Aalborg University

9th semester

Mathematics-Economics

P9-project

Differential Privacy and Synthetic Data Generation using PrivBayes

18/12-2024

AALBORG UNIVERSITY

STUDENT REPORT

AALBORG UNIVERSITET
STUDENTERRAPPORT

Institute of Mathematical Sciences
Mathematics-Economics
Thomas Manns Vej 23
9220 Aalborg East
https://math.aau.dk

Title:

Differential Privacy and Synthetic Data Generation using PrivBayes

Project:

P9-project

Project period:

3rd of September 2024 - 18th of December 2024

Project group:

MO-23

Author:

Jan Reiter Sørensen

Supervisor:

Rasmus Plenge Waagepetersen

Pagecount: 48 + front page
Date of handin: 18/12-2024

Abstract:

Synthetic data is artificially generated data that preserves the same characteristics as the data, it is derived from. This is useful, when it comes to research involving sensitive data, because synthetization of data can impose a degree of privacy, while still preserving the same scientific conclusions of the real dataset. There is however a trade-off between the amount of privacy and the utility of the synthetic data. This project gives an in depth mathematical explanation of differential privacy and studies the privacy-utility trade-off in a simulation study using the smoothed histogram mechanism and `PrivBayes`. These are both differentially private synthetization methods, where `PrivBayes` relies on the use of Bayesian networks, while the smoothed histogram mechanism relies solely on smoothed histograms. The simulation study demonstrates that the utility, represented by the power of hypothesis tests, decreases as the privacy increases. In particular, the utility of the data generated by the `PrivBayes` mechanism, which consists of the composition of two differentially private mechanisms, depends on a mixture parameter that specifies the distribution of the overall privacy budget between the two component mechanisms.

Preface

The project period has elapsed from 03/09-2024 to 18/12-2024.

Reader's guide

Literature and source code references are specified within the text. Literature references are given by author names and year, and refers to the bibliography section before the apendix. All source code used for this project is public and accessible in the following Github-repository:

- *https://github.com/janreiter793/DPAndPrivBayesP9.git*

All source code is free to use publicly without the permission of the author.

Acknowledgements

The work on the project has been supervised by Rasmus Waagepetersen, and throughout the project period he has contributed with lots of useful input and advice. He should therefore have many thanks. In addition, many thanks should also be addressed towards Martin Bøgsted and Heidi Søgaard Christensen, for their help and advice in the preparation of this project.

Contents

1 Introduction

For a researcher to gain access to sensitive data, they will need to obtain approval and follow guidelines for using the requested data. This is for instance, how it works for Danmarks Statistik DST (Danmarks statistik, 2024). Here, a researcher will need to be approved by their institution, but they will also need to pass a test about data safety guidelines. If the researcher fails to follow the guidelines, then sanctions may be imposed both on the researcher and the institution. This system functions as a deterrent to avoid data leaks, but it is not bullet proof, since it only takes one data breach to leak the personal information of many individuals. A way to mitigate this could be to anonymize datasets by simply removing columns that contain identifying information. But this may not always be enough to fully prevent adversaries from identifying individuals in the dataset. An example of a dataset that was released publicly after identifying variables were removed, is the Netflix Prize dataset. This dataset was a subset of their entire database, and it consisted of the full rating history along with dates for the ratings for about 500.000 of their customers with slight perturbations in the data. As is explained by Narayanan and Shmatikov (2008) Netflix had confidence in the anonymity of the dataset. However, in the article they developed a model that was able to correctly identify 99% of the individuals using eight movie ratings of which, two of the ratings may be wrong, along with dates of the ratings, which also just needed to be within 14 days of the real rating day. Using only two ratings and dates with up to three days of error the model could succesfully identify 68% of individuals. This is clearly not a sufficiently private dataset for publication, since as Narayanan and Shmatikov (2008) put it, a boss could from a casual conversation with an employee find the complete movie rating history of that employee. A potential solution to this is to use synthetically generated data instead of the real senstive data. Synthetic data is as explained in Bartell et al. (2024) artificially generated data that tries to preserve the same characteristics, patterns, and scientific conclusions that are present in the dataset, which the synthetic data was derived from. The catch is that to synthesize private data, it is necessary to perturb the real observations in some sense, and this may blur the characteristics and patterns of the original data. It is therefore necessary to find a good balance between the amount of imposed privacy and the utility. So how can privacy be measured? In Reiter et al. (2014) they propose to use *the percentage of correct guesses*. That is, the probability that an adversary with large amounts of auxiliary information is able to correctly guess the attributes of an individual in the original dataset. They also propose to use the difference between the expected guess of an adversay with large amounts of auxiliary information and the real observation. Both proposals are good in the sense that they are intuitively easy to understand, but they can easily become computationally heavy to estimate for high dimensional data. It is also difficult to generally state a suffi-

cient threshold for the difference between the expected guess by an adversary and the real observation, since this metric is highly dependent on the units used for the variables in the dataset. Another solution is to use Differential Privacy, which was first proposed by Dwork (2006). This measure gives an upper bound on how much the probability of synthesizing a specific dataset can differ, when changing a single row in the original dataset. Differential privacy does not need to be estimated, since it is a probabilistic property of the synthesizer that is used. That makes it much easier to use, since the data posesssing agent just needs to determine the wanted amount of privacy and then synthesize the data. There are several different methods for generating data differentially private. Methods span from injecting noise into the original dataset, to estimating the joint distribution of the variables in the dataset, and then perturbing this distribution somehow. For the latter, methods using copula models, machine learning models, and Bayesian network models can be applied. Examples of implementations of these methods can be seen in for instance Zhang et al. (2014) and Tao et al. (2022). Bayesian network modeling has recently become the prefered choice. This is, among other things, because copula models yield the dependence relationship, but not the conditional independence relationship that Bayesian networks can yield. Bayesian network based modeling may therefore be a better choice than copula, especially for high dimensional data, since the property of conditional independence allows for factorization of the joint distribution into a product of low dimensional conditionals. These conditional distributions can be estimated more efficiently than the joint distribution. Graphical model based synthetization, which is what Bayesian network modeling is, also seem to consistently outperform GAN-based methods, which are methods that rely on generative machine learning models, according to Tao et al. (2022). A promising Bayesian network based synthesizer is `PrivBayes`, which was first described by Zhang et al. (2014). In summary, there may be uses for synthetic data generated differentially private, but there is a trade-off between privacy and utility. This trade-off could be interesting to look into, and a good synthesizer to use in the investigation is `PrivBayes`. This boils down to the following problem statement.

Problem statement

Differential privacy is a mathematical property of synthetic data generation methods, and rigorously showing that a synthesizer satisfies differential privacy needs thorough mathematical reasoning. Not all differentially private methods are introduced in a rigorous setting, and neither are they built on the same mathematical terminology. This leads to the following question. How are differentially private methods defined and explained using a consequent and precise mathematical language? There are several methods for synthetization of differentially private data such as the smoothed histogram mechanism and `PrivBayes`*, but they are quite theoretical. How can these mechanisms, if possible, be implemented in practice using a programming language such as* `R`*? Furthermore, there is an inherent trade-off between the privacy and the utility of synthetically generated data. What is the relationship between the privacy and the utility of synthetic data?*

2 Preliminaries

2.1 Bayesian networks

The `PrivBayes` synthesizer relies on Bayesian networks as mentioned in the introduction. Bayesian networks are a special set of directed acyclic graphs, which can be used to model the dependence structure and the conditional independence of a set of random variables. The following definition of directed acyclic graphs is adapted from Koller and Friedman (2009).

Definition 2.1.1. (Directed acyclic graph)
Let $d \in \mathbb{N}$, let $V = \{1, 2, \ldots, d\}$ denote the set of vertices, and let $\Pi_1, \Pi_2, \ldots, \Pi_d \subset V$. The set of tuples $\mathcal{N} = \big\{(1, \Pi_1), (2, \Pi_2), \ldots, (d, \Pi_d)\big\}$, is called a directed acyclic graph if it satisfies the following two conditions:

1. Each Π_i is a subset of $V \setminus \{i\}$, and
2. For any $1 \leq i < j \leq d$, we have $j \notin \Pi_i$.

If $\mathcal{N}$ is a directed acyclic graph, then Π_i is said to be the parent set of i for any $i = 1, 2, \ldots, d$. Furthermore, $j \in V$ is said to be an ancestor of $i \in V$, if $j < i$.

By construction, the parent set of 1 in a directed acyclic graph is always the empty set. A way to visualize a directed acyclic graph is by considering a parent set Π_i as the set of all nodes, which have a directed edge pointing towards i. Figure 2.1 illustrates how the following directed acyclic graph can be visualized

$$\mathcal{N} = \big\{(1, \emptyset), (2, \{1\}), (3, \{2\})\big\}. \tag{2.1}$$

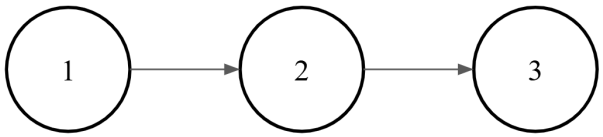


Figure 2.1: A visualization of the directed acyclic graph defined by (2.1).

A Bayesian network is a directed acyclic graph, where all nodes represent a unique random variable, and it shows the structure of conditional independence of the random variables. The following definition of Bayesian networks is adapted from Nielsen and Jensen (2007).

Definition 2.1.2. (Bayesian network)
Let $(X_1, X_2, \ldots, X_d)$ for a $d \in \mathbb{N}$ be a stochastic vector, and let

$$\mathcal{N} = \left\{(1, \Pi_1), (2, \Pi_2), \ldots, (d, \Pi_d)\right\},$$

be a directed acyclic graph. The directed acyclic graph $\mathcal{N}$ is said to be a Bayesian network, if for all $i = 1, 2, \ldots, d$, we have that

$$\mathbb{P}\left(X_i | X_1, X_2, \ldots, X_{i-1}\right) = \mathbb{P}\left(X_i | X_j, j \in \Pi_i\right). \tag{2.2}$$

If $\mathcal{N}$ is a Bayesian network, then its degree is given by

$$k := \max\left\{\#\Pi_i | i = 1, 2, \ldots, d\right\}.$$

Also, if $X_j \in \Pi_i$ then X_j is said to be a parent of X_i, and X_j is called an ancestor of X_i, if j is an ancestor of i.

Equation (2.2) states the conditional independence property of Bayesian networks. Consider again the example given in (2.1). If this directed acyclic graph is a Bayesian network, then for instance X_3 is independent from X_1, when conditioned on X_2. Consider the random vector $V = (X_1, X_2, \ldots, X_d)$ with the Bayesian network

$$\mathcal{N} = \left\{1, \Pi_1), (2, \Pi_2), \ldots, (d, \Pi_d)\right\},$$

representing the conditional independence between the X_i's. Using the definition of conditional probability inductively the joint distribution of V can be rewritten in the following way

$$\begin{aligned}\mathbb{P}\left(X_1, X_2, \ldots, X_d\right) &= \mathbb{P}\left(X_d | X_{d-1}, X_{d-2}, \ldots, X_1\right) \mathbb{P}\left(X_{d-1} | X_{d-2}, \ldots, X_1\right) \ldots \mathbb{P}\left(X_1\right) \\ &\overset{(2.2)}{=} \prod_{i=1}^{d} \mathbb{P}\left(X_i | \Pi_i\right). \end{aligned} \tag{2.3}$$

This is why, Bayesian networks can be used to find joint distributions of sets of attributes.

3 Differential Privacy

A database is according to Gillis et al. (2024) a collection of digital information stored into records or files on a server. The information stored could for instance be sales transactions, customer data, product information, etc. There are different ways to store data in a database, but this project will only take basis in databases, where the data is stored in rows and columns. Formally, a database will be defined in the following way.

Definition 3.0.1. (Database)
A database is a random variable $X : \Omega \to \mathbb{R}^{n\times d}$ with $n, d \in \mathbb{N}$. Each row is assumed to be independent and identically distributed. A realisation of a database is called a dataset.

Definition 3.0.1 states that a database is a random matrix. We will consider each row as an observation of d attributes, hence each column represents its own attribute. The density of a database is the joint density of its rows, which, as a consequence of the independence assumption, can be expressed as the product of the row densities. In Wasserman and Zhou (2018) a synthetically generated database of X, say $Z : \Omega \to \mathbb{R}^{k\times d}$ for some choice of $k \in \mathbb{N}$ not necessarily equal to n, is generated by sampling from a conditional distribution of X. These conditional distribution functions are called data release mechanisms, and they are defined in the following way.

Definition 3.0.2. (Data release mechanism)
Let $X : \Omega \to \mathbb{R}^{n\times d}$ with $n, d \in \mathbb{N}$ and $Z : \Omega \to \mathbb{R}^{k\times d}$ with $k \in \mathbb{N}$ be databases. The conditional probability measure of Z denoted $Q(\cdot|X) : \mathcal{B}(\mathbb{R}^{k\times d}) \to [0, 1]$ defined by

$$Q(B|X) \coloneqq \mathbb{P}(Z \in B|X),$$

for all $B \in \mathcal{B}(\mathbb{R}^{k\times d})$ is called a data release mechanism, where $\mathcal{B}(\mathbb{R}^{k\times d})$ is the Borel σ-algebra generated by $\mathbb{R}^{k\times d}$.

The goal of differential privacy is to classify whether or not a data release mechanism satisfies a specified degree of privacy. The term privacy refers to the risk of disclosing individuals or attributes in a dataset given a synthetically generated one. In other words, given a dataset that is synthetically generated from a private mechanism, how difficult is it for an adversary to identify, whether or not an individual appeared in the original dataset? The definition of differential privacy takes basis in adjacent datasets. Adjacent datasets are datasets, which differ in exactly one row, or more formally, they are datasets, which

have a Hamming distance of one. The Hamming distance is defined by Wasserman and Zhou (2018) in the following way.

Definition 3.0.3. (The Hamming distance)
Let $n, d \in \mathbb{N}$, and let $\delta : \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times d} \to \{0, 1, \ldots, n\}$ be given by

$$\delta(x, x') \coloneqq \#\left\{i \mid x_i \neq x'_i\right\},$$

for any $x, x' \in \mathbb{R}^{n\times d}$, where x_i, x'_i denotes the i^{th} row of x and x' respectively. Then the map δ is called the Hamming distance.

Two outcomes $\omega, \omega' \in \Omega$ are called adjacent if $\delta\left(X(\omega), X(\omega')\right) = 1$. The set of adjacent datasets is given by

$$N(X) \coloneqq \left\{\left(X(\omega), X(\omega')\right) \mid \omega, \omega' \in \Omega, \delta\left(X(\omega), X(\omega')\right) = 1\right\}.$$

That is for any tuple $(x, x') \in N(X)$, we have that x and x' are two adjacent datasets which X may assume. The following definition of differential privacy is from Wasserman and Zhou (2018).

Definition 3.0.4. (α-differential privacy)
Let $X : \Omega \to \mathbb{R}^{n\times d}$ with $n, d \in \mathbb{N}$ be a database, let $Q(\cdot|X)$ be a data release mechanism, and let $\alpha \in \mathbb{R}_{\geq 0}$. The data release mechanism $Q(\cdot|X)$ is said to satisfy α-differential privacy if

$$\sup_{\substack{B\in\mathcal{B}(\mathbb{R}^{k\times d})\\(x,x')\in N(X)}} \frac{Q(B|X = x)}{Q(B|X = x')} \leq \exp(\alpha),$$

for some choice of $k \in \mathbb{N}$.

The definition of α-differential privacy states that for any event $B \in \mathcal{B}(\mathbb{R}^{k\times d})$, and two adjacent datasets $(x, x') \in N(X)$, we have that

$$\frac{Q(B|X = x)}{Q(B|X = x')} \in \left[\exp(-\alpha), \exp(\alpha)\right].$$

In other words, for α close to zero the likelihood of an event/sample of the synthetically generated database is unaffected by changes to the values of any single row in the original dataset. This is useful in the context of publishing sensitive datasets, since for α close to zero, it becomes difficult to determine whether or not an individual was included in the original dataset.

An example of a data release mechanism

Consider the experiment of flipping a not necessarily fair coin n times, and denote tail as 1 and head as 0. We may represent the output of this experiment as a vector $X \in \{0,1\}^n$. Denote the number of tails as

$$T = \sum_{i=1}^{n} X_i,$$

and let $p = T/n$ define the proportion of tails. Defining the following conditional distribution function

$$Q(z|X) = p^r(1-p)^{k-r},$$

for $z \in \{0,1\}^k$, where $r = \sum z_i$, we obtain a data release mechanism, from which we can sample a synthetically generated dataset. Let $(x, x') \in N(X)$ be given, and define

$$T' = \sum_{i=1}^{n} x'_i,$$

and $p' = T'/n$. Using the fact that $T' = T \pm 1$, we obtain $p' = p \pm \frac{1}{n}$, which yields for any $z \in \{0,1\}^k$ that

$$\begin{aligned}
\frac{Q(z|X=x)}{Q(z|X=x')} &= \frac{p^r(1-p)^{k-r}}{p'^r(1-p')^{k-r}} \\
&= \frac{p^r(1-p)^{k-r}}{\left(p \pm \frac{1}{n}\right)^r \left(1 - \left(p \pm \frac{1}{n}\right)\right)^{k-r}} \\
&\leq \frac{p^r(1-p)^{k-r}}{\left(p - \frac{1}{n}\right)^r \left(1 - \left(p + \frac{1}{n}\right)\right)^{k-r}}.
\end{aligned}$$

Applying the logarithm yields

$$\ln\left(\frac{p^r(1-p)^{k-r}}{\left(p - \frac{1}{n}\right)^r \left(1 - \left(p + \frac{1}{n}\right)\right)^{k-r}}\right) = r\ln\left(\frac{p}{1-p}\right) - r\ln\left(\frac{p - \frac{1}{n}}{1 - p - \frac{1}{n}}\right) + k\ln\left(\frac{1-p}{1-p-\frac{1}{n}}\right).$$

This shows that Q satisfies α-differential privacy, if for any $z \in \{0,1\}^k$ and $(x, x') \in N(X)$, α satisfies that

$$r\ln\left(\frac{p}{1-p}\right) - r\ln\left(\frac{p - \frac{1}{n}}{1 - p - \frac{1}{n}}\right) + k\ln\left(\frac{1-p}{1-p-\frac{1}{n}}\right) \leq \alpha.$$

3.1 Generating synthetic data using a smoothed histogram

A simple method for generating differentially private datasets is by using smoothed histograms. This method is described in Wasserman and Zhou (2018) for databases, where the sample space of the rows is $[0,1]^d$ with $d \in \mathbb{N}$, but in this project the sample space for dataset rows is generalized from $[0,1]^d$ to any compact subset $S \subset \mathbb{R}^d$ to have a method that also allows for categorical attributes. Because of the compactness there exists a finite cover $(B_i)_{i\in\mathcal{I}}$ of S consisting of disjoint closed subsets of $\mathbb{R}^d$ for some finite set $\mathcal{I}$. It is assumed that

$$\bigcup_{i\in\mathcal{I}} B_i = S, \tag{3.1}$$

and that there exists a measure μ, such that $(S, \mathcal{B}(S), \mu)$ is a measure space. Consider a database X that attains values in S^n for some $n \in \mathbb{N}$. Let C_j denote the number of rows falling into B_j, that is

$$C_j \coloneqq \sum_{i=1}^{n} \mathbb{1}_{\{X_i \in B_j\}},$$

for all $j \in \mathcal{I}$, and where $\mathbb{1}$ denotes the indicator function. An unbiased estimator for the marginal probability that a row lies in B_j is

$$\hat{p}_j \coloneqq \frac{C_j}{n},$$

for all $j \in \mathcal{I}$. Using this, the histogram estimator for X and the specified cover is defined by

$$\hat{f}(z) \coloneqq \sum_{j\in\mathcal{I}} \frac{\hat{p}_j}{\mu(B_j)} \mathbb{1}_{\{z\in B_j\}},$$

for $z \in S$. This estimator is an empirical density for the rows of X with respect to μ, since

$$\begin{aligned}
\int_S \hat{f}(z)d\mu(z) &= \int_S \sum_{j\in\mathcal{I}} \frac{\hat{p}_j}{\mu(B_j)} \mathbb{1}_{\{z\in B_j\}} d\mu(z) \\
&= \sum_{j\in\mathcal{I}} \frac{\hat{p}_j}{\mu(B_j)} \int_S \mathbb{1}_{\{z\in B_j\}} d\mu(z) \\
&= \sum_{j\in\mathcal{I}} \frac{\hat{p}_j}{\mu(B_j)} \mu(B_j) \\
&= 1.
\end{aligned}$$

The histogram estimator can be used to construct the smoothed histogram estimator. Let the smoothing parameter $\delta \in]0,1[$ be given, then the smoothed histogram estimator $\hat{f}_\delta$ is given by

$$\hat{f}_\delta(z) \coloneqq (1-\delta)\hat{f}(z) + \delta,$$

for $z \in S$. Likewise, for this estimator we have that

$$\int_S \hat{f}_\delta(z)d\mu(z) = (1-\delta)\int_S \hat{f}(z)d\mu(z) + \delta = 1.$$

From the construction of $\hat{f}_\delta$, when δ is close to one, the smoothed histogram will be smoothed so much that it almost constantly equals one in its support. Conversely, when δ is close to zero, the smoothed histogram will resemble the original histogram estimator closely. Synthetic datasets can be generated by independently sampling rows from $\hat{f}_\delta$. According to the following theorem, this mechanism satisfies a degree of differential privacy.

Theorem 3.1.1. (Differential privacy for smoothed histogram samples)
Let $\delta \in]0,1[$ be given, and let $\hat{f}_\delta$ be a smoothed histogram based on a dataset of $n \in \mathbb{N}$ rows. Sampling $Z_1, Z_2, \ldots, Z_k$ for $k \in \mathbb{N}$ from $\hat{f}_\delta$, yields a dataset $Z = (Z_1, Z_2, \ldots, Z_k)$, that satisfies α-differential privacy for

$$\alpha = k \ln\left(\frac{(1-\delta)m}{n\delta} + 1\right), \tag{3.2}$$

where

$$m = \sup_{i \in \mathcal{I}} \frac{1}{\mu(B_i)}.$$

Proof. Let X be a database attaining values in S^n for some $n \in \mathbb{N}$, let $(x, x') \in N(X)$, let $\hat{f}_\delta$ and $\hat{g}_\delta$ for some given $\delta \in]0,1[$ be the smoothed histogram estimators based on x and x' respectively, and let $\tilde{f}$ and $\tilde{g}$ be the joint densities of x and x' respectively, such that

$$\tilde{f}(z) = \prod_{i=1}^{k} \hat{f}_\delta(z_i), \text{ and } \tilde{g}(z) = \prod_{i=1}^{k} \hat{g}_\delta(z_i), \tag{3.3}$$

for $z \in S^k$, where $k \in \mathbb{N}$, and z_i denotes the i^{th} row of z. Let $(B_i)_{i \in \mathcal{I}}$ denote the closed finite disjoint cover of S satisfying (3.1). It is sufficient to show that

$$\sup_{z \in S^k} \frac{\tilde{f}(z)}{\tilde{g}(z)} \leq \left(\frac{(1-\delta)m}{n\delta} + 1\right)^k.$$

In the same manner as before the number of rows in B_j and the estimator of the probability that a row lies in B_j for x and x', are denoted by

$$C_j = \sum_{i=1}^{n} \mathbb{1}_{\{x_i \in B_j\}}, \text{ and } C'_j = \sum_{i=1}^{n} \mathbb{1}_{\{x'_i \in B_j\}},$$

and

$$\hat{p}_j = \frac{C_j}{n}, \text{ and } \hat{q}_j = \frac{C'_j}{n},$$

respectively. It must hold that $0 \leq |\hat{p}_i - \hat{q}_i| \leq \frac{1}{n}$ for all $i \in \mathcal{I}$. Overall there are two cases depending on the choice of $(x, x') \in N(X)$. Either $|\hat{p}_i - \hat{q}_i| = 0$ for all $i \in \mathcal{I}$, or $|\hat{p}_i - \hat{q}_i| = 0$ for all $i \in \mathcal{I} \setminus \{j\}$ for some $j \in \mathcal{I}$, where $|\hat{p}_j - \hat{q}_j| = \frac{1}{n}$. In the former case the ratio

$$\frac{\tilde{f}(z)}{\tilde{g}(z)} = 1,$$

which satisfies the theorem for any choice of k, δ, m, and n, hence, we may assume the latter case. The condition $|\hat{p}_j - \hat{q}_j| = \frac{1}{n}$, can be summarised in two cases. Either $\hat{p}_j = \frac{1}{n}$ and $\hat{q}_j = 0$, or $\hat{q}_j \geq \frac{1}{n}$ and $\hat{p}_j = \hat{q}_j \pm \frac{1}{n}$. Note that $\hat{p}_j$ and $\hat{q}_j$ may be interchanged, but it is not necessary to show for this case, because of symmetry in the arguments. Since $\tilde{f}$ and $\tilde{g}$ only differ in B_j, we let $z \in S^k$ be chosen, such that $z_1, z_2, \ldots, z_k \in B_j$. Consider now the first case, that is, $\hat{p}_j = \frac{1}{n}$ and $\hat{q}_j = 0$. We have that

$$\max\left(\frac{\tilde{f}(z)}{\tilde{g}(z)}, \frac{\tilde{g}(z)}{\tilde{f}(z)}\right) = \frac{\tilde{f}(z)}{\tilde{g}(z)},$$

since $\hat{q}_j = 0$ implies $\tilde{g}(z) \leq \tilde{f}(z)$ for $z_1, \ldots, z_k \in B_j$. By using (3.3), we obtain that

$$\begin{aligned}
\frac{\tilde{f}(z)}{\tilde{g}(z)} &= \prod_{i=1}^{k} \frac{\hat{f}_\delta(z_i)}{\hat{g}_\delta(z_i)} \\
&= \left(\frac{(1-\delta)\frac{\hat{p}_j}{\mu(B_j)} + \delta}{(1-\delta)\frac{\hat{q}_j}{\mu(B_j)} + \delta}\right)^k \\
&= \left(\frac{(1-\delta)\frac{1}{n} + \mu(B_j)\delta}{\mu(B_j)\delta}\right)^k \\
&= \left(\frac{(1-\delta)}{n\mu(B_j)\delta} + 1\right)^k \\
&\leq \left(\frac{(1-\delta)m}{n\delta} + 1\right)^k.
\end{aligned}$$

Consider now the second case, which is where $\hat{q}_j \geq \frac{1}{n}$, and $\hat{p}_j = \hat{q}_j \pm \frac{1}{n}$. We have that

$$\begin{aligned}
\max\left(\frac{\tilde{f}(z)}{\tilde{g}(z)}, \frac{\tilde{g}(z)}{\tilde{f}(z)}\right) &= \max\left(\left(\frac{(1-\delta)\frac{\hat{q}_j \pm \frac{1}{n}}{\mu(B_j)} + \delta}{(1-\delta)\frac{\hat{q}_j}{\mu(B_j)} + \delta}\right)^k, \left(\frac{(1-\delta)\frac{\hat{q}_j}{\mu(B_j)} + \delta}{(1-\delta)\frac{\hat{q}_j \pm \frac{1}{n}}{\mu(B_j)} + \delta}\right)^k\right) \\
&\leq \max\left(\left(\frac{(1-\delta)(\hat{q}_j + \frac{1}{n}) + \mu(B_j)\delta}{(1-\delta)\hat{q}_j + \mu(B_j)\delta}\right)^k, \left(\frac{(1-\delta)\hat{q}_j + \mu(B_j)\delta}{(1-\delta)(\hat{q}_j - \frac{1}{n}) + \mu(B_j)\delta}\right)^k\right) \\
&= \max\left(\left(\frac{(1-\delta)\frac{1}{n}}{(1-\delta)\hat{q}_j + \mu(B_j)\delta} + 1\right)^k, \left(\frac{(1-\delta)\hat{q}_j + \mu(B_j)\delta}{(1-\delta)(\hat{q}_j - \frac{1}{n}) + \mu(B_j)\delta}\right)^k\right) \\
&\leq \left(\frac{(1-\delta)\frac{1}{n}}{(1-\delta)(\hat{q}_j - \frac{1}{n}) + \mu(B_j)\delta} + 1\right)^k \\
&\leq \left(\frac{(1-\delta)m}{n\delta} + 1\right)^k.
\end{aligned}$$

This shows that

$$\sup_{z \in S^k} \frac{\tilde{f}(z)}{\tilde{g}(z)} \leq \left(\frac{(1-\delta)m}{n\delta} + 1\right)^k,$$

which concludes the proof. ■

Theorem 3.1.1 states that the smoothed histogram mechanism satisfies α-differential privacy for α's satisfying (3.2), however, it does not omit the possibility of an $0 \leq \varepsilon < \alpha$, such that the mechanism satisfies ε-differential privacy as well.

3.2 Simulation study of the smoothed histogram mechanism

This section investigates, how the differential privacy bound in Theorem 3.1.1 affects the power of a Spearman rank test for dependence between two variables by performing Monte Carlo simulation for different levels of privacy calculated using (3.2). This is done iteratively for each level of differential privacy by first generating a dataset consisting of two correlated variables, estimating its smoothed histogram, sampling a new dataset, and then performing the Spearman rank test on the two variables in the synthetically generated dataset. This process of generating data, synthesizing, and then testing is repeated multiple times for each level of differential privacy, and for each iteration a one is returned if the null hypothesis was rejected, and zero if it was not rejected. The estimated power is

then calculated by

$$\tilde{p}(\alpha) \coloneqq \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_i(\alpha),$$

where $N \in \mathbb{N}$ is the number of iterations per level of differential privacy, and $\mathbb{1}_i(\alpha)$ is the indicator function yielding one if iteration i rejected the null hypothesis and zero otherwise. The study takes basis in a database X that assmues values in $[0,1]^{n\times 2}$, thus the database consists of two columns, say X_1 and X_2, with $n \in \mathbb{N}$ rows. The rows of X_1 are sampled from unif$[0,1]$, and X_2 is given by

$$X_2 = X_1 + \nu$$

where $\nu \sim N_n(0, \text{diag}_n(1/10))$. Since we want the entries of X_2 in $[0,1]$, the entries that are outside the interval are repeatedly resampled until all entries are within $[0,1]$. Since X_1 and X_2 are not both normally distributed the correlation, ρ, is estimated empirically using the Spearman rank correlation coefficient, which is given by

$$\rho \coloneqq 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}.$$

Here d_i is the difference between the rank of the i^{th} entry of X_1 and X_2. For this example's data the correlation coefficient is estimated to be $\rho \approx 0.94$ using `R` with 10000 samples of X_1 and X_2. After generating X, the smoothed histogram is estimated, and a new synthetic dataset attaining values in $[0,1]^{k\times 2}$ for some $k \in \mathbb{N}$ is generated by sampling from the smoothed histogram. Finally, a Spearman rank test for correlation is conducted between X_1 and X_2. That is, we are testing the hypothesis $\mathcal{H}_0 : \rho = 0$ against $\mathcal{H}_1 : \rho \neq 0$. Figure 3.1 demonstrates an example of the generated data, and the synthesized data.

The simulation is run with $n = 10000$, $k = 1000$, and $m = 1/10^2$. The value of m is obtained by choosing $(B_i)_{i\in\mathcal{I}}$ to consist of the closed squares all with side length $1/10$, and μ is the Lebesgue measure. To choose the δ corresponding to a given α, the equation

$$k \log\left(\frac{(1-\delta)m}{n\delta} + 1\right) = \alpha,$$

is inverted in order to isolate δ, which yields

$$\delta = m\left(n\exp\left(\frac{\alpha}{k}\right) + m - n\right)^{-1}.$$

The number, N, of iterations per level of differential privacy is chosen, such that the distance between the upper and lower standard deviations is 0.01, that is $2\sigma = 0.01$, where σ is the standard deviation of $\tilde{p}(\alpha)$. To calculate N consider first the variance of $\tilde{p}(\alpha)$

$$\text{Var}\left(\tilde{p}(\alpha)\right) = \text{Var}\left(\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}_i(\alpha)\right) = \frac{1}{N^2}\sum_{i=1}^{N}\text{Var}\left(\mathbb{1}_i(\alpha)\right) = \frac{p(1-p)}{N},$$

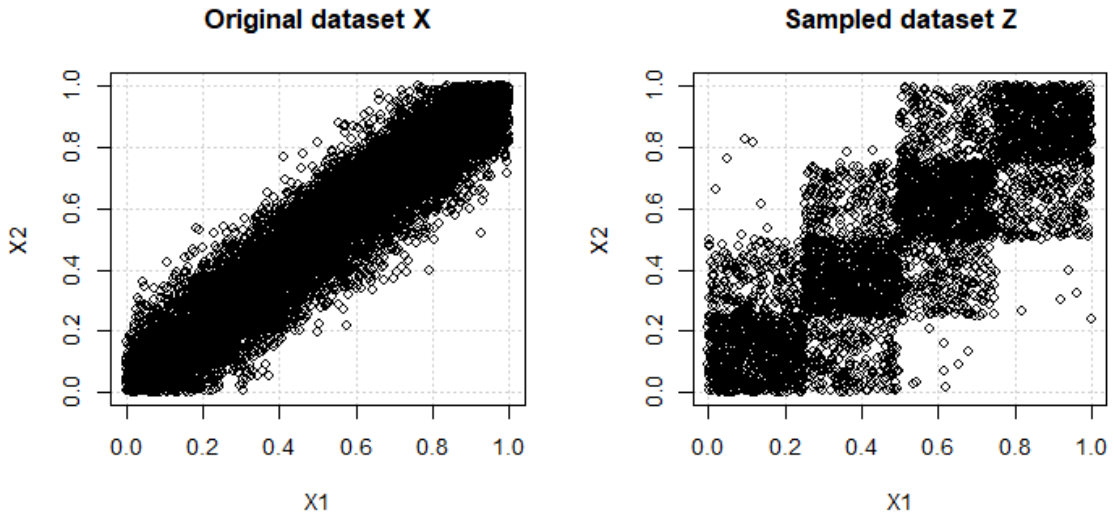


Figure 3.1: Example of a generated dataset and a synthetically generated dataset sampled from the smoothed histogram of the original dataset.

where $p := p(\alpha)$ is the actual power of the Spearman rank test for the synthetically generated dataset. To obtain N, we now solve the following equation for N

$$2\sqrt{\frac{p(1-p)}{N}} = \frac{1}{100} \Leftrightarrow N = (200)^2 p(1-p).$$

The highest value of N is obtained, when $p = \frac{1}{2}$, therefore $N = 10000$ will be used. The simulation is run using `R`, and the source code is available in the following git-repository:

- *https://github.com/janreiter793/DPAndPrivBayesP9.git*

in the file `smoothed_histogram_sampling.R`. The results are displayed in Figure 3.2.

3.3 The Laplace and the exponential mechanism

In this section, two other differentially private mechanisms will be discussed. These mechanisms are the Laplace mechanism and the exponential mechanism, and they are used by the `PrivBayes` synthesizer, which will be discussed in Chapter 4. The Laplace mechanism is a mechanism that can be used to turn the outputs of a function differentially private. The following definition, which is adapted from Wasserman and Zhou (2018), defines differential privacy for functions.

Definition 3.3.1. (α-differential privacy for functions)
Let $X : \Omega \to \mathbb{R}^{n \times d}$ with $n, d \in \mathbb{N}$ be a database, let $R : \Omega \to \mathbb{R}^p$ with $p \in \mathbb{N}$ be a

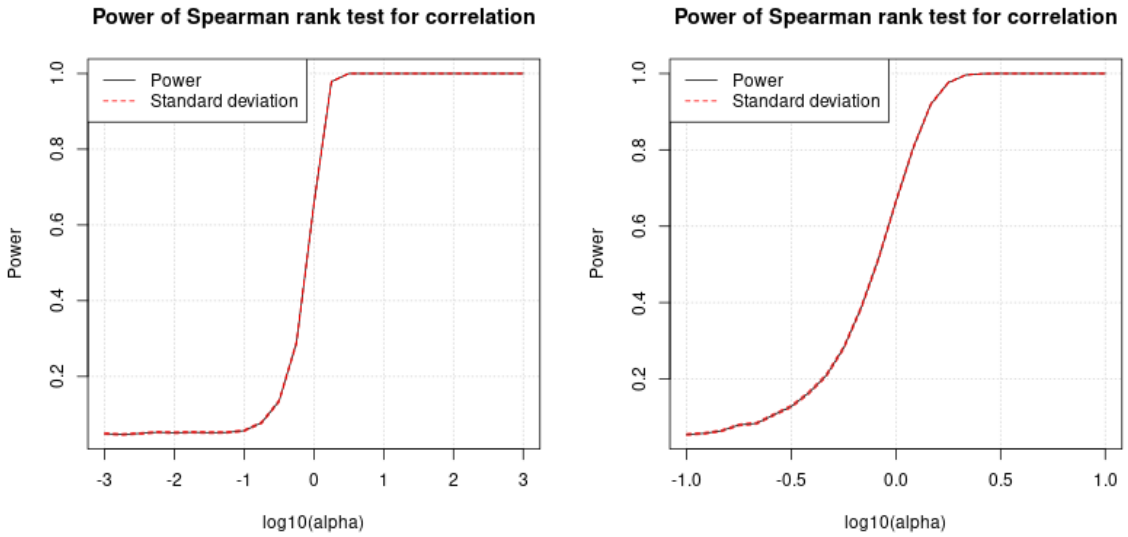


Figure 3.2: The estimated power of the Spearman rank test using Monte Carlo simulation for different levels of α. Right plot is a subplot of the left plot.

random vector, and let $T : \mathbb{R}^{n\times d} \times \mathbb{R}^p \to M$ for some non-empty set M. Then $T(\cdot, R)$ is said to satisfy α-differential privacy for $\alpha \in \mathbb{R}_{\geq 0}$, if for any $B \subset M$, we have that

$$\sup_{(x,x')\in N(X)} \frac{\mathbb{P}(T(X,R) \in B | X = x)}{\mathbb{P}(T(X,R) \in B | X = x')} \leq \exp(\alpha).$$

If $T(\cdot, R)$ satisfies differential privacy, it is said to be a differentially private function.

In the case where $M = \mathbb{R}^{k\times d}$, defining $Z = T(X, R)$, we see that Definition 3.0.4 and Definition 3.3.1 are equivalent.

A fundamental property of differentially private functions is that parallel compositions yield new differentially private functions. This result is stated in Dwork et al. (2006), and the following proposition and proof are adapted from that paper.

Proposition 3.3.2. (Simple composition)
Let $R_1 : \Omega \to \mathbb{R}^{p_1}$ and $R_2 : \Omega \to \mathbb{R}^{p_2}$ with $p_1, p_2 \in \mathbb{N}$ be independent random variables, and let $T_1 : \mathbb{R}^{n\times d} \times \mathbb{R}^{p_1} \to M_1$ and $T_2 : \mathbb{R}^{n\times d} \times \mathbb{R}^{p_2} \to M_2$ be α_1- and α_2-differentially private functions, respectively, for some $\alpha_1, \alpha_2 \in \mathbb{R}_{\geq 0}$, and for non-empty sets M_1 and M_2. The parallel composition of T_1 and T_2, denoted $\mathcal{C}_{T_1,T_2} : \mathbb{R}^{n\times d} \times \mathbb{R}^{p_1} \times \mathbb{R}^{p_2} \to M_1 \times M_2$ given by

$$\mathcal{C}_{T_1,T_2}(\cdot; R_1, R_2) \coloneqq \big(T_1(\cdot, R_1), T_2(\cdot, R_2)\big),$$

satisfies $(\alpha_1 + \alpha_2)$-differential privacy.

Proof. Let $X : \Omega \to \mathbb{R}^{n\times d}$ be a database, such that $T_1(\cdot, R_1)$ and $T_2(\cdot, R_2)$ are α_1- and α_2-differentially private respectively. In other words, for any measurable subsets, $B_1 \subset M_1$, and $B_2 \subset M_2$, and for any $(x, x') \in N(X)$, we have that

$$\frac{\mathbb{P}(T_1(X, R_1) \in B_1 | X = x)}{\mathbb{P}(T_1(X, R_1) \in B_1 | X = x')} \leq \exp(\alpha_1), \text{ and } \frac{\mathbb{P}(T_2(X, R_2) \in B_2 | X = x)}{\mathbb{P}(T_2(X, R_2) \in B_2 | X = x')} \leq \exp(\alpha_2).$$

It suffices to show that for $B = (B_1, B_2)$, we have

$$\frac{\mathbb{P}(\mathcal{C}_{T_1,T_2}(X; R_1, R_2) \in B | X = x)}{\mathbb{P}(\mathcal{C}_{T_1,T_2}(X; R_1, R_2) \in B | X = x')} \leq \exp(\alpha_1 + \alpha_2).$$

This can be shown with the following reasoning

$$\begin{aligned}
\frac{\mathbb{P}(\mathcal{C}_{T_1,T_2}(X; R_1, R_2) \in B | X = x)}{\mathbb{P}(\mathcal{C}_{T_1,T_2}(X; R_1, R_2) \in B | X = x')} &= \frac{\mathbb{P}(\big(T_1(X, R_1), T_2(X, R_2)\big) \in (B_1, B_2)) | X = x)}{\mathbb{P}(\big(T_1(X, R_1), T_2(X, R_2)\big) \in (B_1, B_2) | X = x')} \\
&= \frac{\mathbb{P}(T_1(X, R_1) \in B_1, T_2(X, R_2) \in B_2 | X = x)}{\mathbb{P}(T_1(X, R_1) \in B_1, T_2(X, R_2) \in B_2) | X = x')} \\
&\overset{(\dagger)}{=} \frac{\mathbb{P}(T_1(X, R_1) \in B_1 | X = x)\mathbb{P}(T_2(X, R_2) \in B_2 | X = x)}{\mathbb{P}(T_1(X, R_1) \in B_1 | X = x')\mathbb{P}(T_2(X, R_2) \in B_2) | X = x')} \\
&\leq \exp(\alpha_1)\exp(\alpha_2) \\
&= \exp(\alpha_1 + \alpha_2).
\end{aligned}$$

At (†) the assumption of independence between R_1 and R_2 is used. ■

Using Proposition 3.3.2 inductively it can be shown that for any finite set of functions $T_1, T_2, \ldots, T_n$, which are α_1-, α_2-, …, α_n-differentially private respectively, their parallel composition satisfies $\left(\sum_{i=1}^{n} \alpha_i\right)$-differential privacy.

3.3.1 The Laplace mechanism

The Laplace mechanism works by adding noise drawn from a Laplace distribution to a mapping of database realizations. The Laplace distribution is defined by Dwork et al. (2006) in the following way.

Definition 3.3.3. (The Laplace distribution)
A random variable R attaining values in $\mathbb{R}$ is said to follow the Laplace distribution with parameter $\lambda \in \mathbb{R}_{>0}$, if it has the density function

$$p(r) = \frac{1}{2\lambda} \exp\left(-\frac{|r|}{\lambda}\right),$$

for $r \in \mathbb{R}$, and this is denoted by $R \sim \text{Lap}(\lambda)$. A p-dimensional Laplace distributed random vector with $p \in \mathbb{N}$ denoted $\text{Lap}(\lambda)^p$, is a vector attaining values in $\mathbb{R}^p$ with probability density given by the product of p $\text{Lap}(\lambda)$ densities.

In the literature the Laplace distribution is sometimes defined with two parameters instead of one. These are a location and a scale parameter. See for instance Norton et al. (2019). For the purpose of this project we will only use the parameter λ as expressed in Definition 3.3.3, which corresponds to the scale parameter and the location parameter will be set to 0. For a p-dimensional Laplace distributed vector, the density becomes

$$p(r) = \prod_{i=1}^{p} \frac{1}{2\lambda} \exp\left(-\frac{|r_i|}{\lambda}\right) = \frac{1}{(2\lambda)^p} \exp\left(-\frac{\sum_{i=1}^{p} |r_i|}{\lambda}\right) = \frac{1}{(2\lambda)^p} \exp\left(-\frac{\|r\|_{L_1}}{\lambda}\right),$$

for $r \in \mathbb{R}^p$, where r_i is the i^{th} entry in r, and $\|\cdot\|_{L_1}$ is the L_1-norm. The following definition describes the Laplace mechanism as given by Dwork et al. (2006).

Theorem 3.3.4. (The Laplace mechanism)
Let $X : \Omega \to \mathbb{R}^{n \times d}$ with $n, d \in \mathbb{N}$ be a database, let $f : \mathbb{R}^{n \times d} \to \mathbb{R}^p$ be a map with $p \in \mathbb{N}$, and let $T : \mathbb{R}^{n \times d} \times \mathbb{R}^p \to \mathbb{R}^p$ be defined by

$$T(x, r) \coloneqq f(x) + r, \tag{3.4}$$

where $x \in \mathbb{R}^{n \times d}$ and $r \in \mathbb{R}^p$. Then $T(\cdot, R)$ is said to be a Laplace mechanism, and it satisfies α-differential privacy for $\alpha \in \mathbb{R}_{>0}$, if $R \sim \text{Lap}\left(S(f)/\alpha\right)^p$, where $S(f)$ denotes the L_1-sensitivity of f, which is defined by

$$S(f) \coloneqq \sup_{(x,x') \in N(X)} \left\| f(x) - f(x') \right\|_{L_1}.$$

Proof. Let $z \in \mathbb{R}^p$ be given, and let $p_{T,x}$ denote the density of $T(x, R)$. Since $T(x, r) = f(x) + r$ we have that

$$p_{T,x}(z) = p\left(z - f(x)\right),$$

where p is the Laplace density for R. Let $(x, x') \in N(X)$ be given. Then

$$\begin{aligned} \frac{p_{T,x}(z)}{p_{T,x'}(z)} &= \exp\left(\frac{\alpha\left(\left\|f(x')\right\|_{L_1} - \left\|f(x)\right\|_{L_1}\right)}{S(f)}\right) \\ &\leq \exp\left(\frac{\alpha\left\|f(x') - f(x)\right\|_{L_1}}{S(f)}\right) \end{aligned}$$

$$\leq \exp(\alpha).$$

Let $B \subset M$ be a measurable subset, then

$$\begin{aligned}\mathbb{P}\left(T(x,R) \in B\right) &= \int_B p_{T,x}(z)dz \\ &\leq \int_B \exp(\alpha) p_{T,x'}(z)dz \\ &= \exp(\alpha)\mathbb{P}\left(T(x',R) \in B\right),\end{aligned}$$

which concludes α-differential privacy. ■

3.3.2 The exponential mechanism

The exponential mechanism is useful when we want to pick elements of a set M on the basis of a dataset, in a differentially private manner. In this project, we will use the exponential mechanism to choose the parent sets for the nodes in Bayesian networks, when using `PrivBayes`. Here, the set M will represent the set of possible parent sets for each node. A scoring function will score each possible parent set in M with the observed dataset, and a random function on M, will be defined such that it yields exponentially higher values for parent sets with a good score than parent sets with a low score. Sampling the parent sets from this density will result in Bayesian network structures that are generated differentially private. The following definition of the exponential mechanism is adapted from McSherry and Talwar (2007).

Definition 3.3.5. (The exponential mechanism)
Let $(M, \mathcal{A}, \mu)$ be a measure space, let $n, d \in \mathbb{N}$, let $q : \mathbb{R}^{n\times d} \times M \to \mathbb{R}$ be any map, and let

$$\int_M \exp\left(\alpha q(x,r)\right) d\mu(r) < \infty,$$

for $\alpha \in \mathbb{R}_{\geq 0}$, and define the density g_x with respect to μ, such that

$$g_x(r) \propto \exp\left(\alpha q(x,r)\right), \tag{3.5}$$

for $x \in \mathbb{R}^{n\times d}$ and $r \in M$. The exponential mechanism is given by

$$Q(B|X = x) \coloneqq \int_B g_x(r)d\mu(r),$$

for any $B \in \mathcal{A}$, and where X is a database assuming values in $\mathbb{R}^{n\times d}$.

In practice, M is the set that we want to sample from, μ can usually be chosen as either the Lebesgue measure, the counting measure, or a mixture of both, the map q is to be

understood as the score between an input dataset x and an output from M, and g_x is the density from which elements in M can be sampled differentially private. As can be seen in (3.5), outputs with a higher score is exponentially more likely to be sampled relative to outputs with a low score. The following theorem states that the exponential mechanism satisfies differential privacy.

Theorem 3.3.6. (Differential privacy of the exponential mechanism)
Let $n, d \in \mathbb{N}$, let $X : \Omega \to \mathbb{R}^{n\times d}$ be a database, let $(M, \mathcal{A}, \mu)$ be a measure space, and let $q : \mathbb{R}^{n\times d} \times M \to \mathbb{R}$ be a map, and let $Q(\cdot \mid X)$ be an exponential mechanism. Define Δq as

$$\Delta q \coloneqq \sup_{\substack{(x,x')\in N(X)\\ r\in M}} \left|q(x,r) - q(x',r)\right|, \tag{3.6}$$

then $Q(\cdot \mid X)$ satisfies $(2\alpha\Delta q)$-differential privacy.

Proof. Let $(x, x') \in N(X)$, and let g_x and $g_{x'}$ be the densities such that

$$Q(B|X = x) = \int_B g_x(r)d\mu(r), \text{ and } Q(B|X = x') = \int_B g_{x'}(r)d\mu(r),$$

for any $B \in \mathcal{A}$. It is sufficient to show that

$$\frac{g_x(r)}{g_{x'}(r)} \leq \exp(2\alpha\Delta q),$$

since

$$\begin{aligned}\frac{g_x(r)}{g_{x'}(r)} \leq \exp(2\alpha\Delta q) \Leftrightarrow g_x(r) &\leq \exp(2\alpha\Delta q)g_{x'}(r)\\ &\Rightarrow \int_B g_x(r)d\mu(r) \leq \exp(2\alpha\Delta q)\int_B g_{x'}(r)d\mu(r)\\ &\Leftrightarrow \frac{\int_B g_x(r)d\mu(r)}{\int_B g_{x'}(r)d\mu(r)} = \frac{Q(B|X=x)}{Q(B|X=x')} \leq \exp(2\alpha\Delta q).\end{aligned}$$

Here the monotonicity of integrals have been used. The theorem follows therefore from the following reasoning

$$\begin{aligned}\frac{g(r)}{g'(r)} &= \frac{\left(\frac{\exp\left(\alpha q(x,r)\right)}{\int_M \exp\left(\alpha q(x,r)\right)d\mu(r)}\right)}{\left(\frac{\exp\left(\alpha q(x',r)\right)}{\int_M \exp\left(\alpha q(x',r)\right)d\mu(r)}\right)}\\ &= \exp\left(\alpha\left(q(x,r) - q(x',r)\right)\right)\frac{\int_M \exp\left(\alpha q(x',r)\right)d\mu(r)}{\int_M \exp\left(\alpha q(x,r)\right)d\mu(r)}.\end{aligned}$$

It follows from the construction of Δq, that

$$q(x', r) \leq \Delta q + q(x, r),$$

which implies that

$$\begin{aligned}\frac{\int_M \exp\left(\alpha q(x', r)\right) d\mu(r)}{\int_M \exp\left(\alpha q(x, r)\right) d\mu(r)} &\leq \frac{\int_M \exp\left(\alpha\left(\Delta q + q(x, r)\right)\right) d\mu(r)}{\int_M \exp\left(\alpha q(x, r)\right) d\mu(r)} \\ &= \exp(\alpha \Delta q).\end{aligned}$$

Hence,

$$\exp\left(\alpha\left(q(x, r) - q(x', r)\right)\right) \frac{\int_M \exp\left(\alpha q(x', r)\right) d\mu(r)}{\int_M \exp\left(\alpha q(x, r)\right) d\mu(r)} \leq \exp(2\alpha \Delta q),$$

which was to be demonstrated. ■

The Laplace mechanism works by adding Laplace noise to the output of a map, but since the Laplace distribution is a continuous distribution, this omits the possibility of using the Laplace mechanism on maps into countable sets. The exponential mechanism can easily be used for mapping into either countable or uncountable sets as long as q is well defined and μ exists.

4 PrivBayes

The `PrivBayes` mechanism is a solution proposed by Zhang et al. (2014), which utilizes Bayesian networks. The Bayesian networks are generated in a differentially private manner, and the conditional distributions are injected with noise, which all in all results in joint distributions that are generated differentially private. The following definition of `PrivBayes` is adapted from Zhang et al. (2014).

Algorithm 4.0.1. (PrivBayes)
Let $X : \Omega \to \mathbb{R}^{n \times d}$ with $n, d \in \mathbb{N}$ be a database, let $V = (X_1, X_2, \ldots, X_d)$ be a random vector that is equal in distribution to each row of X. The `PrivBayes` mechanism is given by the following algorithm

1. Fit a k-degree Bayesian network

$$\mathcal{N} = \left\{(1, \Pi_1), (2, \Pi_2), \ldots, (d, \Pi_d)\right\},$$

   that resembles the conditional independence relations in V for some chosen $k \in \mathbb{N}$, using an α_1-differentially private method.
2. Estimate conditional probabilities $\mathbb{P}(X_i | X_j, j \in \Pi_i)$ for all $i = 1, 2, \ldots, d$, and inject noise using an α_2-differentially private method.
3. Using the noisy conditional distributions, assemble a noisy joint distribution, and sample a synthetic dataset.

`PrivBayes` can be regarded as a parallel composition of an α_1-differentially private algorithm and an α_2-differentially private algorithm. This is because, the joint distribution that is assembled in step three of Algorithm 4.0.1 reveals both the fitted Bayesian network structure and the noisy conditional probabilites. As a direct consequence of Proposition 3.3.2 `PrivBayes` satisfies $(\alpha_1 + \alpha_2)$-differential privacy. An obvious candiate for conducting step one of Algorithm 4.0.1 is by using the exponential mechanism, and for step two the Laplace mechanism could be used. The following section dwelves into the implementation of `PrivBayes` using these methods.

4.1 Implementing PrivBayes

In this section we will discuss, how `PrivBayes` can be implemented for datasets consisting of columns with Bernoulli distributed entries. `PrivBayes` can also be adapted to work for non-binary attributes, but implementation of this will not be discussed in this project. The goal of the implementation is to test the power of an independence test on synthetic data,

similar to how it was done in Section 3.2 for the Spearman rank test. First, we will specify the data that will be used for synthetization. In this project, we will take basis in a database of $d \in \mathbb{N}$ attributes, where each attribute is a Bernoulli distributed random variable, that is, each attribute assumes either zero or one. Furthermore, the attributes are dependent on each other. In order to generate sample datasets like this, we use a copula technique. We could also have used Bayesian networks, but for the sake of simplicity we use the copula technique. This is because, we just want to generate dependent data, and also manually constructing the conditional distributions for a Bayesian network, can quickly turn into a complicated task. To generate a single row in the dataset, we start by sampling a vector from a multivariate normal distribution. We specify this random vector as $(Y_1, Y_2, \ldots, Y_d)$, and we let it have mean zero and some specified covariance matrix $A \in \mathbb{R}^{d \times d}$. Assuming that each Y_i has the same mean and variance, we can denote the marginal cdf as F, and obtain a random vector of correlated uniformly distributed variables in the following way,

$$(U_1, U_2, \ldots, U_d) \coloneqq \big(F(Y_1), F(Y_2), \ldots, F(Y_d)\big) .$$

We can now obtain a vector of Bernoulli distributed random variables with dependence by using the indicator function,

$$(\mathbb{1}_{\{U_1 \leq u_1\}}, \mathbb{1}_{\{U_2 \leq u_2\}}, \ldots, \mathbb{1}_{\{U_d \leq u_d\}}).$$

Here the thresholds $u_1, u_2, \ldots, u_d$ can be chosen freely in $[0, 1]$, but in this project we use $u_i = 0.5$ for all $i = 1, 2, \ldots, d$. This threshold is chosen to maximize the amounts of zeros and ones in each column, which makes the Pearson's test more robust when sampling from the perturbed conditional distributions. In essence, this helps to minimize the chance of having columns in the synthetic dataset that consists of only zeros or only ones. The sampling procedure is repeated $n \in \mathbb{N}$ times in order to construct a sample dataset of size $n \times d$.

4.1.1 Generating a Bayesian network differentially private

Zhang et al. (2014) show that Bayesian networks can be generated differentially private using the exponential mechanism. Let P_V denote the joint density for a stochastic vector $V = (X_1, X_2, \ldots, X_d)$ representing the attributes in some database that we wish to synthesize a private version of, and let $P_{\mathcal{N}}$ be the approximation of P_V based on the estimated conditionals specified by $\mathcal{N}$. Here $\mathcal{N}$ is a fitted k-degree Bayesian network that tries to specify the conditional independence in V. A well fitting Bayesian network is an $\mathcal{N}$ for which the difference between P_V and $P_{\mathcal{N}}$ is minimized. This difference can be measured by the Kullback-Leibler divergence, but in order to define it, the entropy and mutual information are defined. The following definition is adapted from Zhang et al. (2014).

Definition 4.1.1. (Entropy and mutual information)
Let $X : \Omega \to M$ be a random variable, where M is countable. The entropy of X is

given by

$$H(X) \coloneqq -\sum_{x \in M} \mathbb{P}(X = x) \log_2 \big(\mathbb{P}(X = x)\big). \tag{4.1}$$

Let $Y : \Omega \to M'$ be a random variable, where M' is countable. The mutual information between X and Y is given by

$$I(X, Y) \coloneqq \sum_{\substack{x \in M \\ y \in M'}} \mathbb{P}(X = x, Y = y) \log_2 \left(\frac{\mathbb{P}(X = x, Y = y)}{\mathbb{P}(X = x)\mathbb{P}(Y = y)} \right), \tag{4.2}$$

and here we use the convention that $0/0 = 0$. Note that for both the entropy and the mutual information, we also use the convention that $\log_2(0) = 0$.

The Kullback-Leibler divergence is a measure of the difference between two probability distributions. For the purpose of this project the following definition, adapted from Zhang et al. (2014) will be used, however for the original definition of the Kullback-Leibler divergence refer to Kullback and Leibler (1951).

Definition 4.1.2. (Kullback-Leibler divergence)
Let $V = (X_1, X_2, \ldots, X_d)$ for some $d \in \mathbb{N}$ be a stochastic vector, let P_V denote the joint density of V, let $\mathcal{N} = \big\{(1, \Pi_1), (2, \Pi_2), \ldots, (d, \Pi_d)\big\}$ be a k-degree Bayesian network, and let

$$P_{\mathcal{N}}(x) \coloneqq \prod_{i=1}^{d} \mathbb{P}(X_i = x_i | X_j = x_j, j \in \Pi_i),$$

for all $x \in \mathbb{R}^d$, where x_i and x_j are the i^{th} and j^{th} entry in x respectively. The Kullback-Leibler divergence is given by

$$D_{KL}(P_V || P_{\mathcal{N}}) \coloneqq -\sum_{i=1}^{d} I(X_i, \Pi_i) + \sum_{i=1}^{d} H(X_i) - H(V).$$

Here, $I(X_i, \Pi_i)$ is the mutual information between X_i and the stochastic vector $(X_j, j \in \Pi_i)$.

A well fitting $\mathcal{N}$ is found by minimizing $D_{KL}(P_V || P_{\mathcal{N}})$ with respect to $\mathcal{N}$. Since $H(X_i)$ and $H(V)$ are constant with respect to the choice of $\mathcal{N}$, it is sufficient to find an $\mathcal{N}$ that

maximizes

$$\sum_{i=1}^{d} I(X_i, \Pi_i).$$

As proposed by Zhang et al. (2014) this could be done by iterating from X_1 to X_d, and greedily choosing the parent set for each attribute that yields the largest $I(X_i, \Pi_i)$. In order to introduce differential privacy, the exponential mechanism can be used to pick the parent set for each attribute instead of choosing the parent set greedily. With regards to Definition 3.3.5, since we want to sample parent sets using the exponential mechanism, we let $M = \Omega_i$ be the set of all possible parent sets for X_i, that is, all subsets of $\{1, 2, \ldots, i-1\}$ containing at most k elements. We see that $\big(\Omega_i, \sigma(\Omega_i), \mu\big)$ is a measure space, when μ is the counting measure. Then for each i define the density g_{X_i} by

$$g_{X_i}(\Pi) \propto \exp\left(\frac{I(X_i, \Pi)}{2\gamma}\right), \tag{4.3}$$

for any $\Pi \in \Omega_i$ using the mutual information I as the score function q and Π as r in Definition 3.3.5. Furthermore

$$\gamma \coloneqq \frac{(d-1)\Delta I}{\alpha_1},$$

where ΔI is the largest possible distance in I as defined in (3.6). This density is integrable over Ω_i, since Ω_i is finite, and so is g_{x_i}. We see that defining Q_i as

$$Q_i(B|X_i) \coloneqq \int_B g_{X_i}(\Pi) d\mu(\Pi),$$

for $B \in \sigma(\Omega_i)$, yields the exponential mechanism satisfying $\alpha_1/(d-1)$-differential privacy. Since the parent set of X_1 is always $\emptyset$, we will let $\Pi_1 = \emptyset$, and sample Π_i from $Q_i(\cdot|X_i)$ for $i = 2, 3, \ldots, d$. By Proposition 3.3.2 the mechanism of generating $\mathcal{N}$ will satisfy α_1-differential privacy. This approach assumes that X_i is an ancestor to X_j for any $i \in \{1, 2, \ldots, j-1\}$, but this might not be a correct assumption, since the order of attributes can be arbitrary in practice. Thus, for a more flexible approach Bayesian networks should be fitted for all permutations $\pi : \mathbb{N} \to \mathbb{N}$. This would allow for structures, where $X_{\pi(i)}$ may be an ancestor to $X_{\pi(j)}$ for $i \in \{1, 2, \ldots, j-1\}$, even if $\pi(i) > \pi(j)$. Then the network with the smallest Kullback-Leibler divergence should be chosen. However, for the sake of simplicity, we will only use the permutation $\pi(i) \coloneqq i$.

Estimating the mutual information

To estimate (4.3), we need to estimate the mutual information. A way to estimate it is by rewriting the expression (4.2) in terms of entropy. This is useful since the entropy can easily be estimated for binary variables. Let $x \in \{0, 1\}^n$ be a vector of realisations for a

Bernoulli variable X, with x_i being the i^{th} entry. The probability that X assumes one can be estimated using by

$$\hat{\mathbb{P}}(X = 1) = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

and $\hat{\mathbb{P}}(X = 0) = 1 - \hat{\mathbb{P}}(X = 1)$. An estimate of the entropy is obtained by using the estimates $\hat{\mathbb{P}}(X = 1)$ and $\hat{\mathbb{P}}(X = 0)$ in (4.1). In the case where X is a vector of binary variables, say a vector of length $d \in \mathbb{N}$, we may consider it instead as a categorical variable assuming one out of 2^d possible states. Let $h : \{0,1\}^d \to \{0, 1, \ldots, 2^d - 1\}$ be a bijective map that assigns a number to each outcome of X. Then the estimate of $\mathbb{P}(X = h^{-1}(k)) = \mathbb{P}(h(X) = k)$ for $k = 0, 1, \ldots, 2^d - 1$ is

$$\hat{\mathbb{P}}(X = h^{-1}(k)) = \hat{\mathbb{P}}(h(X) = k) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{h(x_i)=k\}}.$$

In the same manner as before, this can be used in (4.1) to estimate the entropy of X. The expression (4.2) can be rewritten in terms of entropy in the following way

$$\begin{aligned}
I(X,Y) &= \sum_{\substack{x\in M\\ y\in M'}} \mathbb{P}(X = x, Y = y) \log_2\left(\frac{\mathbb{P}(X = x, Y = y)}{\mathbb{P}(X = x)\mathbb{P}(Y = y)}\right) \\
&= \sum_{\substack{x\in M\\ y\in M'}} \mathbb{P}(X = x, Y = y) \log_2\big(\mathbb{P}(X = x, Y = y)\big) \\
&\quad - \sum_{\substack{x\in M\\ y\in M'}} \mathbb{P}(X = x, Y = y) \log_2\big(\mathbb{P}(X = x)\big) \\
&\quad - \sum_{\substack{x\in M\\ y\in M'}} \mathbb{P}(X = x, Y = y) \log_2\big(\mathbb{P}(Y = y)\big) \\
&= H(X) + H(Y) - H(X,Y).
\end{aligned}$$

Here, $H(X,Y)$ is the entropy of the pair (X,Y). Using this formula, we can estimate the mutual information. Finally, we need to calculate ΔI. For this we use the following lemma, which is given in Zhang et al. (2014).

Lemma 4.1.3. (The sensitivity of the mutual information)
Let X be a database with $n \in \mathbb{N}$ rows and attributes X_1 and X_2. The sensitivity of the mutual information between X_1 and X_2 is given by

$$\Delta I = \begin{cases} \frac{1}{n}\log_2(n) + \frac{n-1}{n}\log_2\left(\frac{n}{n-1}\right), & \text{if } X_1 \text{ or } X_2 \text{ is binary,} \\ \frac{2}{n}\log_2\left(\frac{n+1}{2}\right) + \frac{n-1}{n}\log_2\left(\frac{n+1}{n-1}\right), & \text{otherwise.} \end{cases} \tag{4.4}$$

The proof is omitted, but can be found in the appendix of Zhang et al. (2014).

In this project, we take basis in attributes with binary outcomes, and therefore the first case in (4.4) will be used.

4.1.2 Estimating noisy conditional distributions

In this section, we will discuss the implementation of step two of Algorithm 4.0.1 for datasets with Bernoulli distributed entries, using the Laplace mechanism. Let $V = (X_1, X_2, \ldots, X_d)$ be a stochastic vector of Bernoulli distributed variables representing the attributes in a database with n rows. Assume that a Bayesian network has been fitted, and let it be denoted by

$$\mathcal{N} = \left\{(1, \Pi_1), (2, \Pi_2), \ldots, (d, \Pi_d)\right\}.$$

In this section, we will estimate the conditional distributions $\mathbb{P}(X_i|\Pi_i)$ for $i = 1, 2, \ldots, d$, and then noise will be injected into the distributions using the Laplace mechanism. Contingency tables will be used to represent the distributions. Consider the pair X_i and Π_i, where Π_i only contains one parent, say X_j. In this case, a contingency table for the distribution of (X_i, X_j) is given by

X_i \ X_j	0	1
0	$p_{0,0}$	$p_{0,1}$
1	$p_{1,0}$	$p_{1,1}$

where the estimates of the probabilities $p_{x,y}$ for $x, y \in \{0, 1\}$ are given by

$$\hat{p}_{x,y} \coloneqq \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}_{\left\{x_{k,i}=x \wedge x_{k,j}=y\right\}}. \tag{4.5}$$

Here $x_{k,i}$ and $x_{k,j}$ are the k^{th} observations of X_i and X_j in the observed dataset. By the definition of conditional probability we have that

$$\mathbb{P}(X_i|X_j) = \frac{\mathbb{P}(X_i, X_j)}{\mathbb{P}(X_j)}.$$

The distribution of X_j can be found by marginalizing X_i out of the probability table. This yields the following probabilites

$$\mathbb{P}(X_j = 0) = p_{0,0} + p_{1,0}, \text{ and } \mathbb{P}(X_j = 1) = p_{0,1} + p_{1,1}.$$

Dividing $\mathbb{P}(X_j)$ into each entry of the probability table yields a new table that represents the distribution $X_i|X_j$, which is what we wanted. In the case, where X_i has multiple

parents, say Π_i, the contingency table becomes a multidimensional table. The way of estimating the probabilites are the same as in (4.5), but with more variables. Again, we can marginalize X_i out of the joint distribution, divide through in the probability table, and then obtain the conditional distribution of $X_i|\Pi_i$. Finally, we need to inject some noise into the estimated probabilities. We do this with the Laplace mechanism to satisfy differential privacy. According to Theorem 3.3.4 we need to know the L_1-sensitivity of the contingency tables. Since each entry in the contingency table varies at most by $\frac{1}{n}$, the sensitivity must equal $\frac{1}{n} \cdot 2^{1+q_i}$, where $q_i \in \mathbb{N}$ is the number of parents to X_i. Laplace distributed noise with scale parameter equal to $2^{1+q_i}d/(n\alpha_2)$ is added to each entry in the probability table for X_i and this is done for all $i = 1, 2, \ldots, d$. This may yield entries that are outside the interval $[0, 1]$. To accomdate for this, negative values are set to zero, and then the probabilities are normalized. All in all, this procedure yields α_2-differential privacy.

4.2 Simulation study of PrivBayes

In this section we evaluate an implementation of `PrivBayes`. The implementation follows the description in the former section, and the source code is available in the following git-repository:

- *https://github.com/janreiter793/DPAndPrivBayesP9.git*

The implementation will be evaluated in three ways. First, a sanity check will be performed on the part that generates differentially private Bayesian networks. Then the relationship between the relative sizes of α_1 and α_2 of Algorithm 4.0.1 and the utility of the synthetic data is investigated. Finally, the relationship between the level of differential privacy and utility will be checked.

4.2.1 Sanity check for generating Bayesian networks

The code used for this section can be found in the file "`PrivBayes_sanity_check.R`" in the github-repository. The program takes basis in datasets of Bernoulli-variables that are generated using a copula that is based on a multivariate normal distribution with the following covariance matrix:

$$\begin{bmatrix} 1 & 0.6 & 0.5 & 0.4 & 0.3 \\ 0.6 & 1 & 0.6 & 0.5 & 0.4 \\ 0.5 & 0.6 & 1 & 0.6 & 0.5 \\ 0.4 & 0.5 & 0.6 & 1 & 0.6 \\ 0.3 & 0.4 & 0.5 & 0.6 & 1 \end{bmatrix}. \tag{4.6}$$

The datasets are of length 1000 and the synthetic datasets that are sampled are of the same length. The degree of the Bayesian networks that are fitted is 3. Fitting a Bayesian

network by greedily choosing each parent set that maximizes $I(X_i, \Pi_i)$ for all $i = 1, 2, \ldots, d$ yields Figure 4.1.

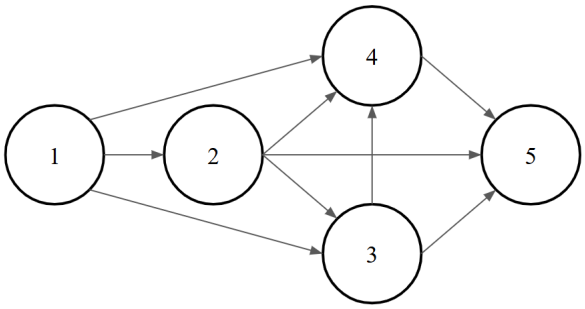
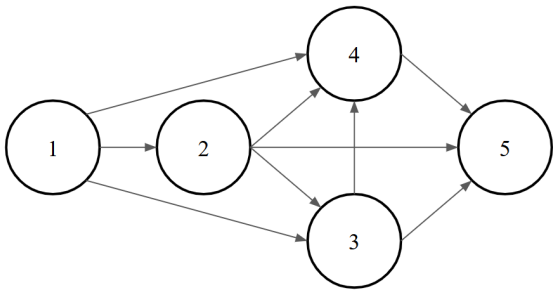
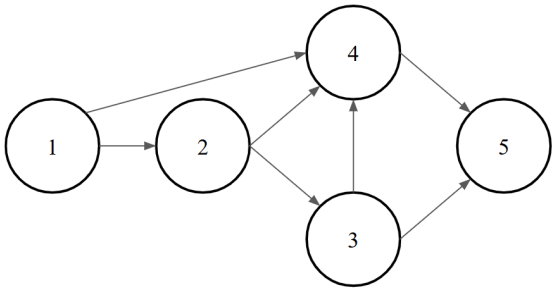


Figure 4.1: Bayesian network fitted to Bernoulli data that is generated using a copula technique. Each parent set is picked such that it maximizes $I(X_i, \Pi_i)$ for each $i = 1, 2, \ldots, d$.

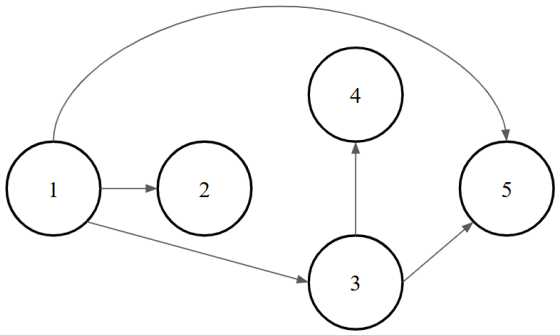
In Figure 4.2 the Bayesian networks that are fitted differentially private, are illustrated. It can be seen that at a differential privacy level of $\alpha = 50$ the network is identical to the greedily sampled one. At a level of 5 the Bayesian network has deteriorated a bit. This is an expected behavior, since the lower level of differential privacy decreases the amount of signal from the original dataset carried into the Bayesian network. At 0.5- and 0.05-differential privacy the Bayesian network changes even further. The parent sets have shrunk in size, and 1 has turned into a parent of 5, which might be a spurious relation. In summary, the implementation seems to behave as expected, since the differentially private Bayesian networks seem to resemble the greedily fitted network, mostly when the privacy is low and least when the privacy is high.



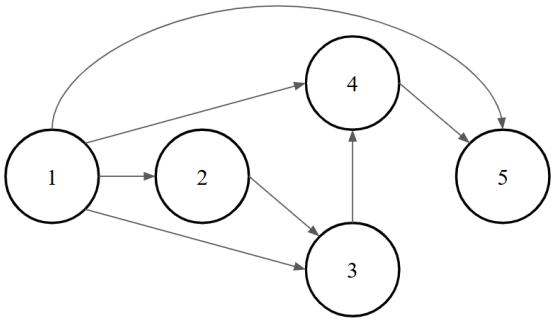
(a) Generated under 50-differential privacy.



(b) Generated under 5-differential privacy.



(c) Generated under 0.5-differential privacy.



(d) Generated under 0.05-differential privacy.

Figure 4.2: The Bayesian networks generated with different levels of differential privacy using the exponential mechanism.

4.2.2 The relationship between the mixture parameter and Pearson's test

In Algorithm 4.0.1 it is described that `PrivBayes` consists of an α_1- and an α_2-differentially private mechanism, which all in all yields a level of $\alpha := (\alpha_1 + \alpha_2)$-differential privacy for `PrivBayes`. Consider the mixture parameter $\beta \in [0, 1]$ defined by

$$\beta := \frac{\alpha_1}{\alpha}.$$

Using this parameter, we may reexpress the α_1 and α_2 in the following way

$$\alpha_1 = \beta\alpha, \text{ and } \alpha_2 = (1 - \beta)\alpha.$$

The mixture parameter shows how the privacy budget, α of `PrivBayes`, is distributed between step one and step two in Algorithm 4.0.1. In this section, we investigate how the mixture parameter affects the power of Pearson's χ^2-test for independence. This simulation study takes basis in datasets of five attributes and 1000 observations. These observations

are Bernoulli variables that are generated using a copula technique, which takes basis in a multivariate normal distribution correlated according to (4.6). `PrivBayes` is used to sample synthetic datasets of the same length differentially private, and then Pearson's χ^2-test is performed between the first and second variable, Z_1 and Z_2, the first and the third variable, Z_3, the first and the fourth variable, Z_4, and finally the first and the fifth variable, Z_5. This is done for all

$$\alpha \in \{0.1, 1, 10, 100\},$$

and for each α, we estimate the power for 100 β-values, which are equidistantly distributed in the interval $[0.1, 0.9]$ with 500 iterations for each pair (α, β). That is, for each pair (α, β) the procedure of generating a sample dataset, synthesizing a new dataset from the sample data using `PrivBayes`, and then performing the Pearson's χ^2-test, is done 500 times. An accepted null-hypothesis is considered as a type II error, since the original data sample is generated with variable dependence, and the χ^2-test on the originial data rejects the null-hypothesis $\sim 100\%$ of the time. The number of accepted null-hypotheses is counted and divided by 500 for each variable pair. This yields the proportion of misclassifications for each variable pair, and subtracting this number from one for each pair yields the power of the test. The results of the simulation study can be seen in Figure 4.3, Figure 4.4, Figure 4.5, and Figure 4.6.

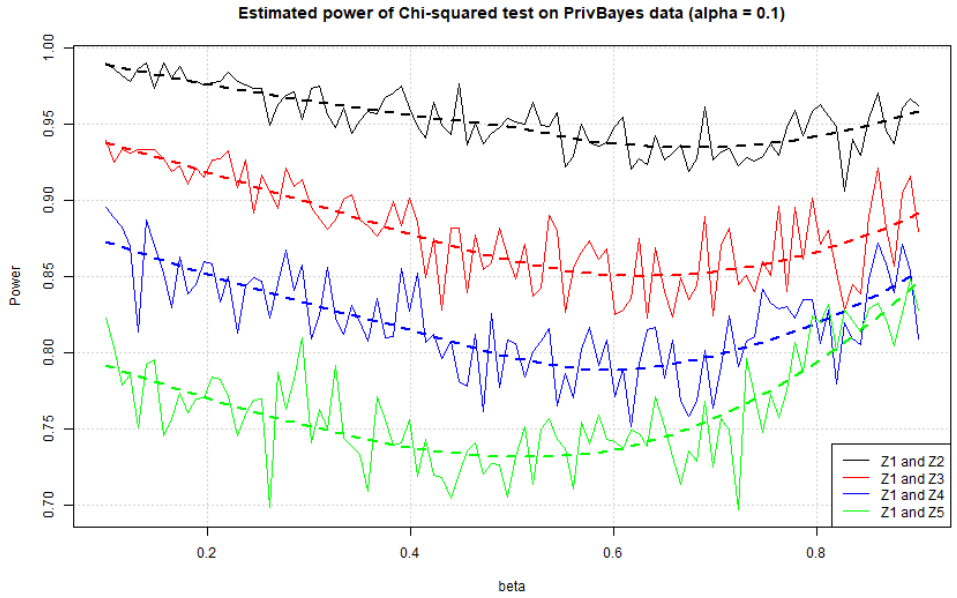


Figure 4.3: Power of Pearson's χ^2-test for different values of β with $\alpha = 0.1$. Lines are smoothed using *loess*-smoothing and illustrated with dashed lines.

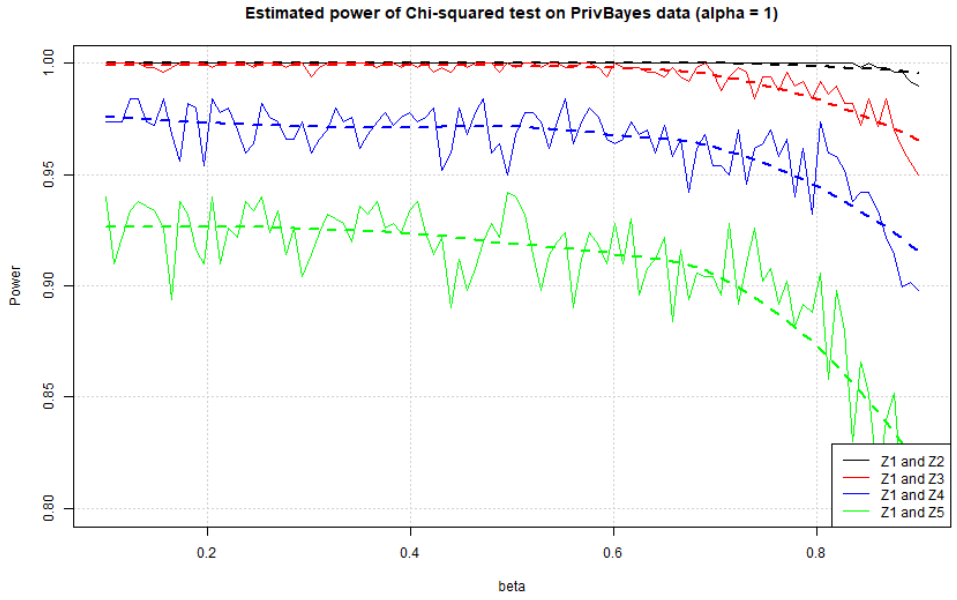


Figure 4.4: Power of Pearson's χ^2-test for different values of β with $\alpha = 1$. Lines are smoothed using *loess*-smoothing and illustrated with dashed lines.

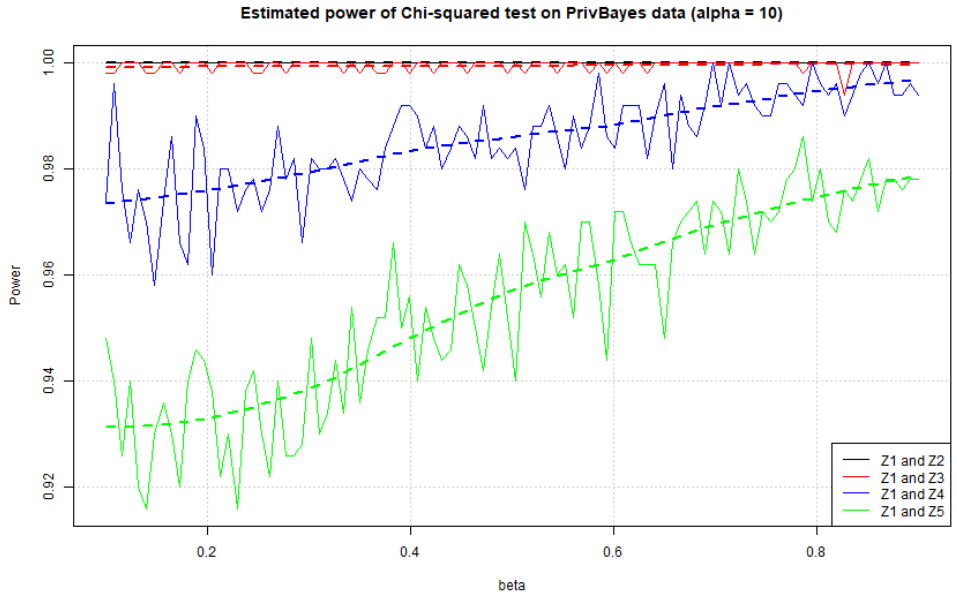


Figure 4.5: Power of Pearson's χ^2-test for different values of β with $\alpha = 10$. Lines are smoothed using *loess*-smoothing and illustrated with dashed lines.

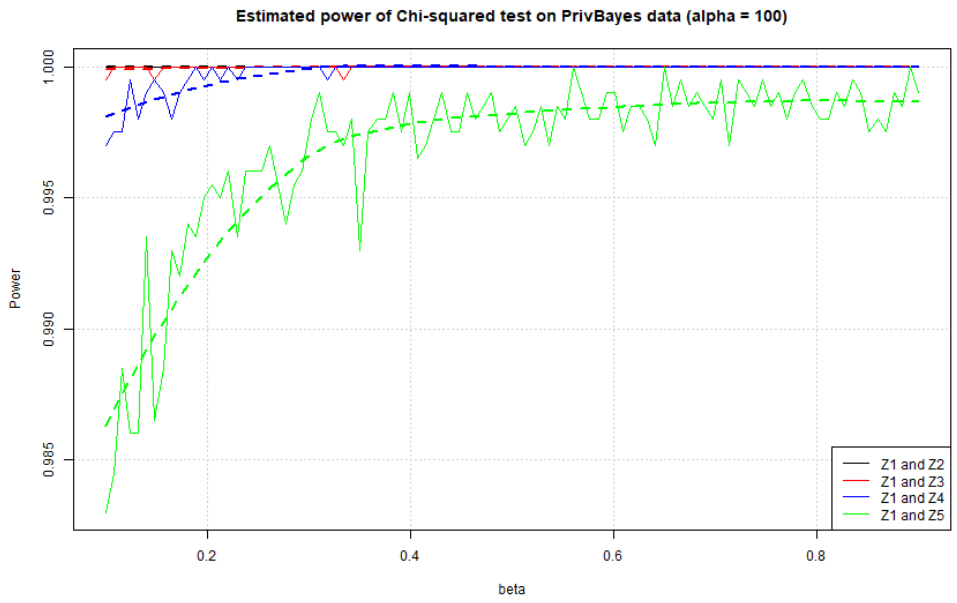


Figure 4.6: Power of Pearson's χ^2-test for different values of β with $\alpha = 100$. Lines are smoothed using *loess*-smoothing and illustrated with dashed lines.

The code used for generating the figures is available in "`PrivBayes_privacy_mixture.R`" in the github-repository. It appears that the relationship between the power and β depends on the privacy budget. In Figure 4.4, when β is close to one the power decreases. However, Figure 4.5 and Figure 4.6 show that the power is low when β is close to zero. This suggests that when α is high, it is more important that the Bayesian network is generated with low privacy than the noisy conditionals, however, when α is low, it is more important to prioritize low privacy for the noisy conditionals. In Figure 4.3 it seems that there is a U-shaped relationship between the power and β. This might be because, the privacy is so high that the Bayesian networks and the noisy conditionals deteriorate too much, and, hence, the best compromise is either a low og a high mixture parameter. The figures show in addition that the overall power increases as α increases. In other words, the power increases, as the privacy decreases.

4.2.3 The relationship between the privacy and Pearson's test

In this section we will investigate the relationship between the power of the Pearson's χ^2-test for independence and the privacy budget. We will use the same sample data as in the former section, but the simulationstudy will be run for pairs (α, β), where

$$\beta \in \{0.2, 0.5, 0.8\},$$

and for each β, we iterate over 1000 values of $\log_{10}(\alpha)$ equidistantly distributed in the interval $[-2, 2]$.

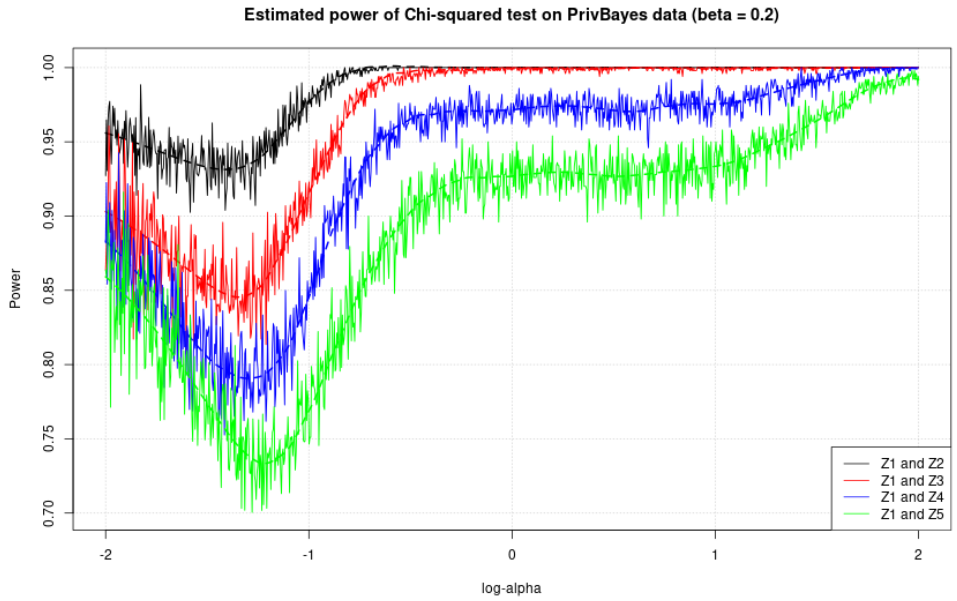


Figure 4.7: Power of Pearson's χ^2-test for different values of α with $\beta = 0.2$. Lines are smoothed using *loess*-smoothing and illustrated with dashed lines.

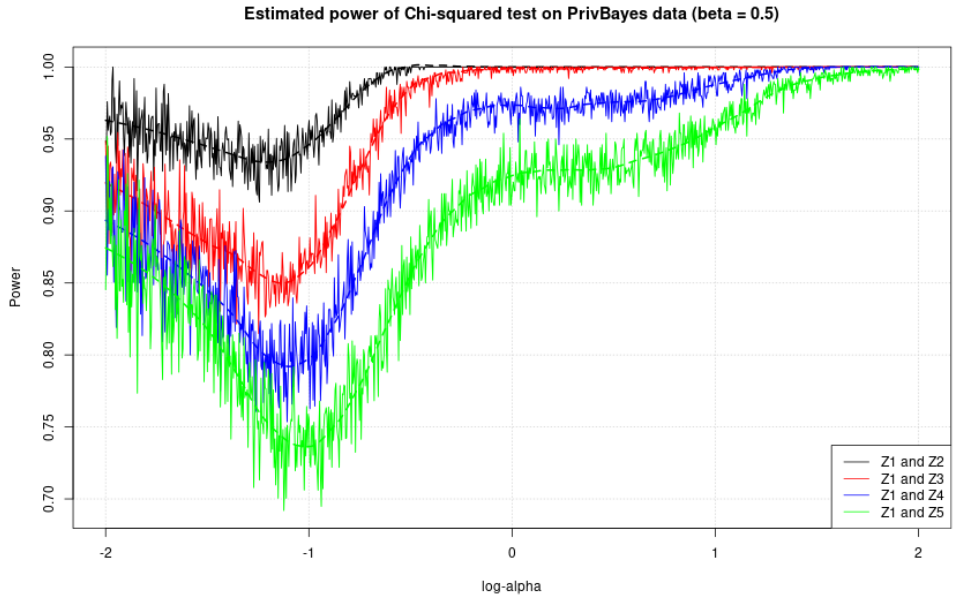


Figure 4.8: Power of Pearson's χ^2-test for different values of α with $\beta = 0.5$. Lines are smoothed using *loess*-smoothing and illustrated with dashed lines.

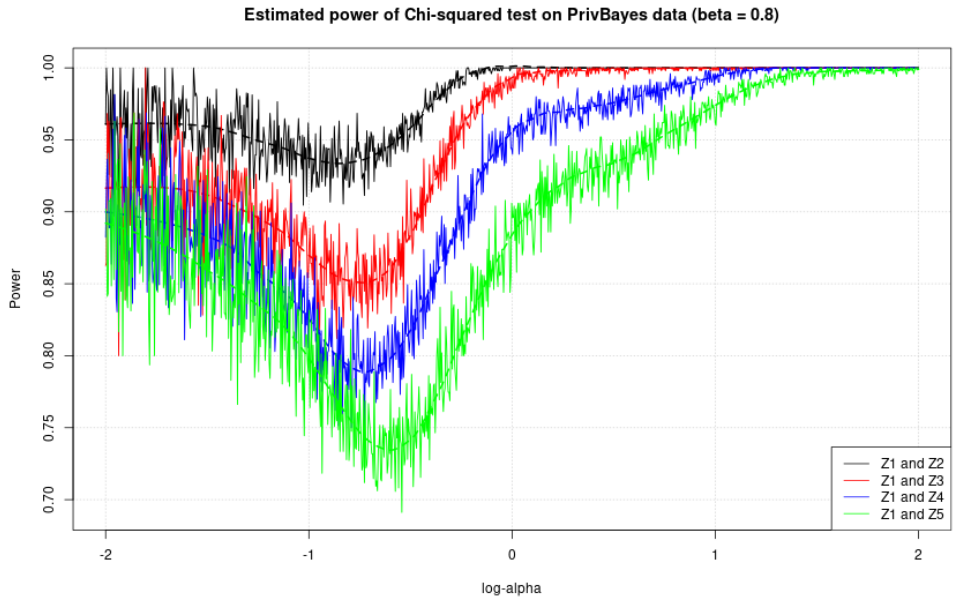


Figure 4.9: Power of Pearson's χ^2-test for different values of α with $\beta = 0.8$. Lines are smoothed using *loess*-smoothing and illustrated with dashed lines.

Again for each pair (α, β) the procedure of generating a sample dataset using the copula technique with the correlation matrix (4.6), then sampling a synthetic dataset using `PrivBayes`, and finally testing for independence using Pearson's χ^2-test, is repeated 500 times. The number of of accepted null-hypoteses is counted, divided by 500, and then subtracted from one to yield the power of the test. The results of the simulation study can be seen in Figure 4.7, Figure 4.8 and Figure 4.9.

When privacy increases, that is, the privacy budget α approaches zero, the power is expected to decrease. This is because the exponential mechanism is less likely to choose the most optimal Bayesian network, and the Laplace-mechanism injects noise with higher variance, when the privacy is high. This washes out the original signal in the data and blurs the dependence between the variables. This is partially apparent in the plots. In general it can be seen that the power decreses from right to left, but for all three plots the power starts to increase again at around $\log_{10}(\alpha) = -1$. This is in contradiction with the hypothesis that more privacy means less signal, but there may be a good explanation. It is likely that the increase is caused by the Laplace mechanism. When the privacy becomes high, the Laplace mechanism will inject noise with a large variance into the probability tables, and this makes it very likely that many of the probabilities become negative. When a probability becomes negative, it is set to zero, and because of normalization, its complement becomes one. When this happens, many of the entries in some columns of the synthetic dataset will become either ones or zeros. It is likely that this behavior causes the Pearson's test to reject independence more often, hence, the power will seem to increase, however, the utility of these tests is not good. To accomodate for this problem another mechanism

for injecting noise into the conditionals without the risk of translating the probabilities out of $[0, 1]$ is necessary, however, that is not within the scope of this project, and we will take basis in the results obtained using the Laplace-mechanism.

For Figure 4.7, Figure 4.8, and Figure 4.9 it can be seen that the power decreases, when $\log_{10}(\alpha)$ approaches -1 from the right. It seems that the power is slightly higher at around $\log_{10}(\alpha) = -1$ for $\beta = 0.2$ than $\beta = 0.5$ and $\beta = 0.8$. This conforms with what is seen in Figure 4.3 and Figure 4.4, which show that for low α it is important to have low values of β. From $\log_{10}(\alpha) = 2$ to $\log_{10}(\alpha) = 1$ it can be seen that the power decreases quicker for $\beta = 0.2$ and $\beta = 0.5$ than for $\beta = 0.8$. This agrees with Figure 4.5 and Figure 4.6, which show that it is important to not have a low β, when α is high. All in all, this show that when we want to generate synthetic data with `PrivBayes`, it is good to choose a high β-value when working with low privacy, that is high α-values. On the other side, it is good to choose a low β-value, when we want high privacy, since this yields a higher power.

5 Discussion

The definition of differential privacy states that there is an upper bound on how much the probability of obtaining some synthetic dataset can change when changing one row in the original dataset. High amounts of privacy means that the possible change in probability is low. From a Bayesian perspective this is not quite, how one would think about privacy. The Bayesian approach would instead be to measure what could be learned about the original dataset given a synthetic one. That is, how is the conditional distribution $\mathbb{P}(X|Z)$ given for some database X and some synthetic database Z. A way to asses privacy could be to calculate the probability that a single row of X given Z equals the correct observation, that is the probability $\mathbb{P}(X_i = x_i|Z)$, since this is the probability that an adversary would correctly guess the attributes of an individual in the dataset given a synthetic one. This sounds quite intuitive, but in practice using this measure of privacy instead of differential privacy might be putting the horse before the cart. Differential privacy considers the distribution of the synthetic data given a real observed dataset. This means that privacy can be assessed before generating the synthetic data, which allows the data releasing agent to pick a certain amount of privacy, generate a synthetic dataset, and then release it. The Bayesian privacy approach takes basis in the distribution of the observed dataset given the synthetic dataset. There are rarely closed forms for the probabilities of these distributions, and therefore estimation techniques are often required. It is therefore usually necessary to generate a synthetic dataset before the level of privacy can be estimated with the Bayesian approach, and that makes it difficult to obtain a synthetic dataset with a predetermined level of privacy.

As is mentioned in the introduction, an important aspect of synthetic data is the trade-off between privacy and utility. It was demonstrated in the simulation studies that the utility decreased as the privacy increased. This was also expected, since more privacy means more perturbation, which in the end blurs the characteristics and patterns of the original dataset. The simulation study of the smoothed histogram mechanism demonstrated that the power of the Spearman rank test for correlation remained constant at approximately 100% for low privacy, that is $\alpha > \sqrt{10}$. Then from $\alpha = \sqrt{10}$ to $\alpha = 1/\sqrt{10}$ the power drops drastically and levels off at around 0%. This shows that there can be limits to the amount of privacy imposed, before the data becomes completely useless. This is important to take into consideration, when determining the trade-off between privacy and utility.

The simulation study of `PrivBayes` demonstrated a more nuanced relationship between the utility and privacy budget. It showed that not only is the power of Pearson's χ^2-test dependent on the privacy budget, but also the mixture parameter. In particular, the relationship between the power of the test and the mixture parameter changed with respect

to the overall privacy budget. We saw from the simulations that when the privacy budget was low, it was more important to allocate more privacy towards the Bayesian network structure rather than the estimates of the conditional distribution functions. However for high privacy budgets, the simulations show that capturing the Bayesian network structure is more important than the noisy conditionals. These relationships might depend on the type of dataset, and the hypothesis that is being tested. Therefore an optimal choice of privacy budget and mixture parameter for this study might not be the same in other contexts. This also applies to the smoothed histogram method.

In summary, both the smoothed histogram method and `PrivBayes` are useful methods for synthesizing differentially private data. The synthetic data can be generated with high or low amounts of privacy, but specific amounts of privacy might do benefit in different contexts. For instance, when conducting light exploratory analysis it might suffice with medium or even high amounts of privacy, since overall trends and patterns may still be aparent. When working with highly private synthetic data, the blurring of characterstics and patterns should then be taken into account, thus, finding a mild trend might in fact suggest the presence of a strong trend in the real dataset. On the other hand, when using data to asses something critical, like for instance the pressence of a disease, a low amount of privacy might be necessary, since having a high power becomes important. Therefore, the amount of imposed privacy should not only be considered in context of the sensitiviy of the data, but also in context of the purpose of the data.

6 Conclusion

In this project, we began by defining differential privacy, and then we went on to introduce the smoothed histogram method. A mathematical proof for the differential privacy of the smoothed histogram mechanism was given, and this was utilized in a simulation study. The simulation study demonstrated the privacy-utility trade-off very well. It showed that increasing the privacy parameter α increases the utility, however, this in turn decreases the privacy of the synthetic data.

For the purpose of implementing `PrivBayes` the Laplace-mechanism and the exponential mechanism were introduced as well. Both mechanisms satisfy differential privacy, and this was also proven mathematically. After this, the `PrivBayes` mechanism was defined, and a solution for a practical implementation of `PrivBayes` was included. This solution however, was just a solution for implementing `PrivBayes` for datasets with Bernoulli distributed entries. Using the solution, `PrivBayes` was implemented using `R`, and all code was uploaded to Github. The simulation study of `PrivBayes` was seperated into the following three parts.

Sanity check for generating Bayesian networks

We fitted a Bayesian network using a greedy algorithm to find the best fitted Bayesian network. We saw that for a low amount of privacy the differentially private Bayesian network resembled the greedily fitted network. As expected for higher levels of privacy more deviations from the greedily fitted Bayesian network appeared.

The relationship between the mixture parameter and Pearson's test

The mixture parameter determined, how the privacy budget was distributed between the mechanism for generating the Bayesian network and the mechanism for estimating the noisy conditionals. We saw that for high levels of privacy, it was best to allocate more privacy towards the Bayesian network generating mechanism, suggesting that it was important to obtain good estimates of the noisy conditionals. On the other hand, for low levels of privacy it was important to choose a high mixture parameter, suggesting that it was important to estimate a good network structure.

The relationship between the privacy budget and Pearson's test

This study demonstrated the privacy-utility trade-off in the same manner, as the simulation study of the smoothed histogram mechanism did. This time however, there was overall a higher utility in the form of the power of Pearson's χ^2-test. The power did also not fall as drastically, when increasing privacy, as it did for the smoothed histogram mechanism.

This suggested that `PrivBayes` may have a higher utility with respect to privacy, than the smoothed histogram mechanism had. To state this as a fact, further investigation is needed, since the two simulation studies relied on different types of data and different hypothesis tests.

Contributions

This project unifies several topics from the literature about differential privacy on a common and rigorous mathematical foundation. For this purpose, the mathematics of the different topics have been elaborated further in this project, than what is found in the literature. This applies, for instance, to the proof of differential privacy for the exponential mechanism, which is virtually nonexistent in its original paper, McSherry and Talwar (2007). An unambiguous version of the composition theorem from Dwork et al. (2006) is also done and proven, and a more thorough explanation of the `PrivBayes` implementation is included as well. Implementations of the smoothed histogram method and `PrivBayes` are done using `R`, and made publicly available in the github-repository:

- *https://github.com/janreiter793/DPAndPrivBayesP9.git*

Results from simulation studies on the smoothed histogram mechanism and `PrivBayes` showing how, the power of two hypothesis tests is affected by the associated parameters, are included and discussed. All in all, this project serves as a solid mathematical introduction to differential privacy methods and `PrivBayes`, as well giving an insight into the utility of differentially private synthetic data.

Bibliography

[1] Jennifer A Bartell et al. A primer on synthetic health data. *arXiv*, 2024. URL https://doi.org/10.48550/arXiv.2401.17653.

[2] DST. *Hjemtagelse af analyseresultater*, Danmarks statistik, 2024. URL https://www.dst.dk/da/TilSalg/Forskningsservice/hjemtagelse-af-analyseresultater.

[3] Cynthia Dwork. Differential privacy. *Automata, Languages and Programming. 33rd International Colloquium, Proceedings, Part II*, pages 1–12, 2006. URL https://link.springer.com/chapter/10.1007/11787006_1.

[4] Cynthia Dwork et al. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, pages 17–51, 2006. URL https://people.csail.mit.edu/asmith/PS/sensitivity-tcc-final.pdf.

[5] Alexander S. Gillis et al. *What is a database*, 2024. URL https://www.techtarget.com/searchdatamanagement/definition/database.

[6] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models Principles and Techniques*. The MIT Press, 2009. ISBN 978-0-262-01319-2. URL https://mitpress.mit.edu/9780262013192/probabilistic-graphical-models/.

[7] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951. URL https://projecteuclid.org/journalArticle/Download?urlId=10.1214%2Faoms%2F1177729694.

[8] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. *Foundations of Computer Science*, pages 94–103, 2007. URL https://ieeexplore.ieee.org/document/4389483.

[9] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, 2008. URL 10.1109/SP.2008.33.

[10] Thomas Dyhre Nielsen and Finn Verner Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2007. ISBN 0-387-68281-3.

[11] Matthew Norton et al. Calculating cvar and bpoe for common probability distributions with application to portfolio optimization and density estimation. *Annals of Operations Research*, pages 1281–1315, 2019. URL https://doi.org/10.1007/s10479-019-03373-1.

[12] Jerome Reiter et al. Bayesian estimation of disclosure risks for multiply imputed, synthetic data. *Journal of Privacy and Confidentiality*, pages 17–33, 2014. URL https://journalprivacyconfidentiality.org/index.php/jpc/article/view/635.

[13] Yuchao Tao et al. Benchmarking differentially private synthetic data generation algorithms. 2022. URL https://doi.org/10.48550/arXiv.2112.09238.

[14] Larry Wasserman and Shuheng Zhou. A statistical framework for differential privacy. *arXiv*, pages 1–42, 2018. URL https://arxiv.org/abs/0811.2501.

[15] Jun Zhang et al. Privbayes: Private data release via bayesian networks. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1423–1434, 2014. URL http://wrap.warwick.ac.uk/63459.

A | Appendix

A.1 A PrivBayes implementation

The code listed in R-code A.1 demonstrates the implementation of `PrivBayes`, which this project takes basis in. At the top of the script the parameters for the privacy budget, mixture parameter, number of sampled observations etc. are found. The script is then separated into five sections. In the first section the sample data is generated using the copula technique, and threshold-value of 0.5. In the next section the code generates the Bayesian network differentially private using the exponential mechanism. In the third section the script estimates the conditional distributions, and then injects Laplace-noise to obtain differential privacy. In the fourth section the synthetic dataset is sampled using the Bayesian network, and the noisy conditionals. Finally, in the last section the program estimates a Bayesian network structure using a greedy algorithm. The listed code as well as all other code used for this project can be downloaded from

- *https://github.com/janreiter793/DPAndPrivBayesP9.git*

R-code A.1: `PrivBayes_prototype_II.R`

```
1  library(tidyverse)
2  library(MASS)
3  library(entropy)
4  library(magrittr)
5  library(VGAM)
6
7  # Parameters
8  eps <- 1      # Privacy budget
9  beta <- 0.5   # Mixture parameter
10 alpha_1 <- beta * eps # Level of diff-privacy for Bayesian
11                       # network-generation
12 alpha_2 <- (1 - beta) * eps # Level of diff-privacy for noise
13                             # injected in conditional props
14 k <- 3        # Degree of the Bayesian network
15 M <- 1000     # Number synthetic samples
16 n <- 1000     # Number of rows in original dataset
17 d <- 5        # Number of attributes (Will be binary)
18
19 # Correlation matrix
20 A <- matrix(
```

```
21    c(1.0, 0.6, 0.5, 0.4, 0.3,
22      0.6, 1.0, 0.6, 0.5, 0.4,
23      0.5, 0.6, 1.0, 0.6, 0.5,
24      0.4, 0.5, 0.6, 1.0, 0.6,
25      0.3, 0.4, 0.5, 0.6, 1.0),
26    nrow = d
27  )
28
29  # Test if matrix is symmetric and positive definite
30  all(A == t(A))
31  chol(A)
32
33  ######################### Generate Data ########################
34  # Generate data using copula-technique
35  X <-
36    mvrnorm(n = n,
37            mu = numeric(d),
38            Sigma = A) %>%
39    pnorm %>%
40    {. >= 0.5} %>%
41    as.numeric %>%
42    matrix(nrow = n,
43           ncol = d)
44
45  ############### Generate a Bayesian Network ##################
46  # Generate a Bayesian network differentially private using
47  # the exponential mechanism, and the mutual information as
48  # the scoring function.
49
50  N <- list() # Init the Bayesian network.
51  N[[1]] <- list(node = 1, parents = NULL) # First node has no
                                             # ancestors.
52
53  # Takes a node number and a set of potential parent-nodes, and
54  # then returns the estimated mutual information.
55  I <- function(node, parents) {
56    subset <- X[, c(node, parents)]
57
58    # Estimate the joint distribution by counting cases
59    joint_entropy <-
60      subset %>%
61      as.data.frame %>%
62      table %>%
```

```
63      prop.table %>%
64      entropy(method = "ML")
65
66    # Distribution of Y
67    Y_entropy <-
68      subset[, -1] %>%
69      as.data.frame %>%
70      table %>%
71      prop.table %>%
72      entropy(method = "ML")
73
74    # Distribution of the attribute
75    X_entropy <-
76      subset[, 1] %>%
77      as.data.frame %>%
78      table %>%
79      prop.table %>%
80      entropy(method = "ML")
81
82    # I(X, Y) = H(X) + H(Y) - H((X, Y))
83    return(X_entropy + Y_entropy - joint_entropy)
84  }
85
86  # The sensitivity of I according to Lemma 4.1 of the
87  # PrivBayes-article
88  Delta_I <- 1 / n * log2(n) + (n - 1) / n * log2(n / (n - 1))
89
90  # Takes a vector of elements v, and a size parameter s. Then
91  # it finds all subsets of v with at most s elements in them.
92  # Does not generate an empty subset, all nodes except first
93  # one must have parents
94  generate_subsets <- function(v, s) {
95    subsets <- list()
96    for(i in 1:min(s, length(v))) {
97      # Concatenate list of subsets with length i
98      subsets <- c(subsets,
99                   combn(v, i, simplify = FALSE))
100   }
101   return(subsets)
102 }
103
104 # Takes a vector of unnormalized density values and normalizes
105 # it
```

```
106 normalize <- function(vec) {
107   return(vec / sum(vec))
108 }
109
110 # This part generates the Bayesian network
111 gamma <- (d - 1) * Delta_I / alpha_1
112 for(i in 2:d) {
113   subsets <- generate_subsets(1:(i - 1), k)
114   g_x <- subsets %>% length %>% numeric
115   for(j in 1:length(subsets)) {
116     g_x[j] <- exp(I(i, subsets[[j]]) / (2 * gamma))
117   }
118
119   g_x %<>% normalize
120   N[[i]] <- list(node = i,
121                  parents = subsets[[
122                    sample(1:length(g_x), size = 1, prob = g_x)
123                    ]])
124 }
125
126 ############## Estimate the noisy conditionals ################
127 # Estimate the noisy conditionals based on the structure of
128 # the Bayesian network obtained in the former section. Inj-
129 # ects noise using the Laplace-mechanism
130 conditional_props <- list()
131 conditional_props[[1]] <- table(X[,1]) / n
132
133 # Takes a probability table representing a joint distribution.
134 # Marginalizes first variable out, and returns probability
135 # table representing joint distribution of the rest of the
136 # variables
137 marginalizeFirstVar <- function(tab) {
138   temp <- array(dim = dim(tab)[-1])
139
140   temp_entries <-
141     temp %>%
142     dim %>%
143     lapply(seq_len) %>%
144     expand.grid
145
146   for(i in 1:nrow(temp_entries)) {
147     temp_coord <- temp_entries[i, ] %>% as.numeric
148     temp <-
```

```
149       do.call("[<-",
150               c(list(temp),
151                 as.list(temp_coord),
152                 list(
153                   do.call("[", c(list(tab), c(list(TRUE),
154                   temp_coord))) %>%
155                   sum
156                 ))
157       )
158   }
159
160   return(temp)
161 }
162
163 # Constructs arrays that resemble the conditional
164 # distributions of the attributes
165 for(i in 2:d) {
166   conditional_props[[i]] <-
167     table(as.data.frame(X[, c(N[[i]]$node,
168                               N[[i]]$parents)])) / n
169   marg_density <- marginalizeFirstVar(conditional_props[[i]])
170   indices_1 <-
171     c(list(1), rep(TRUE,
172                    length(dim(conditional_props[[i]])) - 1))
173   indices_2 <-
174     c(list(2), rep(TRUE,
175                    length(dim(conditional_props[[i]])) - 1))
176
177   conditional_props[[i]] <-
178     do.call("[<-",
179             c(list(conditional_props[[i]]),
180               indices_1,
181               list(
182                 do.call("[",
183                         c(list(conditional_props[[i]]),
184                           indices_1)) /
185                   marg_density
186                 )
187               )
188             )
189
190   conditional_props[[i]] <-
191     do.call("[<-",
```

```
            c(list(conditional_props[[i]]),
              indices_2,
              list(
                do.call("[",
                        c(list(conditional_props[[i]]),
                          indices_2)) /
                  marg_density
                )
              )
            )
}

# Inject Laplace-noise into the arrays
for(i in 1:d) {
  if(is.numeric(conditional_props[[i]])) {
    conditional_props[[i]] <-
      conditional_props[[i]] +
      rlaplace(length(conditional_props[[i]]),
               location = 0,
               scale = d / n * 2 / alpha_2)
  } else {
    laplace <-
      dim(conditional_props[[i]]) %>%
      prod %>%
      rlaplace(scale = d / n *
        2^length(dim(conditional_props[[i]])) /
                 alpha_2) %>%
      array(dim = dim(conditional_props[[i]]))

    conditional_props[[i]] <-
      conditional_props[[i]] + laplace
  }

  # If estimate is below zero set to zero, and normalize
  conditional_props[[i]][conditional_props[[i]] < 0] <- 0
  indices_1 <-
    c(list(1), rep(TRUE,
                   length(dim(conditional_props[[i]])) - 1))
  indices_2 <-
    c(list(2), rep(TRUE,
                   length(dim(conditional_props[[i]])) - 1))
  normalizing_const <-
    1 / (do.call("[",
```

```
                  c(list(conditional_props[[i]]),
                    indices_1)) +
          do.call("[",
                  c(list(conditional_props[[i]]),
                    indices_2))
        )
  conditional_props[[i]] <-
    do.call("[",
            c(list(conditional_props[[i]]),
              indices_2)) *
    normalizing_const
  conditional_props[[i]][
    is.infinite(conditional_props[[i]])
    ] <- 1
  conditional_props[[i]][
    is.nan(conditional_props[[i]])
    ] <- 0
}

################ Sample a synthetic dataset ##################
# This function samples a single sample from conditional_props
sampleObservation <- function() {
    synth_sample <- numeric(d)

    # The first node has no parents and is therefore sampled
    # directly from its distribution
    synth_sample[1] <- rbinom(1, 1,
                              prob = conditional_props[[1]])

    # For each node sample based on conditional_props
    for(i in 2:d) {
      parent_samples <- as.list(
        synth_sample[N[[i]]$parents] + 1
        )
      prob <-
        do.call("[",
                c(list(conditional_props[[i]]),
                  c(parent_samples))
        )
      synth_sample[i] <- rbinom(1, 1, prob)
    }

    return(synth_sample)
```

```
278 }
279
280 Z <- matrix(ncol = d, nrow = M)
281 for(i in 1:M) {
282   Z[i,] <- sampleObservation()
283 }
284
285 # Takes a dataframe of bernoulli columns, then returns
286 # matrix with 1's and 0's, where a 1 represents an accepted
287 # null-hypothesis and 0 represents a rejected null-hypothesis
288 chisquare_test <- function(df) {
289   temp <- matrix(nrow = d, ncol = d)
290   for(i in 1:d) {
291     for(j in 1:d) {
292       tab <- table(df[,i], df[,j])
293       temp[i, j] <- chisq.test(tab)$p.value >= 0.05
294     }
295   }
296
297   return(temp + 0)
298 }
299
300 chisquare_test(Z)
301 chisquare_test(X)
302
303 ######################## Greedy Bayes ############################
304 N_greedy <- list() # Init the Bayesian network.
305 N_greedy[[1]] <- list(node = 1, parents = NULL) # First node
306                                                 # has no
307                                                 # ancestors.
308 for(i in 2:d) {
309   subsets <- generate_subsets(1:(i - 1), k)
310   mutual_informations <- subsets %>% length %>% numeric
311   for(j in 1:length(subsets)) {
312     mutual_informations[j] <- I(i, subsets[[j]])
313   }
314
315   N_greedy[[i]] <- list(node = i,
316                         parents = subsets[[
317                           which.max(mutual_informations)
318                         ]])
319 }
```