$ is (ϵ, δ)-differentially private.* □

A query is defined as a function $q : \mathcal{X}^n \to \mathbb{R}$.

Definition 2 (Counting queries and point functions)**.** A *counting* query is specified by a predicate $q : \mathcal{X} \to \{0, 1\}$ and extended to datasets $D \in \mathcal{X}^n$ by summing up the predicate on all n rows of the dataset as

$$q(D) = \sum_{x \in D} q(x).$$

A point function [Vadhan, 2017] is the sum of the predicate $q_y : \mathcal{X} \to \{0, 1\}$, which evaluates to 1 if the row is equal to the point $y \in \mathcal{X}$ and 0 otherwise, over the dataset D. Note that computing all point functions, i.e., answering the query $q_y(D)$ for all $y \in \mathcal{X}$, amounts to computing the histogram of the dataset D. □

Definition 3 (Global sensitivity [Dwork and Roth, 2014, Vadhan, 2017])**.** The global sensitivity of a counting query $q : \mathcal{X}^n \to \mathbb{N}$ is

$$\Delta q = \max_{\substack{D, D' \in \mathcal{X}^n \\ D \sim D'}} |q(D) - q(D')|.$$

□

Definition 4 ((α, β)-utility)**.** The mechanism $\mathcal{M}$ is said to be (α, β)-useful for the query class Q if for any $q \in Q$,

$$\mathbb{P}\left[\mathsf{util}(\mathcal{M}(q, D), q(D)) \leq \alpha\right] \geq 1 - \beta,$$

where the probability is over the coin tosses of $\mathcal{M}$, and util is a given metric for utility.

The Laplace mechanism is employed as a building block in our algorithm to generate the synthetic dataset.

Definition 5 (Laplace mechanism [Dwork et al., 2006b])**.** The Laplace distribution with mean 0 and scale b has the probability density function

$$\mathrm{Lap}(x \mid b) = \frac{1}{2b} e^{-\frac{|x|}{b}}.$$

We shall remove the argument x, and simply denote the above by $\mathrm{Lap}(b)$. Let $q : \mathcal{X}^n \to \mathbb{R}$ be a query. The mechanism

$$\mathcal{M}_{\mathrm{Lap}}(q, D, \epsilon) = q(D) + Y$$

where Y is drawn from the distribution $\mathrm{Lap}\left(\Delta q / \epsilon\right)$ is known as the Laplace mechanism. The Laplace mechanism is ϵ-differentially private[Dwork et al., 2006b],[Dwork and Roth, 2014][§3.3]. Furthermore, with probability at least $1 - \beta$ [Dwork and Roth, 2014][§3.3]

$$\max_{q \in Q} |q(D) - \mathcal{M}_{\mathrm{Lap}}(q, D, \epsilon)| \leq \frac{\Delta q}{\epsilon} \ln\left(\frac{1}{\beta}\right) \doteq \alpha,$$

where $\beta \in (0, 1]$. If q is a counting query, then $\Delta q = 1$, and the error α is

$$\alpha = \frac{1}{\epsilon} \ln\left(\frac{1}{\beta}\right)$$

with probability at least $1 - \beta$. □

The advanced composition theorem (Theorem 3) allows us to set an ϵ and δ, giving us a budget ϵ' for each of the k mechanisms used in the composition. We can then use k applications of the Laplace mechanism: $\mathcal{M}_{\text{Lap}}(q, D, \epsilon')$ resulting in (ϵ, δ)-differential privacy. Alternatively, we can use k applications of the Gaussian mechanism to achieve (ϵ, δ)-differential privacy. Abusing notation for $k \geq 1$, let $q : \mathcal{X}^n \to \mathbb{N}^k$ now denote a sequence of counting queries. Then $q(D)$ denotes the tuple obtained after applying each of the k counting queries on the database D.

Definition 6 (Global l_2 sensitivity [Dwork and Roth, 2014])**.** The global l_2 sensitivity of a sequence of $k \geq 1$ counting queries $q : \mathcal{X}^n \to \mathbb{N}^k$ is

$$\Delta_2 q = \max_{\substack{D, D' \in \mathcal{X}^n \\ D \sim D'}} ||q(D) - q(D')||_2.$$

□

Definition 7 (Gaussian mechanism [Dwork et al., 2006a] [Dwork and Roth, 2014](§3.5.3))**.** The Gaussian mechanism with scale σ adds zero-mean Gaussian noise with variance σ^2, denoted $\mathcal{N}(0, \sigma^2)$, to each of the $k \geq 1$ counting queries in q. Let $\epsilon \in (0, 1)$ be arbitrary. The Gaussian mechanism with scale

$$\sigma > \frac{\Delta_2 q}{\epsilon} \sqrt{2 \ln \left(\frac{5}{4\delta} \right)} \tag{2.1}$$

is (ϵ, δ)-differentially private. □

2.4. **Overview of Copulas.** For illustration, we assume the bivariate case, i.e., we have only two variables (attributes) X_1 and X_2. Let F_1 and F_2 be their margins, i.e., cumulative distribution functions (CDFs), and let H be their joint distribution function. Then, according to Sklar's theorem [Sklar, 1959, Nelsen, 2006], there exists a copula C such that for all $X_1, X_2 \in \overline{\mathbb{R}}$,

$$H(X_1, X_2) = C(F_1(X_1), F_2(X_2)),$$

which is unique if F_1 and F_2 are continuous; otherwise it is uniquely determined on $\text{Ran}(F_1) \times \text{Ran}(F_2)$, where $\text{Ran}(\cdot)$ denotes range. In other words, there is a function that maps the joint distribution function to each pair of values of its margins. This function is called a copula. Since our treatment is on binary variables $X \in \{0, 1\}$, the corresponding margins are defined over $X \in \overline{\mathbb{R}}$ as

$$F(X) = \begin{cases} 0, & X \in [-\infty, 0) \\ a, & X \in [0, 1) \\ 1, & X \in [1, \infty] \end{cases}$$

where, $a \in \mathbb{I}$. The above satisfies the definition of a distribution function [Nelsen, 2006, §2.3]. This allows us to define the *quasi-inverse* of F, denoted F^{-1}, (tailored to our case) as

$$F^{-1}(t) = \begin{cases} 0, & t \in [0, a] \\ 1, & t \in (a, 1] \end{cases}$$

Now, using the quasi-inverses, we see that

$$C(u, v) = H(F_1^{-1}(u), F_2^{-1}(v)),$$

where $u, v \in \mathbb{I}$. If an analytical form of the joint distribution function is known, the copula can be constructed from the expressions of F_1^{-1} and F_2^{-1} (provided they exist). This then allows us to generate random samples of X_1 and X_2 by first sampling a uniform $u \in \mathbb{I}$, and then extracting v from the conditional distribution. Using the inverses we can extract the pair X_1, X_2. See Nelsen [2006][§2.9] for more details. However, in our case, and in general for real-world datasets, we seldom have an analytical expression for H, which could allow us to construct C. There are two approaches to circumvent this, which rely on obtaining an empirical estimate of H from D_{B}.

- The first approach relies on constructing a discrete equivalent of an empirical copula [Nelsen, 2006, §5.6, p. 219], using the empirical H obtained from the dataset D_{B}. However, doing this in a differentially private manner requires computing answers to the set of all point functions (Definition 2) of the original dataset [Vadhan, 2017], which amounts to finding the histogram of D_{B}. Unfortunately, for high dimensional datasets, existing *efficient* approaches of differentially private histogram release [Vadhan, 2017] would release a private dataset that discards most rows of the original dataset. This is due to the fact that high dimensional datasets are expected to have high number of rows with low multiplicity.
- The other approach, and indeed the one taken by us, is using some existing copula and adjusting its parameters according to the dataset D_{B}. For this paper we choose the Gaussian copula, i.e., the copula
$$C(u, v) = \mathbf{\Phi}_r(\Phi^{-1}(u), \Phi^{-1}(v)),$$
where Φ is the standard univariate normal distribution, r is the Pearson correlation coefficient and $\mathbf{\Phi}_r$ is the standard bivariate normal distribution with correlation coefficient r. We can then replace the Φ's with the given marginals F and G resulting in a distribution which is not standard bivariate, if F and G are not standard normal themselves. The underlying assumption in this approach is that the given distribution H is completely characterised by the margins and the correlation. This in general may not be true of all distributions H. However, in practice, this can potentially provide good estimates of one way margins and a subset of the two way margins as discussed earlier. Another advantage of this approach is computational efficiency: the run time is polynomial in m (the number of attributes) and n (the number or rows).

While our introduction to copulas has focused on two variables, this can be extended to multiple variables [Sklar, 1959].

3. Proposed Mechanism

Our method has three main steps as shown in Figure 1: data pre-processing, differentially private statistics, and copula construction. Among these, only the second step involves privacy treatment. The last step preserves privacy due to the post-processing property of differential privacy (Theorem 4). The first step, if not undertaken carefully, may result in privacy violations, as we shall discuss shortly. We shall elaborate each step in the following.

3.1. **Data Pre-processing.**

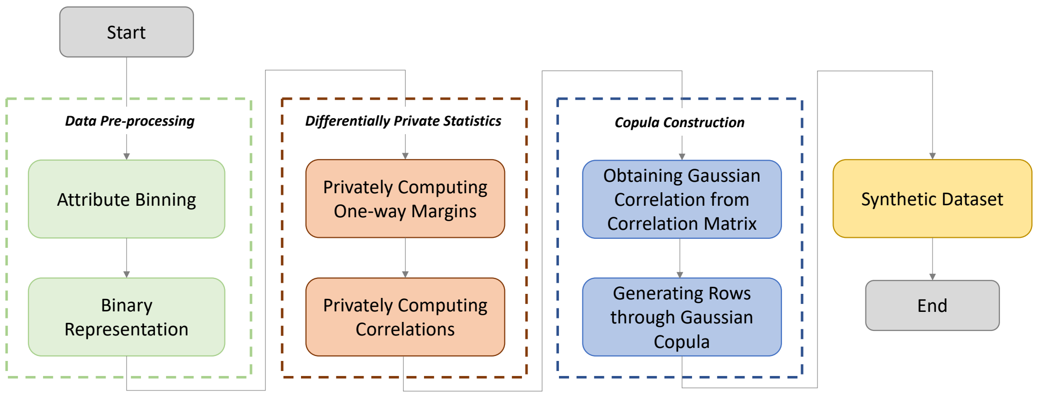

Start
Data Pre-processing
Attribute Binning
Binary Representation
Differentially Private Statistics
Privately Computing One-way Margins
Privately Computing Correlations
Copula Construction
Obtaining Gaussian Correlation from Correlation Matrix
Generating Rows through Gaussian Copula
Synthetic Dataset
End

Figure 1: Flowchart of our method.

3.1.1. *Binning of Continuous and Ordinal Data.* Since we will convert the original data into its binary format we need to ensure that the resulting expansion, i.e., $d = \sum_{i=1}^{m} |A_i|$, does not become prohibitively large. This will in general be the case with continuous attributes or discrete valued attributes with large domains. To overcome this, we bin these attributes into discrete intervals. However, care must be taken to ensure that the resulting binning is done independent of the input dataset D to avoid privacy violations. For instance, data-specific binning could introduce a bin due to an outlier which would reveal the presence and absence of that particular individual, violating differential privacy. We, therefore, assume that this binning is done only through what is publicly known about each attribute. For instance, when binning the year of birth into bins of 10 years, we assume that most individuals in the dataset have an age less than 120 (without checking this in the dataset). This of course depends on the nature of the attributes. While binning can be done in a differentially private manner, we use a simpler approach as this is not the main theme of our paper. We also note that data binning as a pre-processing step is almost always adopted in related work [Gaboardi et al., 2014].

3.1.2. *Binary Representation.* In order to use the Gaussian copula, we need to determine correlations between attributes via some fixed correlation *coefficient*; which is a numeric measure. We use the Pearson product moment correlation coefficient for this purpose. This means that in order to measure correlations, either the attribute values in the dataset need to be real valued or have to be mapped to a real number. These mapped values then follow their order in the set of real numbers. If an attribute is ordinal, e.g., age or salary, a natural order exists. However, categorical attributes do not have a natural order, e.g., gender. One way to solve this conundrum is to induce an artificial order among the values of the categorical attribute, e.g., through a hierarchical tree [Iyengar, 2002]. However, this ordering is inherently artificial and any change in order results in a markedly different correlation. An illustration of this fact is shown in Appendix A (we also briefly discuss this in Section 5.1).[6] Our approach instead is to work on the binary version of the dataset obtained via dummy coding, as explained in Section 2.2. Pairwise correlations now amount to finding Pearson correlation between pairs of binary variables. This way of interpreting

[6]There are other correlation measures that can be used for categorical attributes, e.g., Cramér's V [Cramér, 2016]. However, we need a unified approach that is applicable to all attributes alike.

data makes no distinction between ordinal and categorical variables by not assigning any order to the latter.

3.2. **Differentially Private Statistics.** For the differentially private statistics we can either use the Laplace mechanism (Definition 5) or the Gaussian mechanism (Definition 7). We will use the Laplace mechanism as the "default" and show how the Gaussian mechanism can be used in place of Laplace.

3.2.1. *Privately Computing One-Way Margins.* The first ingredient in our method is the one-way margins, i.e., marginal CDFs, of the attributes X_i, for $i \in \{1, 2, \ldots, d\}$. We denote these margins by $\hat{F}_i(x)$, where $x \in \{0, 1\}$. Since each X_i is binary, these margins can be calculated as

$$\hat{F}_i(0) = \frac{\sum_{k=1}^{n} I\{X_i^{(k)} = 0\}}{n}, \quad \hat{F}_i(1) = 1$$

To make $\hat{F}_i(x)$ differentially private, we add Laplace noise to the counts $\hat{n}_i^0 = \sum_{k=1}^{n} I\{X_i^{(k)} = 0\}$ and $\hat{n}_i^1 = \sum_{k=1}^{n} I\{X_i^{(k)} = 1\}$ to obtain $\tilde{F}_i(x)$, which is summarized in Algorithm 1. If the differentially private sum is negative, we fix it to 0. Note that this utilizes the post-processing property (cf. Theorem 4) and hence maintains privacy. The reason we add noise to both $\hat{n}_i^0$ and $\hat{n}_i^1$ instead of just adding noise to $\hat{n}_i^0$ is to avoid the noisy $\hat{n}_i^0$ exceeding n.[7]

ALGORITHM 1: Obtaining Differentially Private Marginals $\tilde{F}_i(x)$

---

(1) For the ith attribute, count the numbers of events $X_i = 0$ and $X_i = 1$ as

$$\hat{n}_i^0 = \sum_{k=1}^{n} I\left\{X_i^{(k)} = 0\right\}, \quad \hat{n}_i^1 = \sum_{k=1}^{n} I\left\{X_i^{(k)} = 1\right\}$$

(2) Add Laplace noise to $\hat{n}_i^0$ and $\hat{n}_i^1$, and obtain the noisy counts as

$$\tilde{n}_i^0 = \hat{n}_i^0 + \text{Lap}\left(\frac{2}{\epsilon_i'}\right), \quad \tilde{n}_i^1 = \hat{n}_i^1 + \text{Lap}\left(\frac{2}{\epsilon_i'}\right)$$

where ϵ_i' is the privacy budget associated with computing the ith margin.

(3) Obtain $\tilde{F}_i(x)$ as

$$\tilde{F}_i(0) = \frac{\tilde{n}_i^0}{\tilde{n}_i^0 + \tilde{n}_i^1}, \quad \tilde{F}_i(1) = 1$$

Algorithm 1 is $(\epsilon_i', 0)$-differentially private due to differential privacy of the Laplace mechanism and the fact that the algorithm is essentially computing the histogram associated with attribute X_i [Dwork and Roth, 2014, §3.3, p. 33]. Importantly, the privacy budget ϵ_i' is impacted only by the number of attributes m in the original dataset D and not by the number of binary attributes d in the binary version D_{B}. This is shown by the following lemma.

Lemma 1. *Fix an attribute A in D, and let $X_1, \ldots, X_{|A|}$ denote the binary attributes constructed from A. Then if the computation of each marginal X_j, $j \in [|A|]$, is $(\epsilon_i', 0)$-differentially private, the computation of the marginal A is $(\epsilon_i', 0)$-differentially private.*

[7]Alternatively, we could obtain $\hat{n}_i^1$ as $n - \hat{n}_i^0$, since n is considered public, and project it within $[0, n]$ if it falls outside.

Proof. See Appendix B. □

Using the Gaussian Mechanism. To obtain the Gaussian equivalent of Algorithm 1 we simply replace Laplace noise with Gaussian noise $\mathcal{N}(0, \sigma'^2)$, where σ' is any σ satisfying Eq. 2.1. Notice that since the query function is essentially a histogram function, its global l_2 sensitivity (Definition 6) is $\sqrt{2}$. Likewise, following Lemma 1, the global l_2 sensitivity of the query over the attribute A in D remains $\sqrt{2}$, and hence we can use Gaussian noise of the same scale added to all binary attributes constructed from A. The resultant algorithm is (ϵ, δ)-differentially private.

3.2.2. *Privately Computing Correlations.* The other requirement of our method is the computation of pairwise correlations given by

$$\hat{r}_{i,j} = \frac{\mathbb{E}\left\{X_i X_j\right\} - \mu_i \mu_j}{\sqrt{\operatorname{var}(X_i)\operatorname{var}(X_j)}}, \quad i, j \in \{1, 2, \ldots, d\}. \tag{3.1}$$

To obtain the differentially private version of $\hat{r}_{i,j}$, denoted $\tilde{r}_{i,j}$, one way is to compute $\hat{r}_{i,j}$ directly from D_{B} and then add Laplace noise scaled to the sensitivity of $\hat{r}$. However, as we show in Appendix C, the empirical correlation coefficient from binary attributes has high global sensitivity, which would result in highly noisy $\tilde{r}_{i,j}$. The other approach is to compute each of the terms in Eq. 3.1 in a differentially private manner and then obtain $\tilde{r}_{i,j}$. Notice that for binary attribute X_i, its mean μ_i is given by $\hat{n}_i^1/n$. This can be obtained differentially privately as

$$\tilde{\mu}_i = 1 - \tilde{F}_i(0) = 1 - \frac{\tilde{n}_i^0}{\tilde{n}_i^0 + \tilde{n}_i^1},$$

which we have already computed. Likewise, the variance $\hat{\operatorname{var}}(X_i)$ is given by $\hat{\mu}_i(1 - \hat{\mu}_i)$, whose differentially private analogue, i.e., $\tilde{\operatorname{var}}(X_i)$, can again be obtained from the above. Thus, the only new computation is the computation of $\mathbb{E}\left\{X_i X_j\right\}$, which for binary attributes is equivalent to computing $\frac{1}{n}\sum_{k=1}^{n} I\{(X_i^{(k)} = 1) \wedge (X_j^{(k)} = 1)\}$. Algorithm 2 computes this privately using the Laplace mechanism.

ALGORITHM 2: Obtaining Differentially Private Two-Way Positive Conjunctions $\tilde{\mathbb{E}}\left\{X_i X_j\right\}$

(1) For the ith and jth attributes ($1 \leq i < j \leq d$), and for $a, b \in \{0, 1\}$, count the number of events $(X_i, X_j) = (a, b)$ as

$$\hat{n}_{i,j}^{ab} = \sum_{k=1}^{n} I\left\{(X_i^{(k)} = a) \cap (X_j^{(k)} = b)\right\} \tag{3.2}$$

(2) Add Laplace noise onto $\hat{n}_{i,j}^{ab}$ for $a, b \in \{0, 1\}$ and obtain the noisy count as

$$\tilde{n}_{i,j}^{ab} = \hat{n}_{i,j}^{ab} + \operatorname{Lap}\left(\frac{2}{\epsilon''_{i,j}}\right), \tag{3.3}$$

where $\epsilon''_{i,j}$ is the privacy budget associated with computing the (i, j)th two-way marginal.

(3) Obtain $\tilde{\mathbb{E}}\left\{X_i X_j\right\}$ as

$$\tilde{\mathbb{E}}\left\{X_i X_j\right\} = \frac{\tilde{n}_{i,j}^{ab}}{\sum_{a,b\in\{0,1\}} \tilde{n}_{i,j}^{ab}}. \tag{3.4}$$

Algorithm 2 is $(\epsilon''_{i,j}, 0)$-differentially private due to the differential privacy of the Laplace mechanism and the fact that the algorithm is essentially computing the histogram associated with attribute pairs (X_i, X_j) [Dwork and Roth, 2014, §3.3, p. 33]. Once again, the privacy budget $\epsilon''_{i,j}$ is impacted only by the number of pairs of attributes $\binom{m}{2}$ in the original dataset D and not by the number of pairs of binary attributes $\binom{d}{2}$ in the binary version D_{B}. This is presented in the following lemma, whose proof is similar to that of Lemma 1, and hence omitted for brevity.

Lemma 2. *Fix two attributes A_i and A_j, $i \neq j$, in D. Let $X_{i,1}, \ldots, X_{i,|A_i|}$ and $X_{j,1}, \ldots, X_{j,|A_j|}$ denote the binary attributes constructed from A_i and A_j, respectively. Then, if the computation of each of the two-way marginals $(X_{i,k}, X_{j,k'})$, $k \in [|A_i|], k' \in [|A_j|]$, is $(\epsilon''_{i,j}, 0)$-differentially private, the computation of all two-way marginals of A_i and A_j is $(\epsilon''_{i,j}, 0)$-differentially private.* □

The differentially private correlation coefficients $\tilde{r}_{i,j}$ thus obtained can be readily used to construct the differentially private correlation matrix

$$\tilde{\mathbf{R}} = \begin{bmatrix} \tilde{r}_{1,1} & \tilde{r}_{1,2} & \cdots & \tilde{r}_{1,d} \\ \tilde{r}_{2,1} & \tilde{r}_{2,2} & \cdots & \tilde{r}_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{r}_{d,1} & \tilde{r}_{d,2} & \cdots & \tilde{r}_{d,d} \end{bmatrix}. \tag{3.5}$$

Notice that the above algorithm computes the correlations in time $O(d^2 n)$. This can be prohibitive if d is large, i.e., if each of the m (original) attributes have large domains. An alternative algorithm to compute the two-way positive conjunctions that takes time only $O(m^2 n)$ is shown in Appendix D.

Using the Gaussian Mechanism. To obtain the Gaussian mechanism equivalent of Algorithm 2 we once again replace Laplace noise with Gaussian noise $\mathcal{N}(0, \sigma''^2)$, where σ'' is any σ satisfying Eq. 2.1. The global l_2 sensitivity is again $\sqrt{2}$, and following Lemma 2, the global l_2 sensitivity of the query over any attribute pair (A_i, A_j) in D remains $\sqrt{2}$, and hence we can use Gaussian noise of the same scale added to all pairs of binary attributes constructed from (A_i, A_j). The resulting algorithm is (ϵ, δ)-differentially private.

3.3. Copula Construction.

For this section, we do not need access to the database any more. Hence, any processing done preserves the previous privacy budget due to closure under post-processing (Theorem 4).

3.3.1. *Obtaining Gaussian Correlation from the Correlation Matrix.*

Our aim is to sample standard normal Gaussian variables Y_i's corresponding to the attributes $\tilde{X}_i$'s[8] where the correlations among Y_i's, given by the correlation matrix $\mathbf{P}$, are *mapped* to the correlations among the X_i's, given by the (already computed) correlation matrix $\tilde{\mathbf{R}}$. A sample from the Gaussian variable Y_i is transformed backed to $\tilde{X}_i$ as

$$\tilde{X}_i = \tilde{F}_i^{-1}\left(\Phi\left(Y_i\right)\right). \tag{3.6}$$

Obviously, if the attributes $\tilde{X}_i$ are independent, then Y_i are also independent. However, in practice $\tilde{X}_i$'s are correlated, which is characterized by $\tilde{\mathbf{R}}$ in our case. Hence, the question

[8] i.e., the synthetic versions of the X_i's.

becomes: *How to choose a correlation matrix* $\mathbf{P}$ *for* Y_i*'s, so that* $\tilde{X}_i$*'s have the target correlation relationship defined by* $\tilde{\mathbf{R}}$*?*

From Eq. 3.1 and its perturbation through $\tilde{r}_{i,j}$, we can obtain

$$\mathbb{E}\left\{\tilde{X}_i\tilde{X}_j\right\} = \tilde{r}_{i,j}\sqrt{\operatorname{var}\left(\tilde{X}_i\right)\operatorname{var}\left(\tilde{X}_j\right)} + \tilde{\mu}_i\tilde{\mu}_j, \tag{3.7}$$

On the other hand, from Eq. 3.6, we can get

$$\begin{aligned}\mathbb{E}\left\{\tilde{X}_i\tilde{X}_j\right\} &= \mathbb{E}\left\{\tilde{F}_i^{-1}\left(\Phi\left(Y_i\right)\right)\tilde{F}_j^{-1}\left(\Phi\left(Y_j\right)\right)\right\}\\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\tilde{F}_i^{-1}\left(\Phi\left(y_i\right)\right)\tilde{F}_j^{-1}\left(\Phi\left(y_j\right)\right)\boldsymbol{\Phi}_{i,j}\left(y_i,y_j\right)dy_i dy_j, \qquad (3.8)\end{aligned}$$

where $\boldsymbol{\Phi}_{i,j}\left(y_i,y_j\right)$ denotes the standard bivariate probability density function (PDF) of the correlated standard normal random variables Y_i and Y_j, given by

$$\boldsymbol{\Phi}_{i,j}\left(y_i,y_j\right) = \frac{1}{2\pi\sqrt{1-\rho_{i,j}^2}}\exp\left(-\frac{y_i^2+y_j^2-2\rho_{i,j}y_iy_j}{2\left(1-\rho_{i,j}^2\right)}\right). \tag{3.9}$$

Here $\rho_{i,j}$ is the Pearson correlation coefficient, which for the standard normal variables Y_i and Y_j is given by $\mathbb{E}\left\{Y_iY_j\right\}$. Our task is to find the value of $\rho_{i,j}$ such that Eq. 3.7 and Eq. 3.8 are equal. In other words, Eqs. 3.7 and 3.8 define the relationship between $\tilde{r}_{i,j}$ and $\rho_{i,j}$. Notice that by construction, in general, we do *not* have $\rho_{i,j} = \tilde{r}_{i,j}$. We can obtain $\rho_{i,j}$ by means of a standard bisection search (e.g., see Nelson [2015][p. 148]). The two-fold integral in Eq. 3.8 with respect to y_i and y_j, is evaluated numerically in the bisection search. In more detail, dy_i and dy_j are set to a small value of 0.01, and the lower and upper limits of the integral are set to -10 and 10, respectively. Such lower and upper limits make the numerical results sufficiently accurate since $\boldsymbol{\Phi}_{i,j}\left(y_i,y_j\right)$ is the standard bivariate PDF of two correlated normal random variables (and hence have negligibly small probability mass beyond the limits of integration). With each such $\rho_{i,j}$, we can construct the matrix $\mathbf{P}$ corresponding to $\tilde{\mathbf{R}}$.

Remark. Notice that the choice of $\rho_{i,j}$ ensures that the resulting sampled Gaussian variables have the property that when transformed back to $\tilde{X}_i$ and $\tilde{X}_j$, we get $\mathbb{E}\left\{\tilde{X}_i\tilde{X}_j\right\} \approx \mathbb{E}\left\{X_iX_j\right\}$, where the latter is the input expectation. For binary attributes, recall that $\mathbb{E}\left\{X_iX_j\right\} = \frac{1}{n}\sum_{k=1}^{n} I\left\{(X_i^{(k)}=1)\cap(X_j^{(k)}=1)\right\}$. Thus the method ensures that the "11"'s are well approximated to the input distribution. However, the correlation coefficient, and the distribution of 01's, 10's and 00's might not be the same as the original dataset. This is evident from Eq. 3.7. However, since the remaining quantities in Eq. 3.7 depend on the one-way margins, maintaining a good approximation to margins would imply that the distribution of 01's, 00's and 10's would also be well approximated. However, error in one or both of the margins would propagate to the error in the 01's, 10's and 00's. Thus, on average we would expect 01's, 10's and 00's to have twice the error than the 11's (since they can be obtained via the 11's and the one-way marginals). It is due to this reason that we target positive two-way conjunctions for utility.

ALGORITHM 3: Algorithm to Obtain the Nearest Correlation Matrix ($\mathbf{P}$)

```
(1) Initialization: $\triangle \mathbf{S}_0 \leftarrow \mathbf{0}, \mathbf{Y}_0 \leftarrow \mathbf{P}$, $k \leftarrow 1$.
(2) While $k <$ iters, where iters is the maximum of iterations
    (a) $\mathbf{R}_k = \mathbf{Y}_{k-1} - \triangle \mathbf{S}_{k-1}$
    (b) Projection of $\mathbf{R}_k$ to a positive semidefinite matrix: $\mathbf{X}_k \leftarrow \mathbf{V}^{\mathrm{T}} \mathrm{diag}\left(\max\left\{\mathbf{\Lambda}, \mathbf{0}\right\}\right) \mathbf{V}$, where $\mathbf{V}$
        and $\mathbf{\Lambda}$ contain the eigenvectors and the eigenvalues of $\mathbf{R}_k$, respectively, and diag $(\cdot)$
        transforms a vector into a diagonal matrix.
    (c) $\triangle \mathbf{S}_k \leftarrow \mathbf{X}_k - \mathbf{R}_k$
    (d) Projection of $\mathbf{X}_k$ to a unit-diagonal matrix: $\mathbf{Y}_k \leftarrow \mathrm{unitdiag}\left(\mathbf{X}_k\right)$, where unitdiag$(\cdot)$ fixes
        the diagonal of the input matrix to ones.
    (e) $k \leftarrow k + 1$.
(3) Output: $\mathbf{P}' \leftarrow \mathbf{Y}_k$.
```

3.3.2. *Generating Records Following a Multivariate Normal Distribution.* Since each Y_i follows the standard normal distribution and the correlation among Y_i's is characterized by $\mathbf{P}$, we can generate records by sampling from the resulting multivariate normal distribution. However, the matrix $\mathbf{P}$ obtained through this process should have two important properties for this method to work:

(1) $\mathbf{P}$ should be a correlation matrix, i.e., $\mathbf{P}$ should be a symmetric positive semidefinite matrix with unit diagonals [Higham, 2002].
(2) $\mathbf{P}$ should be positive definite[9] to have a unique Cholesky decomposition defined as $\mathbf{P} = \mathbf{L}^T\mathbf{L}$ where $\mathbf{L}$ is a lower triangular matrix with positive diagonal entries [Golub and Van Loan, 2013, p. 187].

To ensure Property 1, we use the algorithm from Higham [2002][§3.2], denoted nearest correlation matrix (NCM), to obtain the matrix $\mathbf{P}'$ as

$$\mathbf{P}' = \mathrm{NCM}\left(\mathbf{P}\right), \tag{3.10}$$

Furthermore, to satisfy Property 2, i.e., positive definiteness, we force the zero eigenvalues of $\mathbf{P}'$ to small positive values. For completeness, we describe the simplified skeleton of the algorithm from Higham [2002] in Algorithm 3. The algorithm searches for a correlation matrix that is closest to $\mathbf{P}$ in a weighted Frobenius norm. The output is asymptotically guaranteed to output the nearest correlation matrix to the input matrix [Higham, 2002]. For details see Higham [2002][§3.2, p. 11]. The overall complexity of the procedure is $O(d^\omega)$, where $\omega < 2.38$ is the coefficient in the cost of multiplying two $d \times d$ matrices [Bardet et al., 2003]. Note that matrix multiplication is highly optimized in modern day computing languages.

Finally, we generate records from a multivariate normal distribution using the well known method described in Algorithm 4. The rationale of invoking Cholesky decomposition is to ensure that

$$\mathbb{E}\left\{\mathbf{Y}^T\mathbf{Y}\right\} = \mathbb{E}\left\{\mathbf{L}^T\mathbf{Z}^T\mathbf{Z}\mathbf{L}\right\} = \mathbf{L}^T\mathbb{E}\left\{\mathbf{Z}^T\mathbf{Z}\right\}\mathbf{L} = \mathbf{L}^T\mathbf{L} = \mathbf{P}',$$

where we have used the fact that $\mathbb{E}\left\{\mathbf{Z}^T\mathbf{Z}\right\} = \mathbf{I}$ because each record in $\mathbf{Z}$ follows i.i.d. multivariate normal distribution. The output dataset D'_{B} is the final synthetic dataset.

[9]Note that while a positive definite matrix implies a positive semidefinite matrix, the former is not a requirement for a matrix to be a correlation matrix. Hence we state this property separately.

ALGORITHM 4: Generating Records Following a Multivariate Normal Distribution

```
(1) Generate $n$ records, with the attributes in each record following i.i.d. standard normal
    distribution, i.e.,
        $\mathbf{Z} = \begin{bmatrix} Z^{(1)} & Z^{(2)} & \cdots & Z^{(n)} \end{bmatrix}^T$
    where $Z^{(k)}$ is given by $\begin{bmatrix} Z_1^{(k)} & Z_2^{(k)} & \cdots & Z_d^{(k)} \end{bmatrix}^T$ and $Z_i^{(k)}$'s are i.i.d. standard normal random
    variables.
(2) Compute $\mathbf{Y}$ as $\mathbf{Y} = \mathbf{ZL}$ where $\mathbf{L}^T\mathbf{L} = \mathbf{P}'$ and $\mathbf{L}$ is obtained by the Cholesky decomposition as
    $\mathbf{L} = \text{chol}(\mathbf{P}')$.
(3) From the matrix $\mathbf{Y} = \begin{bmatrix} Y^{(1)} & Y^{(2)} & \cdots & Y^{(n)} \end{bmatrix}^T$, map every element $Y_i^{(k)}$ in each record $Y^{(k)}$ to
    $\tilde{X}_i^{(k)}$ using Eq. 3.6, where $i \in \{1, 2, \ldots, d\}$.
(4) Output the mapped data as $D'_{\mathsf{B}}$.
```

3.4. **Privacy of the Scheme.** We first assume that the Laplace mechanism is used in Algorithms 1 and 2. Let $m' = m + \binom{m}{2}$, where m is the number of attributes in D. Let $\epsilon < 1$, say $\epsilon = 0.99$. Fix a δ. We set each of the ϵ'_i's and $\epsilon''_{i,j}$'s for $i, j \in [m]$ to ϵ', where ϵ' is such that it gives us the target ϵ through Theorem 3 by setting $k = m'$. According to the advanced composition theorem (Theorem 3), since each of our m' mechanisms are $(\epsilon', 0)$ differentially private, the overall construction is (ϵ, δ)-differentially private. Privacy budget consumed over each of the m' mechanisms is roughly $\frac{1}{\sqrt{m'}}$. Note that all the algorithms in Section 3.3 do not require access to the original dataset and therefore privacy is not impacted due to the post-processing property (see Theorem 4).

For the Gaussian mechanism variants of Algorithms 1 and 2, we again let $m' = m + \binom{m}{2}$, where m is the number of attributes in D. We fix an $\epsilon < 1$, say $\epsilon = 0.99$, and a δ. Then the sequence of m' histogram queries have l_2 sensitivity of $\sqrt{2m'}$. Putting these values in Eq. 2.1 gives us the scale of the Gaussian mechanism, i.e., σ. Thus, adding $\mathcal{N}(0, \sigma^2)$ noise in each invocation of the Gaussian mechanism satisfies (ϵ, δ)-differential privacy overall. As in the case of the Laplace mechanism, the remaining steps are merely post-processing. Note that we do not need to apply the advanced composition theorem in the case of Gaussian mechanism, as the guarantee is readily in terms of (ϵ, δ)-differential privacy.

We note that instead of (ϵ, δ)-differential privacy, we can also use the Gaussian mechanism under the notions of concentrated differential privacy (CDP) [Dwork and Rothblum, 2016] or zero-concentrated differential privacy (zCDP) [Bun and Steinke, 2016] which would result in further improvement in utility [Dwork and Rothblum, 2016]. Any utility gain will also be reflected in the sythetic dataset obtained via our copula construction. We continue with (ϵ, δ)-differential privacy, since we show relative gain in utility over independent noise addition.

4. Experimental Evaluation and Utility Analysis

4.1. **Query Class and Error Metrics.** As mentioned in Section 1 and explained in Section 3.3.1, our focus is on all one-way marginal and two-way positive conjunction queries. In addition, we will also evaluate the performance of our method on the set of three-way positive conjunction queries to demonstrate that our method can also give well approximated answers to other types of queries. Thus, we use the following query class to evaluate our method, $Q := Q_1 \cup Q_2 \cup Q_3$. Here, Q_1 is the set of one-way marginal counting queries, which consists of queries q specified as

$$q(i, D_{\mathsf{B}}) = \sum_{k=1}^{n} I\{X_i^{(k)} = b\}$$

where $i \in [d]$ and $b \in \{0, 1\}$. The class Q_2 is the set of positive two-way conjunctions and consists of queries q specified as

$$q(i, j, D_{\mathsf{B}}) = \sum_{k=1}^{n} I\{(X_i^{(k)} = 1) \cap (X_j^{(k)} = 1)\}$$

where $i, j \in [d], j \neq i$. We define $Q_{12} = Q_1 \cup Q_2$. The class Q_3 is the set of positive three-way conjunctions, and is defined analogously. Note that only those queries are included in Q_2 and Q_3 whose corresponding binary columns are from *distinct* original columns in D. Answers to queries which evaluate at least two binary columns from the same original column in D can be trivially fixed to zero; as these are "structural zeroes." We assume this to be true for queries in the two aforementioned query sets from here onwards. Our error metric of choice is the absolute error, which for a query $q \in Q$ is defined as

$$|q(D_{\mathsf{B}}) - q(D'_{\mathsf{B}})|$$

We have preferred this error metric over relative error (which scales the answers based on the original answer), since it makes it easier to compare results across different datasets and query types. For instance, in the case of relative error, a scaling factor is normally introduced, which is either a constant or a percentage of the number of rows in the dataset [Xiao et al., 2011, Li et al., 2014]. The scaling factor is employed to not penalize queries with extremely small answers. However, the instantiation of the scaling factor is mostly a heuristic choice.

For each of the datasets (described next), we shall evaluate the differences in answers to queries from Q in the remainder of this section. We shall be reporting (α, β)-utility in the following way. We first sort the query answers in ascending order of error. We then fix a value of β, and report the maximum error and average error from the first $1 - \beta$ fraction of queries. We shall use values of $\beta = 0.05, 0.01$ and 0. The errors returned then correspond to 95%, 99% and 100% (overall error) of the queries, respectively.

4.2. **Datasets and Parameters.** We used three real-world datasets to evaluate our method. All three datasets contained a mixture of ordinal (both discrete and continuous) and categorical attributes. We selected these three datasets as they each present characteristics that we believe are shared by many other real-life datasets, thus supporting the claim that our method can be generalised to a large number of existing datasets. These datasets are as follows.

(1) *Adult Dataset:* This is a publicly available dataset which is an extract from the 1994 US census information [Lichman, 2013]. There are 14 attributes in total which after pre-processing result in 194 binary attributes. There a total of 32,560 rows in this dataset (each belonging to an individual). These attributes are mostly about weakly correlated demographic information, with a combination of ordinal (e.g. year of birth) and categorical (e.g country of birth) attributes. We believe that a large set of real-life datasets being harvested and analysed in several sectors share these attribute characteristics (e.g., census data in other countries, tax or social services data, social

networks data). Moreover, this specific Adult dataset is also commonly employed in the performance evaluations of several studies on differentially private data releases [Li et al., 2014, Gaboardi et al., 2014]. Thus, using this dataset allows easy benchmarking of our own method against other published studies.

(2) *DSS Dataset:* This dataset is a subset of a dataset obtained from the Department of Social Services (DSS), a department of the Government of Australia. The dataset contains transactions of social security payments. The subset contains 27 attributes, which result in 674 binary attributes. There are 5,240,260 rows in this dataset. Compared to the Adult set, it contains a higher number (i.e., 27) of mostly categorical attributes (i.e. the specific characteristics of a single transaction). Again we believe that a large body of existing real-life datasets are of similar transactional nature (e.g. financial transactions, consumer profiles, healthcare services), with high dimension (e.g., above 20). These types of datasets also present subsets of highly correlated attributes (e.g. prices of good/services are related to their types or defining attributes), compared to demographic datasets.

(3) *Hospital Dataset:* This dataset is a subset of the hospital ratings dataset extracted from a national patient survey in the US.[10] The dataset is among many other datasets gathered by the Centers for Medicare & Medicaid Services (CMS), a federal agency in the US. We argue that this dataset is representative of many existing survey-type datasets, such as customer feedback, marketing questionnaires, or employee consultations. For this case, we further extracted 9 highly correlated attributes (resulting in 1,201 binary attributes) and 10,000 rows, as we wanted to illustrate the utility performance of our scheme on preserving such highly related information. This property is common in surveys, as they often have clusters of related questions, with often related answers.

Parameter Values. We set $\delta = 2^{-30}$, following a similar value used in Gaboardi et al. [2016][§3, p. 5]. This is well below n^{-1} for the three datasets, where n is the number of rows. We set the same privacy budget, ϵ', to compute each of the m one-way marginals and $\binom{m}{2}$ two-way positive conjunctions. For the Adult, DSS and Hospital datasets we have $m = 14$, $m = 27$ and $m = 9$, respectively. We search for an ϵ' for these datasets by setting $\delta = 2^{-30}$ and $k = m + \binom{m}{2}$ in Theorem 3 which gives an overall ϵ of just under 1. The resulting computation gives us $\epsilon' = 0.014782$ for Adult, $\epsilon' = 0.007791$ for DSS, and $\epsilon' = 0.022579$ for the Hospital dataset. For the Gaussian variant, we choose ϵ just under 1, e.g., $\epsilon = 0.99$, and $\delta = 2^{-30}$. Together with $\Delta_2 q = \sqrt{2k}$ (see Section 3.4) this gives us the scale of the Gaussian mechanism σ via Eq. 2.1 for each of the three datasets.

4.3. **Artifacts of the Synthetic Dataset.** Since the dataset is in binary format (Section 3.1), we may have two binary attributes corresponding to the same original attribute set to 1. For instance, the gender attribute may have both binary attributes (male and female) set to 1. Since the conversion of the dataset into its binary format is public information, it is understood that two-way queries on different values of the same attributes have an answer of 0. We assume that the analyst is aware of this. We note that this is also true of differentially private mechanisms that release synthetic datasets in the histogram representation [Gaboardi et al., 2014, Hardt et al., 2012].

[10]See `https://data.medicare.gov/Hospital-Compare/Patient-survey-HCAHPS-Hospital/dgck-syfz`.

Output	Mechanism	Differential Privacy	Synthetic
cop	Gaussian copula with original correlation matrix	✗	✓
cop-ID	Gaussian copula with identity correlation matrix	✗	✓
cop-1	Gaussian copula with correlation matrix of all 1's	✗	✓
no-cor	Answers generated assuming no correlation	✗	✗
Lap	Laplace mechanism	✓	✗
Gauss	Gaussian mechanism	✓	✗
dpc-Lap	Gaussian copula with DP correlation matrix via Laplace noise	✓	✓
dpc-Gauss	Gaussian copula with DP correlation matrix via Gaussian noise	✓	✓

Table 1: Notation used for outputs from different mechanisms. DP stands for differential privacy. Note that not all of them are synthetic datasets or differentially private.

4.4. **Experimental Analysis.** To evaluate our method, we generate multiple synthetic datasets from each of the three datasets. We will first evaluate the synthetic dataset generated through the Gaussian copula (with no differential privacy) for each of the three datasets. This will be followed by the evaluation of the Laplace and Gaussian noise-based differentially private versions of the synthetic dataset against the baselines which are the application of the Laplace and Gaussian mechanisms on the set of queries Q. Note that these do not result in a synthetic dataset. For readability, we use abbreviations for the different outputs. These are shown in Table 1.

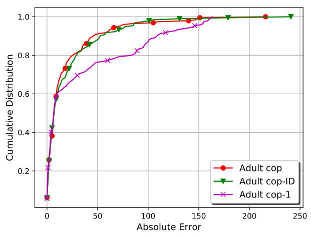
4.4.1. *Error due to Gaussian Copula without Differential Privacy.* We first isolate and quantify query errors from a synthetic dataset obtained directly through the Gaussian copula, i.e., without any differentially private noise added to the one-way marginals and the correlation matrix (see Figure 1). Since generating synthetic datasets through the copula is an inherently random process, this itself may be a source of error. We denote such a dataset by "cop." Thus, Adult cop, DSS cop and Hospital cop are the "cop" versions of the corresponding datasets. We restrict ourselves to the query set Q_{12} and compare error from cop against three other outputs:

cop-ID: A synthetic dataset obtained by replacing the correlation matrix $\mathbf{P}'$ (Eq. 3.10) with the identity matrix. Evaluating against this dataset will show whether cop performs better than a trivial mechanism which assumes all binary attributes to be uncorrelated. Note that this mainly effects answers to Q_2, and not the one-way marginals Q_1.

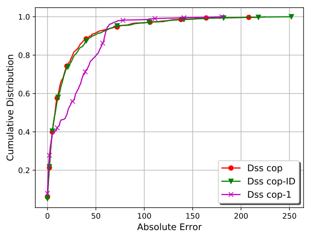
cop-1: Another synthetic dataset obtained by replacing the correlation matrix $\mathbf{P}'$ with the matrix $\mathbf{1}$ of all ones. This serves as the other extreme where all attributes are assumed to be positively correlated. Once again, this is to compare answers from Q_2.

no-cor: For the query class Q_2, we obtain a set of random answers which are computed by simply multiplying the means μ_1 and μ_2 of two binary attributes. This is the same as simulating two-way positive conjunctions of uncorrelated attributes. This should have an error distribution similar to the answers on Q_2 obtained from cop-ID.

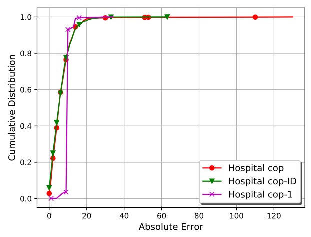
Results. Figure 2 shows the CDF of the absolute error on the query set Q_{12} from different outputs from the three datasets. Looking first at the results on Q_1 (top row in the figure), we see that for all three datasets, cop has low absolute error, yielding (α, β)-utility of $(149, 0.01)$ for Adult, $(170, 0.01)$ for DSS and $(28, 0.01)$ for the Hospital dataset in terms of max-absolute error. The datasets cop-ID and cop-1 exhibit similar utility for all three datasets. This is not surprising since Q_1 contains one-way marginals, whose accuracy is more impacted by the inverse transforms (Eq. 3.6) rather than the correlation matrix. Answers on Q_2 are more intriguing (bottom row of Figure 2). First note that the utility from cop is once again good for all three input datasets with 99% of the queries having a maximum absolute error of 353 for Adult, 83 for DSS and 6 for the Hospital dataset. The cop-1 outputs have poorer utility. However, interestingly, cop-ID and no-cor outputs yield utility very similar to cop. We discuss this in more detail next.



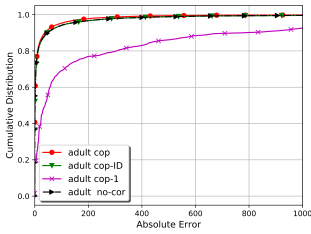
(a) Set Q_1 (one-way marginals)



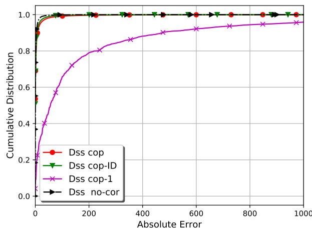
(b) Set Q_1 (one-way marginals)



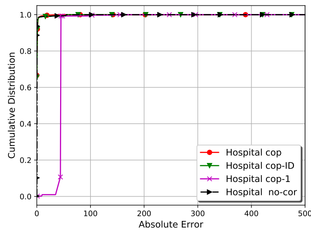
(c) Set Q_1 (one-way marginals)



(d) Set Q_2 (two-way conjunctions)



(e) Set Q_2 (two-way conjunctions)



(f) Set Q_2 (two-way conjunctions)

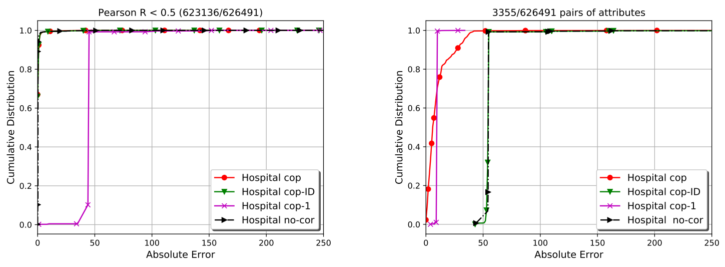
Figure 2: Relative error over Q_{12} on synthetic versions of the Adult, DSS and Hospital datasets (without differential privacy).

Separating Low and High Correlations. There are two possible reasons why cop-ID and no-cor outputs perform close to cop outputs on the set Q_2: (a) our method does not perform better than random (when it comes to Q_2), or (b) uncorrelated attributes dominate the dataset thus overwhelming the distribution of errors. The second reason is also evidenced by the fact that the cop-1 dataset, using a correlation matrix of all ones, performs worse than the other three outputs, indicating that highly correlated attributes are rare in the three datasets.

To further ascertain which of the two reasons is true, we separate the error results on the set Q_2 into two parts: a set of (binary) attribute pairs having high correlations (positive or negative), and another with low correlations. If our method performs better than random, we would expect the error from cop to be lower on the first set when compared to cop-ID and no-cor, while at least comparable to the two on the second set. We use the Hospital dataset for this analysis as it was the dataset with the most highly correlated attributes. There are a total of 626,491 pairs of binary attributes (as mentioned before, we ignore binary attribute pairs that correspond to different attribute values on the same attribute in the original dataset). Out of these, only 3,355 have an (absolute) Pearson correlation coefficient

$|r| \geq 0.5$. Thus, an overwhelming 99.46% of pairs have $|r| < 0.5$. This shows that the dataset does indeed contain a high number of low-correlated attributes, which partially explains similar error profile of cop, cop-ID and no-cor.

Figure 3 shows this breakdown. The error on the set with $|r| < 0.5$ is very similar for cop, cop-ID and no-cor (Figure 3(a)). Looking at Figure 3(b), for the set with $|r| \geq 0.5$, on the other hand, we note that cop outperforms both cop-ID and no-cor. Also note that cop-1 is similar in performance to our method. This is understandable since cop-1 uses a correlation matrix with all ones, and hence is expected to perform well on highly correlated pairs. This indicates that our method outperforms cop-ID and no-cor. We conclude that the *apparent* similarity between cop, cop-ID and no-cor on the set Q_2 is due to an artefact of some real-world datasets which may have a high number of uncorrelated attributes; when the results are analyzed separately, our method is superior. We remark that we arrived at the same conclusion for the Adult and DSS datasets, but omit the results due to repetition.



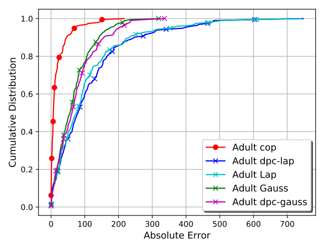
(a) $|r| < 0.5$ (b) $|r| \geq 0.5$

Figure 3: Absolute error over Q_2 on synthetic versions of the Hospital dataset (without differential privacy) for different values of the absolute Pearson correlation coefficient $|r|$ between pairs of attributes.

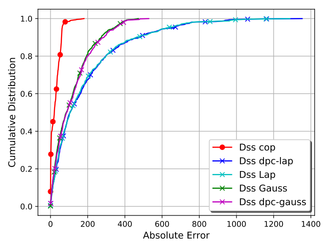
4.4.2. *Error due to Differentially Private Gaussian Copula.* Having established that the error due to Gaussian copula is small, we now turn to the complete version of our method, i.e., with differential privacy. For this section, we are interested in five outputs: (a) cop, i.e., the synthetic dataset via our Gaussian copula method without differential privacy, (b) dpc-Lap, i.e., the synthetic dataset obtained through our Laplace mechanism based differentially private Gaussian copula method, (c) Lap, i.e., a set of answers obtained by adding independent Laplace noise to the answers to the queries in Q_{12}, (d) dpc-Gauss, i.e., the synthetic dataset obtained through our Gaussian mechanism based differentially private Gaussian copula method, and (e) Gauss, i.e., a set of answers obtained by adding independent Gaussian noise to the answers to the queries in Q_{12}. Note that Lap and Gauss are not synthetic datasets. We set the same ϵ' for Lap, as we did for dpc-Lap, and the same ϵ and δ for Gauss as we did for dpc-Gauss.

Results. Figure 4 shows the absolute error CDF on the query set Q_{12} for cop, dpc-Lap, Lap, dpc-Gauss and Gauss versions constructed from the three datasets. For the set Q_1 (top row in figure), we can see that cop outperforms dpc-Lap, Lap, dpc-Gauss and Gauss. This is due to the fact that for privacy, a higher amount of noise is required. Crucially, our method does not introduce further error over the Laplace and Gaussian mechanisms, as is indicated by the similarity of the curves corresponding to dpc-Lap and Lap, and dpc-Gauss and Gauss. Interestingly, the results for Q_2 show that both dpc versions outperform independent Laplace and Gaussian noises (bottom row of Figure 4). While the errors from the dpc versions are still higher than cop, they are closer to it than the error due to Lap and Gauss. This indicates that for the majority of the queries, our method applies less noise than Lap and/or Gauss.

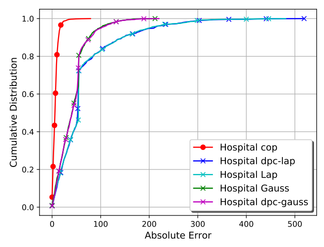
However, in some cases, for a small percentage of queries Lap/Gauss adds less noise than our mechanism. This is clear from Table 2, where we show the maximum and average error[11] from 95%, 99% and 100% percent of the queries from Q_{12} across dpc-Lap, Lap, dpc-Gauss and Gauss versions of the Adult, DSS and Hospital datasets. The error profiles of both dpc versions and Lap/Gauss variants are similar for the query set Q_1. For the set Q_2, we can see that Lap/Gauss only outperforms our method for the Adult and DSS datasets if we consider the maximum absolute error across all queries. On the other hand our method outperforms Lap/Gauss if we consider 95% and 99% of queries. Thus, for less than 1% of queries, the dpc versions of Adult and DSS exhibits less utility than Lap/Gauss.



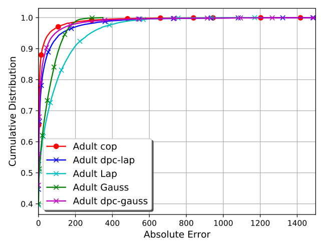
(a) Set Q_1 (Adult)



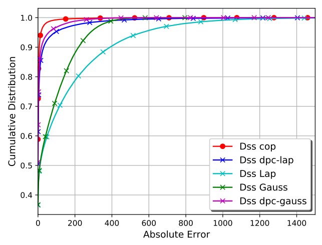
(b) Set Q_1 (DSS)



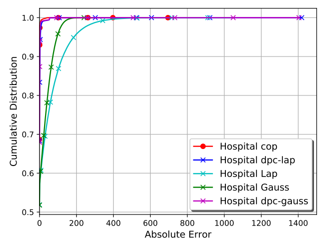
(c) Set Q_1 (Hopspital)



(d) Set Q_2 (Adult)



(e) Set Q_2 (DSS)



(f) Set Q_2 (Hopsital)

Figure 4: Absolute error over Q_{12} of one-way marginals (set Q_1) and two-way positive conjunctions (set Q_2) on synthetic versions of the Adult (left), DSS (middle) and Hospital (right) datasets with and without differential privacy.

4.4.3. *Results on Three-Way Conjunctions.* Even though our method is expected to perform best on the query set Q_{12}, we show that the method performs well on other types of queries as well. For this, we use the set of three-way positive conjunction queries as an example,

[11] Rounded to the nearest integer.

Mechanism	Adult						DSS						Hospital					
	95%		99%		100%		95%		99%		100%		95%		99%		100%	
	ave	max	ave	max	ave	max	ave	max	ave	max	ave	max	ave	max	ave	max	ave	max
dpc-Lap (one-way)	92	389	107	482	106	773	151	657	175	982	185	1267	53	208	61	302	65	622
Lap (one-way)	85	353	99	505	105	741	149	621	172	963	182	1287	52	199	61	299	61	887
dpc-Gauss (one-way)	75	203	84	278	85	336	142	349	161	410	167	528	50	101	66	144	74	220
Gauss (one-way)	70	177	80	235	81	319	138	344	155	407	162	472	48	98	65	144	71	212
dpc-Lap (two-way)	21	189	31	523	39	4788	9	112	17	429	26	5744	1	5	1	24	1	836
Lap (two-way)	60	321	74	539	80	1246	101	599	127	999	138	2947	33	201	41	336	44	1119
dpc-Gauss (two-way)	12	133	20	471	30	5822	4	68	9	232	13	3594	1	3	1	15	1	1048
Gauss (two-way)	35	158	40	214	42	349	58	294	69	414	70	795	18	100	23	140	24	287
dpc-Lap (three-way)	12	120	20	408	28	6148	9	98	16	372	36	9238	1	3	3	55	3	616
Lap (three-way)	102	589	128	1000	139	2746	268	1649	341	2798	373	10360	45	281	57	475	63	1679
dpc-Gauss (three-way)	10	95	16	371	24	7244	5	56	9	190	13	7429	1	2	3	55	3	639
Gauss (three-way)	65	292	76	408	80	722	168	821	200	1157	211	2372	27	138	33	196	35	358

Table 2: Absolute error α of the dpc and Lap/Gauss mechanisms on the three datasets. The columns show average and maximum values of α for 95%, 99% and 100% of the queries (corresponding to $\beta = 0.05, 0.01$ and 0.00, respectively). 🟧 indicates our method significantly outperforms Lap/Gauss; 🟪 indicates Lap/Gauss significantly outperforms our method; significance is defined as an error ratio of approximately 2 or more.

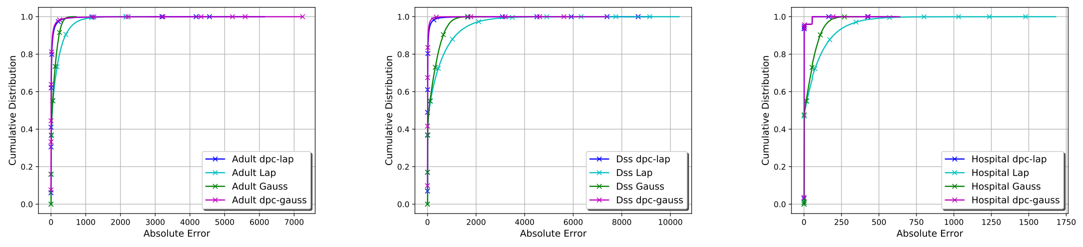
i.e., Q_3. We compare the error against the answers returned from (independent) Laplace mechanism by choosing an appropriate value of ϵ' for each query in Q_3 according to the advanced composition theorem (see Section 4.2), such that overall ϵ is just under 1 for queries in the set Q_3 only. Likewise, for the Gaussian mechanism we fix $\epsilon = 0.99$, and δ as before, and choose the l_2 sensitivity according to the number of queries in Q_3, giving us the scale of the Gaussian mechanism via Eq. 2.1. Note that for these experiments, both Laplace and Gaussian mechanisms do not compute answers to Q_{12}, and hence we do not waste the privacy budget on these queries. The results are shown in Figure 5. Once again our method outperforms the Laplace and Gaussian mechanisms for the majority of the queries in Q_3 for all three datasets. Looking closely, we see from Table 2, that Lap actually performs better than dpc-Lap in terms of maximum absolute error for 1% of the queries in the Adult dataset. However, for majority of the queries, $> 99\%$, dpc-Lap method performs better. In fact, dpc-Lap outperforms Lap in terms of the 95% and 99% error profiles for all three datasets. For the DSS dataset the maximum error from dpc-Lap over all queries is similar to Lap, whereas for the Hospital dataset we again outperform Lap. A similar trend can be seen between the Gauss and dpc-Gauss versions.

High Count Three-Way Positive Conjunctions. The above analysis on three-way counts is perhaps biased in favour of our approach, since most three-way counts are low. To analyze that our method still performs better than Lap and Gauss, we isolated those queries in Q_3 whose original answers are 100 or more. Note that there is nothing specific about 100, and the analysis yields similar results for limits of 500, 1000, etc. We chose the DSS dataset for this analysis, as it had the highest number of queries in Q_3 with answers larger than 100 (more than 178,000 queries). Table 3 shows the breakdown of the error for the Lap/Gauss and dpc variants in a manner similar to Table 2. As we can see, the trend from the latter table is more or less retained, with the dpc variants outperforming Lap/Gauss for 95% and 99% of the queries, and performing slightly worse at the 100% mark in terms of maximum error.

Mechanism	95%		99%		100%	
	ave	max	ave	max	ave	max
dpc-Lap	139	682	175	1906	204	9238
Lap	410	1714	479	2836	510	7708
dpc-Gauss	68	281	80	548	89	7429
Gauss	280	863	309	1180	320	2198

Table 3: Absolute error α of the dpc-Lap, Lap, dpc-Gauss and Gauss mechanisms on the DSS datasets on the query set Q_3 with original counts ≥ 100. The columns show average and maximum values of α for 95%, 99% and 100% of the queries (corresponding to $\beta = 0.05, 0.01$ and 0.00, respectively). 🟧 indicates our method significantly outperforms Lap/Gauss; 🟪 indicates Lap/Gauss significantly outperforms our method; significance is defined as an error ratio of approximately 2 or more.

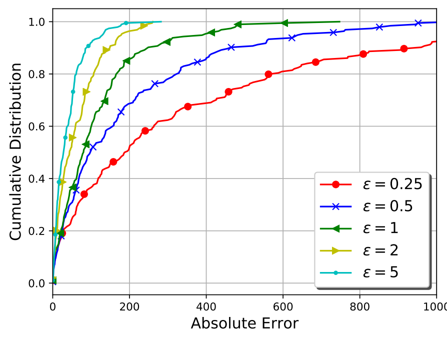
4.4.4. *Laplace versus Gauss.* From Table 2, it is clear that for all query sets Q_1, Q_2 and Q_3, the Gaussian mechanism has lower average and maximum error than the Laplace mechanism which is consistent with what is known about the two mechanisms under (ϵ, δ)-differential privacy, i.e., the noise through the Gaussian mechanism is more concentrated. As a result the Gaussian variant of our method performs better than the Laplace variant in almost all cases considered in Table 2. This suggests that much tighter analysis of the Gaussian mechanism using the notion of concentrated differential privacy, rather than (ϵ, δ)-differential privacy, would yield even better results through our method.



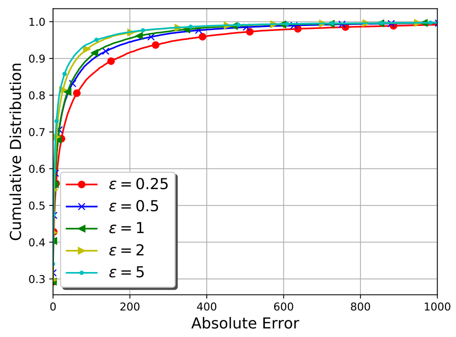
(a) Adult (b) DSS (c) Hospital

Figure 5: CDFs of the absolute error from our method against the Laplace and Gaussian mechanisms on the set Q_3 of three-way positive conjunctions on the three datasets.

4.4.5. *Effect of the Privacy Parameter.* To show the effect of ϵ on utility, we vary it from 0.25 to 5 and report the error on the set Q_{12}. For this, we use the dpc-Lap mechanism and only use the Adult dataset as the effect is similar on the other two datasets. Figure 6 shows the CDF of the absolute error against different values of ϵ. Notice that this is the overall privacy budget. With $\epsilon = 0.25$ we have average and maximum absolute errors of 357 and 3419, respectively, for the set Q_1, and 68 and 6421, respectively, for the set Q_2. With $\epsilon = 5$, the average and maximum absolute errora are much lower at 41 and 179, respectively, for Q_1, and 27 and 5882, respectively, for the set Q_2. As expected, the error profiles gradually improve as we move from $\epsilon = 0.25$ to $\epsilon = 5$. "Compared to the set Q_1, the error profiles are more similar for the set Q_2. As discussed in Section 4.4.1, since the majority of attributes are uncorrelated, this implies that our method maintains that aspect by not adding too much noise on the set Q_2.



(a) Set Q_1



(b) Set Q_2

Figure 6: Absolute error on the set Q_{12} against different values of ϵ on the Adult dataset using the dpc-Lap variant.

4.4.6. *Computational Time and Parallelism.* One of the motivations for using the proposed approach is its computational feasibility when the input data is high dimensional. We first note that our method is highly parallelizable. In particular, the computation of one-way and two-way positive conjunctions can be done in parallel. For the one-way marginals, parallel computation is straightforward. For the two-way conjunctions, we take the ith attribute (in the original dataset) and compute its conjunction with all attributes numbered $i + 1$ to m attributes, for all $i \in [1, m - 1]$, assigning a separate process for each i. Obviously, the number of combinations for the first attribute is the highest, and becomes progressively less for latter attributes. Likewise, we also parallelize the computation of $\binom{d}{2} - d$ Gaussian correlations $\rho_{i,j}$ which uses a bisection search. While other components of our method can also be executed in parallel, e.g., generating synthetic records through the copula, we do not do so as these processes did not consume much computational time.

To generate the synthetic datasets we used a single-CPU Intel Xeon E5-2660 2.6GHz server with 10 cores and 128GB memory. Our implementation was done in `Python`.[12] We parallelized part of our mechanism, as described above. The average run-times (over 10 runs) for the three datasets Adult, DSS and Hospital, are shown in Table 4. Obviously, the run-time is a function of the parameters m (number of original attributes), d (number of binary attributes) and n (the number rows in the dataset). Asymptotically, the run-time of our method is $O(m^2n + d^{2.38})$. The DSS dataset takes the longest time, which is understandable since it is about 3 times bigger in terms of the number of binary attributes and has 150 times more rows than the Adult dataset. If the number of rows is not large, then the run-time is not severely impacted by an increase in the number of binary attributes, as is indicated by the run-times of the Hospital dataset. To further assess the scalability of our algorithm, we constructed two artificial datasets Art1 and Art2 with 1,000 and 2,000 binary attributes,

[12] https://www.python.org/

Mechanism	Rows	Attributes		Ave. Time	Runs
		Original	Binary		
Adult	32,560	14	194	6 min 47 sec (±32 secs)	10
DSS	5,240,260	27	674	2 h 30 min (±15 mins)	10
Hospital	10,000	9	1,201	46 min 21 sec (±11 mins)	10
Art1	5,240,260	1,000	1,000	5 h 43 min (±4 mins)	2
Art2	5,240,260	2,000	2,000	18 h 32 min (±12 mins)	2

Table 4: Run-time.

respectively. Both had the same number of rows as the DSS dataset. The run times of these two datasets are shown under Art1 and Art2 in Table 4. By far, Art2 is the largest dataset, and even with this dataset we can generate a private synthetic dataset via our method in around 19 hours. We stress that since privacy-preserving synthetic datasets need only be produced once, these times are practical. Thus, our method can output a privacy-preserving synthetic datasets of high dimensional datasets in reasonable time.

5. Comparison with DPCopula

The closest work to ours is that of Li et al. [2014], who propose DPCopula. DPCopula also uses the Gaussian copula to generate differentially private synthetic datasets. However, we claim that our method is more general and efficient due to four major differences between our work and theirs. This section details these differences, which are on the treatment of categorical attributes, small domain attributes, correlation matrix, and positive definite matrix. To further support our claim, we also implemented the Kendall version of DPCopula (i.e., one of the two equivalent versions discussed in Li et al. [2014]) and experimentally compared DPCopula-Kendall against our own method, using the Adult dataset. This comparison showed that our method yields higher utility for the Q_1 and Q_2 set of queries.

5.1. **Categorical Attributes.**

DPCopula imposes an order on the values of any categorical (nominal) attributes in the input data set. However, we argue that this order is inherently artificial and arbitrary: different choices of order for categorical attributes produce different pair-wise correlations between them. This in turn affects the accuracy of two-way conjunctions computed on data generated through the Gaussian copula using these pair-wise correlations. A simple example illustrates our point. Consider a database having two attributes X ("country of birth") and Y ("marital status") with possible values (`English`, `Chinese`, `French`) and (`Married`, `Divorced`, `Widowed`), respectively. For the sake of simplicity, assume that the dataset consists of 400 records with 100 pairs of (`English`, `Married`), 200 pairs of `Chinese`, `Divorced`) and 100 pairs of (`French`, `Widowed`). To calculate Pearson correlation between X and Y, let us fix the numerical map $(\texttt{English}, \texttt{Chinese}, \texttt{French}) \rightarrow (1, 2, 3)$ on attribute X. Consider first the numerical map $(\texttt{Married}, \texttt{Divorced}, \texttt{Widowed}) \rightarrow (1, 2, 3)$ on attribute Y. The Pearson correlation between X and Y in this case is exactly 1. However, notice that there is no logical reason to choose any of the two maps. If we change the second map to the (equally valid) map $(\texttt{Married}, \texttt{Divorced}, \texttt{Widowed}) \rightarrow (3, 1, 2)$, the resulting correlation becomes ≈ -0.457. If we use the resulting correlations to generate synthetic outputs via the Gaussian copula, we obtain drastically different results on the two-way counts. A simple program in `R` results[13] in the two-way counts $\#(\texttt{English}, \texttt{Married}) = 112$, $\#(\texttt{Chinese}, \texttt{Divorced}) = 193$, and $\#(\texttt{French}, \texttt{Widowed}) = 95$ for the first map. The second map results in the counts $\#(\texttt{English}, \texttt{Married}) = 47$, $\#(\texttt{Chinese}, \texttt{Divorced}) = 35$ and $\#(\texttt{French}, \texttt{Widowed}) = 46$, from the synthetic output. This simple example illustrates the impact of an arbitrary order on correlations between categorical attributes in the original data set. While the first ordering gives good results, the second ordering gives noticeably bad results. The reason why the first ordering gives good results is mainly an artefact of the simplicity of illustration. With more attributes, where multiple inter-attribute correlations need to be determined, a utility maximizing ordering across all categorical attributes may not be straightforward. Appendix A gives a more analytical treatment on the impact of changing orders (maps) on the correlation. As opposed to DPCopula, our proposed method (Section 3) does not rely on arbitrary orders for nominal attributes. Thus our method is capable of producing synthetic datasets with pair-wise attribute correlations that are closer to the ones in the original dataset. This claim is further supported by the experimental comparison between DPCopula and our method in Section 5.5 (to follow).

5.2. **Small Domain Attributes.** DPCopula is designed only for attributes with large domains, i.e., attributes which have at least 10 different values [Li et al., 2014, §4.4]. For small domain (including binary attributes) a method called DPHybrid is proposed [Li et al., 2014] which partitions the data into smaller datasets (one per attribute value in the small domain attributes) and then generates separate synthetic datasets per partition using Gaussian copulas, before eventually combining them. First, if the dataset has only small domain attributes then DPCopula or its hybrid variant cannot be used. Secondly, depending on the number of small domain attributes DPHybrid can become computationally infeasible, i.e., taking time exponential in the number of small domain attributes. For instance, in our DSS dataset, we have a total of 10 small domain attributes (having number of values less than 10) totalling approximately 2^{18} partitions (product of attribute values). Thus, the time to produce the combined synthetic dataset is 2^{18} times the time to produce individual synthetic datasets via the Gaussian copula for each partition; which itself takes time $O(m'^2 n)$, where m' is the number of large domain attributes (17 in the DSS dataset). This amounts to roughly 2^{50} time to generate a synthetic dataset from the DSS dataset. With more small domain attributes, this is bound to increase.

5.3. **Correlation Matrix.** A third major difference between our work and DPCopula is in the process to generate the differentially private correlation matrix, i.e., $\mathbf{P}'$ in Eq. 3.10. There are two methods described by Li et al. [2014] to generate the counterpart to $\mathbf{P}'$. The first method uses Kendall's rank correlation coefficient τ [Kendall, 1938] to measure correlations between attributes in the original dataset and then uses the relation $\mathbb{E}(\tau) = \frac{2}{\pi} \sin \rho$ to obtain the correlation coefficient ρ between the corresponding normal random variables. A differentially private variant is constructed by showing that τ has low global sensitivity [Li et al., 2014]. We first note that the relation $\mathbb{E}(\tau) = \frac{2}{\pi} \sin \rho$ is proven for continuous random variables [Xu et al., 2013, §3.2],[Esscher, 1924]. This is one reason why DPCopula is targeted for continuous data or at least large domain discrete attributes (approximated as continuous attributes). Secondly Kendall's rank correlation coefficient, as the name suggests, assumes an order between attributes; once again, as argued before, for categorical attributes this

[13]This is done using the `rCopula` function from the `copula` package for `R` Hofert et al. [2017].

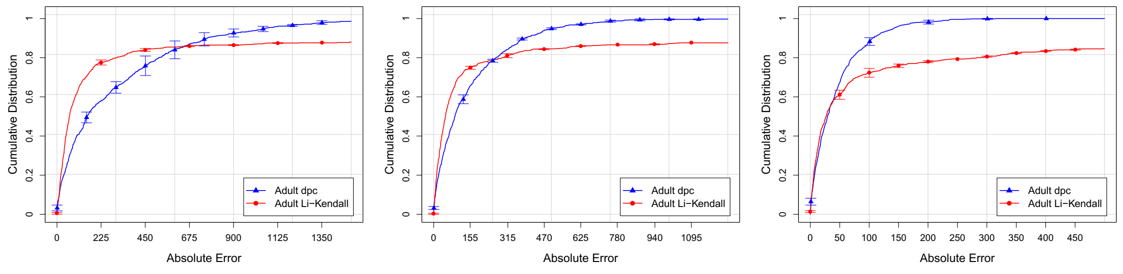
means that an artificial order needs to be induced which is not reflective of the correlations. Since we convert data into a binary format, there is no meaningful rank between two binary variables that could be used to compute Kendall's τ coefficient. Furthermore, the conversion $\tau = \frac{2}{\pi} \sin \rho$ would not apply as well. The second method used by DPCopula is a maximum likelihood estimation method to compute ρ using a known "sample-and-aggregate" method [Nissim et al., 2007, Dwork and Smith, 2010]. This method involves partitioning the dataset into n/l partitions and then adding Laplace noise of scale $2\binom{m}{2}/l\epsilon$ to each of the $\binom{m}{2}$ pairs of attributes. Since our data is in binary format, we would need to add noise of scale $2\binom{d}{2}/l\epsilon$. If we do not want the noise to overwhelm the calculation of ρ, we need l to be at least $\binom{d}{2}$. Unfortunately, this means that we would have the partitions of size much smaller than $\sqrt{n}$ for all three datasets considered in this paper, which is needed for a good approximation of ρ's [Dwork and Smith, 2010, §3.1.2, p. 145]. We therefore use a different method for constructing the differentially private correlation matrix by adding noise to the margins before obtaining Pearson product-moment correlations and then using a bisection search to convert to the corresponding correlations for Gaussian variables.

5.4. **Positive Definite Matrix.** To be able to use the Cholesky decomposition (cf. Section 3.3.2), DPCopula uses a heuristic method to obtain a positive definite matrix with unitary diagonals [Rousseeuw and Molenberghs, 1993, Li et al., 2014]. This procedure first finds the eigen decomposition of the matrix $\mathbf{P}$, i.e., the matrix of Gaussian correlations from the differentially private correlation matrix of input data (see Section 3.3.1). It then fixes the negative eigenvalues to a small value or the absolute value, and finally normalizes the resulting matrix to turn it into a correlation matrix. This indeed returns a valid correlation matrix. However, this heuristic method does not guarantee that its result is the nearest correlation matrix to the input matrix $\mathbf{P}$. In contrast, our own method is using the algorithm from Higham [2002], to obtain a positive definite matrix with unitary diagonals. This algorithm guarantees that the resulting matrix is **the nearest** correlation matrix to the input matrix, as defined by a given matrix norm. Having the nearest possible positive definite correlation matrix as the input in the next stage of the Copula-based data generation allows for higher utility synthetic datasets, as shown in the following experimental comparison.

5.5. **Experimental Comparison.** As mentioned earlier, we implemented the Kendall version of DPCopula in Python.[14] We decided to implement the Kendall variant, as it is the one that Li et al. [2014] used to compare against other private synthetic data generation methods. We performed a series of experiments to compare the Kendall variant of DPCopula, henceforth called "Li-Kendall," to our own method "dpc-Lap", i.e., the variant that uses Laplace noise. For brevity, only the results using the Adult dataset are presented here, as using the DSS and Hospital datasets yielded similar results. Furthermore, since the mechanisms in [Li et al., 2014] are pure differentially private, i.e., with $\delta = 0$, we also use $\delta = 0$ for dpc-Lap.

We used the same value of ϵ for both methods. For dpc-Lap, we used $\epsilon' = \epsilon/m'$ privacy budget per invocation of the Laplace mechanism, where $m' = m + \binom{m}{2}$. This breakdown is over the query set Q_{12} over which we chose to evaluate the two mechanisms. In the case of Li-Kendall, ϵ is further split into two components ϵ_1 and ϵ_2 [Li et al., 2014]. The first

[14]Our implementation is available for peer-review at: `https://github.com/thierryr/dpcopula_kendall`.



(a) $\varepsilon = 0.5$ (b) $\varepsilon = 1$ (c) $\varepsilon = 5$

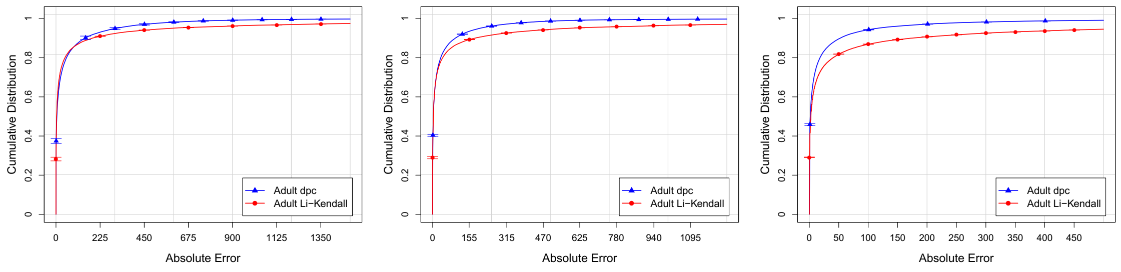
Figure 7: CDFs of the absolute error from our method (dpc) against the Li-Kendall mechanism from Li et al. [2014] on the set Q_1 of one-way marginals for the Adult dataset and different values of ε.

is consumed on computing the one-way marginals, and the second is used to generate the correlation matrix. Following Li et al. [2014], we let $k = \epsilon_1/\epsilon_2$ denote the ratio of the budget splits, and use the same value of $k = 8$ as used in their own experimental evaluations. We execute both Li-Kendall and dpc-Lap four times each on the Adult dataset, resulting in 4 different synthetic outputs each. Furthermore, for each run of the Li-Kendall method, we purposely used a different random order for the values of any categorical (nominal) attributes in the Adult dataset. Indeed, as described in Section 5.1, the Li-Kendall method imposes an arbitrary order for such attributes, which affects subsequently computed attribute correlations. Next, we executed the full set of queries Q_1 and Q_2 on each resulting datasets for each of the two methods. Finally, we computed the absolute error for each of these query results against the same query executed on the original Adult dataset. This series of experiments were then repeated for three different values of $\epsilon \in \{0.5, 1.0, 5.0\}$.

Figure 7 shows the empirical cumulative distribution of the absolute error for the Q_1 queries (computed as described above). On these graphs, each point is the average over the 4 different trials for each method, and the error bars represent the standard deviation over these same trials. Figure 7 shows that dpc-Lap outperforms the Li-Kendall method in terms of absolute errors on the set Q_1 of one-way marginal for $\epsilon = 5$. Our dpc method outperforms Li-Kendall in about 17% and 20% of the queries for $\epsilon = 0.5$ and $\epsilon = 1$, respectively, while still being extremely close to Li-Kendall for all the remaining 80% when $\epsilon = 1$.

Similarly, Figure 8 shows the empirical cumulative distribution of the absolute error for the Q_2 queries, again as average and standard deviation over all the trials. This figure shows that our dpc method consistently outperforms the Li-Kendall method in terms of absolute errors on the set Q_2 of two-way positive conjunctions, for all used values of ϵ. Table 5 also provides the detailed error profiles in terms of average and maximum absolute errors of 90%, 95%, 99% and 100% of the queries for both our dpc and the Li-Kendall methods. These are averaged our 4 runs as explained before. As we can see, in the majority of cases our method significantly outperforms Li-Kendall, where the latter has more than twice the error. This demonstrates experimentally that our dpc method produces differentially private synthetic datasets which have closer pair-wise attribute correlations to the original input dataset, than the ones produced by the Kendall variant of DPCopula.

The better performance of our dpc method over the Kendall-based DPCopula is due to the differences in both methods, which were highlighted earlier in Section 5.1 through to



(a) $\varepsilon = 0.5$ (b) $\varepsilon = 1$ (c) $\varepsilon = 5$

Figure 8: CDFs of the absolute error from our method (dpc) against the Li-Kendall mechanism from Li et al. [2014] on the set Q_2 of two-way positive conjunctions for the Adult dataset and different values of ε.

ϵ	Query Set	Mechanism	90%		95%		99%		100%	
			ave	max	ave	max	ave	max	ave	max
0.5	Q_1	dpc-Lap	256	552	276	706	297	857	303	920
		Li-Kendall	147	2370	373	6812	1044	29918	1349	30874
	Q_2	dpc-Lap	23	160	33	285	48	629	59	6334
		Li-Kendall	16	186	33	581	93	4156	192	28385
1.0	Q_1	dpc-Lap	132	285	142	339	153	463	156	477
		Li-Kendall	131	2328	357	6809	1028	29935	1334	30926
	Q_2	dpc-Lap	14	103	21	185	31	472	42	6249
		Li-Kendall	15	180	31	573	92	4144	190	28394
5.0	Q_1	dpc-Lap	38	80	41	94	44	147	45	156
		Li-Kendall	118	2360	345	6813	1018	30001	1325	30897
	Q_2	dpc-Lap	7	53	11	110	18	429	28	5703
		Li-Kendall	14	173	30	554	90	4151	189	28519

Table 5: Average absolute error α of our method (dpc-Lap) versus Li-Kendall from Li et al. [2014] on the Adult on the query set Q_{12}. The columns show average and maximum values of α for 90%, 95%, 99% and 100% of the queries (corresponding to $\beta = 0.10, 0.05, 0.01$ and 0.00, respectively). 🟧 indicates our method significantly outperforms Li-Kendall; significance is defined as an error ratio of approximately 2 or more.

Section 5.4. Indeed, as our method does not impose arbitrary order to categorical attributes, and at the same time computes a positive definite matrix closer to the original correlation matrix, then the final pair-wise correlations in the resulting synthetic dataset are indeed more similar to the original dataset.

6. Other Related Works

In line with the theme of the paper, we restrict our review of related work to proposals for generating differentially private synthetic datasets. We divide this into two main categories. The first consists of mechanisms that provide *provable* utility guarantees. The second is a class of algorithms that claims high utility in practice possibly relying on assumptions on the distribution of the input dataset, which we call heuristic approaches. Our method lies in this class. We review the two classes in order.

One way to release a synthetic dataset is to add (independent) Laplace noise to all point functions (Definition 2) from the input domain $\mathcal{X}$ [Dwork et al., 2006b, Dwork and Roth, 2014]. The resulting dataset gives good answers to point functions but lower order margins are noisier. However, the main problem with this approach is that its runtime is $O(|\mathcal{X}|)$ which is exponential in the number of attributes; hence, its inapplicability to high dimensional datasets. The stability-based histogram algorithm [Bun et al., 2016, Balcer and Vadhan, 2018, Vadhan, 2017] runs in time only $O(\log |\mathcal{X}|)$ by using the notion of local sensitivity and relying on approximate differential privacy. However, for high dimensional datasets it is likely that the output synthetic datasets will only contain a fraction of the original point functions (Definition 2), due to a high percentage of rows being unique or having low multiplicity in a high dimensional dataset.

For a more general class of counting queries, i.e., not necessarily point functions, the BLR algorithm [Blum et al., 2008] and the MWEM algorithm [Hardt et al., 2012] allow answers to exponentially many queries with noise per query proportional to $n^{2/3}$ and $n^{1/2}$, respectively (n being the number of rows). However, these algorithms are not efficient as both require time polynomial in $|\mathcal{X}|$. This makes these algorithms inefficient for high dimensional datasets. The drawback of exponential runtime (in the number of attributes) is also present in the mechanism from Dwork et al. [2009], the median mechanism from Roth and Roughgarden [2010], and the matrix mechanism from Li et al. [2010] to name a few. For instance, Privlet [Xiao et al., 2011], which can be categorised as an instance of the matrix mechanism, is designed to answer range queries by first creating a full contingency table (frequency matrix) of the input datasets. This is obviously exponential in the number of attributes of the dataset.

Computational inefficiency is not surprising since any synthetic data generation mechanism that answers an arbitrary number of counting queries, or even the set of all two-way marginals, is expected to run in exponential time under the hardness assumption of some well known cryptographic primitives [Ullman and Vadhan, 2011, Ullman, 2013]. However, algorithms that run in exponential-time in theory, might still be efficient in practice. The DualQuery algorithm [Gaboardi et al., 2014] is one such algorithm, which approximates a set of given counting queries, say three-way marginals, to within $n^{2/3}$ (absolute) error. The algorithm requires solving an optimization problem, which is hard in theory but solvable in practice for large parameters using standard optimization software. Likewise, the MWEM algorithm can run in reasonable time for a large number of attributes (up to 77 binary attributes) in practice [Hardt et al., 2012]. Both approaches suggest further improvement in run-time using heuristics.

This leads us to the heuristic approaches for synthetic data release. Unlike the above mentioned class of algorithms, this class does not provide a provable utility guarantee and is often accompanied with some heuristic assumption on the input data distribution; crucially, for utility guarantees and not for privacy. As long as the heuristics hold true, the algorithm is expected to produce a synthetic dataset with good utility. The private spatial decomposition technique from Cormode et al. [2012] decomposes the input dataset into a hierarchical tree and then answers range queries over this structure. The technique is relevant to spatial data, and does not seem generic enough to consider categorical variables. As we discussed earlier, this requires fixing an artificial order on categorical variables which can be completely arbitrary. PrivBayes is another algorithm [Zhang et al., 2014] which constructs a Bayesian network of an input dataset. The Bayesian network maintains attribute correlations and approximates the data distribution as a set of low dimensional marginals. Efficiency is guaranteed so long as the degree of the network is low, where degree is roughly defined as the maximum number of attributes a given attribute depends on in the Bayesian network. The obvious assumption is that most correlations in the input datasets are of low degree. DiffGen [Mohammed et al., 2011] proposes a generalization based approach for releasing data where a hierarchical tree is first constructed and a table corresponding to a given generalization level (in the tree) is released where the generalization level itself is decided by maximising utility through the exponential mechanism McSherry and Talwar [2007]. This implies that the level of generalization of the output data is randomized, thus resulting in different utility on each invocation. This can be a drawback from a usability point-of-view if two datasets on the same domain but, say, different time periods are to be released, each resulting in a different level of generalization. The algorithm also runs in time exponential in the number of attributes.

7. Conclusion

We have presented a generic mechanism to efficiently output differentially private synthetic datasets with high utility using the concept of Gaussian copulas. Our method is generic; while Gaussian copulas are mostly used to generate (non-private) synthetic datasets for numerical attributes, our methods is applicable to both numerical and categorical attributes alike. The proposed mechanism is efficient as it takes time polynomial in the number of attributes, in contrast to exponential time required by many differentially private synthetic data generation algorithms, which makes our algorithm suitable for high-dimensional datasets. Through experiments on three real-world datasets, we have shown that our mechanism provides high utility, matching and even surpassing the utility provided by independent noise through the Laplace and Gaussian mechanisms, and by another existing copula-based mechanism. A shortcoming of our work is the lack of a provable utility guarantee. Nonetheless, we have provided significant experimental evidence of utility. A future direction is to provide theoretical guarantees of utility, perhaps by assuming certain characteristics of the distribution of the input dataset which may make the analysis tractable. A further interesting direction is to assess if other copulas found in literature could also be used to efficiently generate synthetic datasets with high utility.

References

V. Balcer and S. Vadhan. Differential privacy on finite computers. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. doi: 10.4230/LIPIcs.ITCS.2018.43. URL `https://drops.dagstuhl.de/opus/volltexte/2018/8353/`.

M. Bardet, J.-C. Faugere, and B. Salvy. *Complexity of Gröbner basis computation for Semi-regular Overdetermined sequences over F_2 with solutions in F_2*. PhD thesis, INRIA, 2003. URL `https://hal.inria.fr/inria-00071534/file/RR-5049.pdf`.

A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 609–618, New York, NY, USA, 2008. Association for Computing Machinery.

ISBN 9781605580470. doi: 10.1145/1374376.1374464. URL https://doi.org/10.1145/1374376.1374464.

M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016. doi: 10.1007/978-3-662-53641-4_24. URL https://doi.org/10.1007/978-3-662-53641-4_24.

M. Bun, K. Nissim, and U. Stemmer. Simultaneous private learning of multiple concepts. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, pages 369–380, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4057-1. doi: 10.1145/2840728.2840747. URL http://doi.acm.org/10.1145/2840728.2840747.

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. doi: 10.1109/ICDE.2012.16. URL https://doi.org/10.1109/ICDE.2012.16.

H. Cramér. *Mathematical Methods of Statistics (PMS-9)*, volume 9. Princeton university press, 2016. doi: 10.1515/9781400883868. URL https://doi.org/10.1515/9781400883868.

C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

C. Dwork and G. N. Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016. URL https://arxiv.org/abs/1603.01887.

C. Dwork and A. Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2):2, 2010. doi: 10.29012/jpc.v1i2.570. URL https://doi.org/10.29012/jpc.v1i2.570.

C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006a. doi: 10.1007/11761679_29. URL https://doi.org/10.1007/11761679_29.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg, 2006b. Springer-Verlag. ISBN 3-540-32731-2, 978-3-540-32731-8. doi: 10.1007/11681878_14. URL http://dx.doi.org/10.1007/11681878_14.

C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390. ACM, 2009. doi: 10.1145/1536414.1536467. URL https://doi.org/10.1145/1536414.1536467.

C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 51–60, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4244-7. doi: 10.1109/FOCS.2010.12. URL http://dx.doi.org/10.1109/FOCS.2010.12.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. Differential Privacy: A Primer for the Perplexed. In Joint UNECE/Eurostat work session on statistical data confidentiality https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.46/2011/26_Dwork-Smith.pdf, 2011.

F. Esscher. On a method of determining correlation from the ranks of the variates. *Scandinavian Actuarial Journal*, 1924(1):201–219, 1924. doi: 10.1080/03461238.1924.10405384. URL https://doi.org/10.1080/03461238.1924.10405384.

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 *Proceedings of the 31st International Conference on Machine Learning, Cycle 2*, volume 32 of *JMLR Proceedings*, pages 1170–1178. JMLR.org, 2014. URL http://proceedings.mlr.press/v32/gaboardi14.html.

M. Gaboardi, J. Honaker, G. King, J. Murtagh, K. Nissim, J. Ullman, and S. Vadhan. Psi (ψ): a private data sharing interface. *arXiv preprint arXiv:1609.04340*, 2016. URL https://arxiv.org/abs/1609.04340.

G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 4. JHU Press, 2013. URL https://jhupbooks.press.jhu.edu/title/matrix-computations.

M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *NIPS*, pages 2348–2356, 2012. URL https://papers.nips.cc/paper/4548-a-simple-and-practical-algorithm-for-differentially-private-data-release.

N. J. Higham. Computing the nearest correlation matrix-a problem from finance. *IMA journal of Numerical Analysis*, 22(3):329–343, 2002. doi: 10.1093/imanum/22.3.329. URL https://doi.org/10.1093/imanum/22.3.329.

M. Hofert, I. Kojadinovic, M. Maechler, and J. Yan. Package 'copula', 2017. URL https://cran.r-project.org/web/packages/copula/copula.pdf.

V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 279–288. ACM, 2002. doi: 10.1145/775047.775089. URL https://doi.org/10.1145/775047.775089.

M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. doi: 10.1093/biomet/30.1-2.81. URL https://doi.org/10.1093/biomet/30.1-2.81.

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*, PODS '10, pages 123–134, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0033-9. doi: 10.1145/1807085.1807104. URL http://doi.acm.org/10.1145/1807085.1807104.

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. doi: 10.5441/002/edbt.2014.43. URL https://doi.org/10.5441/002/edbt.2014.43.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, volume 7, pages 94–103, 2007. doi: 10.1109/FOCS.2007.41. URL https://doi.org/10.1109/FOCS.2007.41.

F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 19–30, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-551-2. doi: 10.1145/1559845.1559850. URL http://doi.acm.org/10.1145/1559845.1559850.

N. Mohammed, R. Chen, B. Fung, and P. S. Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge* *discovery and data mining*, pages 493–501. ACM, 2011. doi: 10.1145/2020408.2020487. URL https://doi.org/10.1145/2020408.2020487.

A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 111–125. IEEE, 2008. doi: 10.1109/SP.2008.33. URL https://doi.org/10.1109/SP.2008.33.

R. B. Nelsen. *An Introduction to Copulas (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387286594. doi: 10.1007/0-387-28678-0. URL https://doi.org/10.1007/0-387-28678-0.

B. Nelson. *Foundations and Methods of Stochastic Simulation: A First Course*. Springer Publishing Company, Incorporated, 2015. ISBN 1489989811, 9781489989819. doi: 10.1007/978-1-4614-6160-9. URL https://doi.org/10.1007/978-1-4614-6160-9.

K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-631-8. doi: 10.1145/1250790.1250803. URL http://doi.acm.org/10.1145/1250790.1250803.

P. Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *Ucla L. Rev.*, 57:1701, 2009. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1450006.

A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 765–774. ACM, 2010. doi: 10.1145/1806689.1806794. URL https://doi.org/10.1145/1806689.1806794.

P. J. Rousseeuw and G. Molenberghs. Transformation of non positive semidefinite correlation matrices. *Communications in Statistics–Theory and Methods*, 22(4):965–984, 1993. doi: 10.1080/03610928308831068. URL https://doi.org/10.1080/03610928308831068.

M. Sklar. Fonctions de repartition an dimensions et leurs marges. *Publ. Inst. Statist. Univ. Paris*, 8:229–231, 1959.

J. Ullman. Answering n2+O(1) counting queries with differential privacy is hard. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 361–370, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2029-0. doi: 10.1145/2488608.2488653. URL http://doi.acm.org/10.1145/2488608.2488653.

J. Ullman and S. Vadhan. Pcps and the hardness of generating private synthetic data. In *Proceedings of the 8th Conference on Theory of Cryptography*, TCC'11, pages 400–416, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-19570-9. URL http://dl.acm.org/citation.cfm?id=1987260.1987292.

S. Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017. doi: 10.1007/978-3-319-57048-8_7. URL https://doi.org/10.1007/978-3-319-57048-8_7.

X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. on Knowl. and Data Eng.*, 23(8):1200–1214, Aug. 2011. ISSN 1041-4347. doi: 10.1109/TKDE.2010.247. URL http://dx.doi.org/10.1109/TKDE.2010.247.

W. Xu, Y. Hou, Y. Hung, and Y. Zou. A comparative analysis of spearman's rho and kendall's tau in normal and contaminated normal models. *Signal Processing*, 93(1):261–276, 2013. doi: 10.1016/j.sigpro.2012.08.005. URL https://doi.org/10.1016/j.sigpro.2012.08.005.

J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. In *Proceedings of the 2014 ACM SIGMOD International*

Conference on Management of Data, SIGMOD '14, pages 1423–1434, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2376-5. doi: 10.1145/2588555.2588573. URL http://doi.acm.org/10.1145/2588555.2588573.

Appendix A. Artificial Order Disrupts Pearson Correlation

Consider two categorical attributes taking n values each over a database of size n. Let us assign the sequential order $1, \ldots, n$ to the n values of the first attribute (say based on lexicographical order). Let us define another order on the second attribute in which the order of the first λn values are reversed. The remaining $n - \lambda n$ values retain the sequential order, where $\lambda \in [0, 1]$. For instance, $\{3, 2, 1, 4, 5\}$ is the order on the second attribute with $\lambda = 0.6$, i.e., the first 3 values have a reverse order. Note that the mean μ is the same for both orders, given by $\mu = \frac{n+1}{2}$. Let r_λ be the correlation coefficient between the two attributes, and let y_i denote the ith value in the second attribute. Then

$$r_\lambda = \frac{\sum_{i=1}^{n}(i-\mu)(y_i-\mu)}{\sqrt{\sum_{i=1}^{n}(i-\mu)^2 \sum_{i=1}^{n}(y_i-\mu)^2}}. \tag{A.1}$$

Now consider the denominator in the above. After simplification, we get

$$\begin{aligned} \text{den} &= \sqrt{\sum_{i=1}^{n}(i-\mu)^2 \sum_{i=1}^{n}(y_i-\mu)^2} \\ &= \sqrt{\sum_{i=1}^{n}(i-\mu)^2 \sum_{i=1}^{n}(i-\mu)^2} \\ &= \frac{n(n+1)(n-1)}{12}. \end{aligned} \tag{A.2}$$

Consider now the numerator, which after simplification gives

$$\text{num} = \sum_{i=1}^{n} i y_i - \frac{n(n+1)^2}{4}. \tag{A.3}$$

Now let r_0 be the correlation when the two attributes have the same order, i.e., $\lambda = 0$. We are interested in finding $r_0 - r_\lambda$ as a function of λ, where different values of λ indicate the level of change in the order. Using the fact that $y_i = i$, when $\lambda = 0$, through Eqs. A.2 and A.3 we obtain

$$\begin{aligned} r_0 - r_\lambda &= \frac{1}{\text{den}} \cdot \sum_{i=1}^{n}(y_i - i)i \\ &= \frac{1}{\text{den}} \cdot \sum_{i=1}^{\lambda n}(\lambda n - i + 1 - i)i \\ &= \frac{1}{\text{den}} \left((\lambda n + 1)\left(\sum_{i=1}^{\lambda n} i\right) - 2\left(\sum_{i=1}^{\lambda n} i^2\right)\right) \\ &= \frac{1}{\text{den}} \frac{\lambda n(\lambda n+1)(\lambda n-1)}{6} \\ &= \frac{12}{n(n+1)(n-1)} \frac{\lambda n(\lambda n+1)(\lambda n-1)}{6} \\ &= 2\lambda^3 \left(1 - \frac{\lambda^{-1}-1}{n+1}\right)\left(1 - \frac{\lambda^{-1}-1}{n-1}\right), \end{aligned} \tag{A.4}$$

where $\lambda \neq 0$ in the last equality. Thus, for instance if $\lambda = 0.5$, we get $r_0 - r_{0.5} \approx \frac{1}{4}$. And when $\lambda = 1$, i.e., complete reversal of order, we get the difference as 2. Since $r_0 = 1$, this means that $r_1 = -1$, a complete reversal in correlation.

Appendix B. Proof of Lemma 1

Note that conversion of A into X_j's creates $|A|$ distinct partitions of the domain $\mathcal{X}$. From the parallel composition theorem, i.e., Theorem 2, since each marginal is computed with $(\epsilon'_i, 0)$-differential privacy, the overall differential privacy guarantee remains $(\epsilon'_i, 0)$.

Another way of looking at this is as follows. Suppose, instead of converting A into binary attributes, we compute its marginal distribution directly from the histogram of the values A takes, where each histogram bin corresponds to the number of occurrences of a unique value of attribute A. Since each row of D can only be in one of the bins, the private version of the histogram can be obtained by adding Laplace noise of scale $2/\epsilon'_i$ to each count, and then publishing the counts. The resulting mechanism remains $(\epsilon'_i, 0)$-differentially private [Dwork and Roth, 2014, §3.3, p. 33]. Now, we can convert A to binary attributes and deduce the marginals of these binary attributes from the histogram counts. This does not further impact privacy, as it is simply post-processing (See Theorem 4).

Appendix C. Pearson Correlation over Binary Attributes has High Global Sensitivity

Consider an n-row binary database D_1 having two attributes X and Y with only the first entry in each attribute set to 1 and the rest to 0, i.e., $X_1 = Y_1 = 1$ and $X_i = Y_i = 0$ for all $i \in \{2, \ldots, n\}$. Consider the neighbouring database D_2 which is the same as D_1 except that $X_2 = 1$. Let r_1 be the correlation coefficient between x and y in D_1, and let r_2 be its counterpart in D_2. We will show that $r_1 - r_2$ is large, meaning that the correlation coefficient has high global sensitivity and any noise scaled to the sensitivity of the correlation coefficient will overwhelm the accuracy of the results. Let $\overline{X}$ and $\overline{Y}$ denote the mean of the attributes X and Y, respectively. We have

$$\begin{aligned} r_1 &= \frac{\sum_{i=1}^{n} X_i Y_i - n\overline{X} \cdot \overline{Y}}{\sqrt{\sum_{i=1}^{n} X_i^2 - n\overline{X}^2}\sqrt{\sum_{i=1}^{n} Y_i^2 - n\overline{Y}^2}} \\ &= \frac{1 - n \cdot \frac{1}{n}\frac{1}{n}}{\sqrt{1 - n \cdot \frac{1}{n^2}}\sqrt{1 - n \cdot \frac{1}{n^2}}} \\ &= \sqrt{1 - \frac{1}{n}}. \end{aligned} \tag{C.1}$$

Similarly,

$$\begin{aligned} r_2 &= \frac{1 - \frac{2}{n}}{\sqrt{2 - \frac{4}{n}}\sqrt{1 - \frac{1}{n}}} \\ &= \frac{\sqrt{1 - \frac{2}{n}}}{\sqrt{1 - \frac{1}{n}}} \times \frac{1}{\sqrt{2}}. \end{aligned} \tag{C.2}$$

From Eqs. C.1 and C.2, with large enough n, we get

$$r_1 - r_2 \approx 1 - \frac{1}{\sqrt{2}} \approx 0.29$$

Thus, the global sensitivity of the Pearson product moment correlation for binary attributes is at least 0.29. Adding Laplace noise scaled to this will substantially alter the correlation between attributes and hence the corresponding counts.

Appendix D. Computing Two Way Margins in $O(m^2n)$ time

A straightforward way to compute all two-way margins over the binary dataset is as follows: for each pair of binary attributes (B_1, B_2) scan the dataset of n rows and record the number of times each possible value (b_1, b_2) occurs, where $b_1, b_2 \in \{0, 1\}$. However, this takes time proportional to $O(d^2n)$, which can be prohibitive if d is large. We will show a method below that requires time only $O(m^2n + d^2)$. Recall that m is the number of attributes in the original dataset D, and d in its binary expansion D_{B}.

First we compute one-way margins for an attribute A by scanning the database and creating a new hash entry for any new entry $a \in A$ and updating its count in the hash table, all in $O(n)$ time. We then go through each element a in the hash table for A, letting $b = \text{bin}(a)$ be its binary representation, and creating the corresponding binary version of the hash entry $\text{hist}_{\mathsf{B}}(b)$. This can be done in $O(dn)$ time.

Now for each possible pairs of attributes A_1, A_2 in the database D, we initialise another hash table: "hist." If we see a new pair of values (a_1, a_2), we create the entry $\text{hist}(a_1, a_2)$ and set it to 1. Otherwise we increment the counter. Now for each existing value (a_1, a_2), we set $b_1 = \text{bin}(a_1)$ and $b_2 = \text{bin}(a_2)$. Let b_1b_2, $b_1\bar{b}_2$, $\bar{b}_1b_2$ and $\bar{b}_1\bar{b}_2$ denote the number of occurrences of $(1, 1)$, $(1, 0)$, $(0, 1)$ and $(0, 0)$, respectively. Then, these can be computed as

$$\begin{aligned} b_1b_2 &= \text{hist}(a_1, a_2), \\ b_1\bar{b}_2 &= \text{hist}_{\mathsf{B}}(b_1) - \text{hist}(a_1, a_2), \\ \bar{b}_1b_2 &= \text{hist}_{\mathsf{B}}(b_2) - \text{hist}(a_1, a_2), \\ \bar{b}_1\bar{b}_2 &= n - b_1b_2 - b_1\bar{b}_2 - \bar{b}_1b_2. \end{aligned}$$

It is easy to see that the above can be computed in $O(m^2n + d^2)$ time.