The Pennsylvania State University
The Graduate School

DIFFERENTIALLY PRIVATE HYPOTHESIS TESTING FOR NORMAL RANDOM VARIABLES

A Thesis in
Statistics
by
Eftychia Solea

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2014

The thesis of Eftychia Solea was reviewed and approved* by the following:

Aleksandra Slavkovic
Associate Professor of Statistics

Bing Li
Professor of Statistics

David Hunter
Head of Department of Statistics

*Signatures are on file in the Graduate School.

Abstract

Data privacy has become a fundamental problem in statistical data analysis. Consider a database that contains sensitive information. The goal of "private" data analysis is to publish valid statistical results without compromising the privacy of the individuals whose data are stored in the dataset.

In this thesis, we design private algorithms with rigorous privacy guarantees. The formal privacy notion we use is differential privacy [Dwork et al., [6]], which has received significant attention because it offers meaningful guarantees of privacy in the presence of any external or auxiliary of information. Differential privacy guarantees that no more information can be extracted about an individual from the statistical database than what is already available in the presence of any external information. This is critical in fields that handle sensitive data, such as medical data, financial data and personal data from social networking sites.

In our work, we design differentially private estimators for statistical inference. We focus on normal distribution and we compare the performance of the differentially private estimators with the classical maximum likelihood estimator. We also propose approximate sample size adjustment factors needed for sample size calculation in classical hypothesis testing to achieve certain power and Type I error. Finally, we illustrate the effect of privacy on Type I and Type II error rates for two statistical hypothesis procedures subject to differential privacy.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank Aleksandra Sesa Slavkovic for being an amazing advisor to me. She generously gave her time, knowledge and advice to help me completing this project. I am also very grateful for her being understanding and for giving me encouregement and confidence over these years. I would also like to thank Vishesh Karwa for his knowledge about differential privacy and his insightful discussions and suggestions about this project. Thanks also to the entire Department of Statistics at Penn State University. Part of this work was supported by the National Science Foundation Grant NSF BCS-0941553. Lastly, I would like to thank my friend Eliana Christou for her continuous support.

Chapter 1

Introduction

Rapid advances in technology for data collection, storage and analysis, especially of sensitive and personal data have made privacy an increasingly challenging problem. Potential for analyzing these data and impact on policy is great, but strong protection of the privacy of the individuals whose data are shared is still lacking. As the amount of data in the public realm accumulates and record linkage methodologies improve, the threat to confidentiality and privacy magnifies. Past studies demonstrate that even anonymization techniques can leak sensitive important information when the intruder has partial knowledge about the data from external sources [Fienberg and Slavkovic, [12]]. Therefore, statistical strategies have been developed with the goal of using data for statistical analysis, with the promise of protecting the privacy of individuals whose data are stored in the databases [e.g. see Fienberg and Slavkovic, [12], Hundepool et al., [13], Wasserman and Zhou, [24] for review of some of these techniques].

There is a rich literature of work on data confidentiality from the statistics community and from diverse areas of computer science such as algorithms and cryptography. Many techniques have been proposed from both communities on protecting privacy of individuals while analyzing big data sets. In statistics, disclosure-limitation techniques have been developed that investigate the trade-off between disclosure risk and utility of the protected released data. In computer science the focus has been on the development of privacy-preserving data mining (PPDM) algorithms that provide strong privacy (such as k-anonymization) to the data and more recently on formal definition of privacy.

The goal of Statistical Disclosure Control (SDC) or Statistical Disclosure Limitation (SDL) is to protect statistical data in such a way that they can be released without giving away confidential information of the respondents. SDC approaches optimize the trade-off between disclosure risk and data utility. Data utility is a measure of usefulness of a dataset for an intended analyst. A disclosure occurs when a person learns something that they did not know already about another person, via released data. There are three kinds of disclosure: identity, attribute and inferential disclosure. Disclosure risk measures the degree to which a dataset and its released statistics reveal sensitive information and can lead to re-identification; a disclosure risk could be viewed as a probability of re-identification. For more details on SDC and related concepts see Hundepool et al., [13], Fienberg and Slavkovic, [12].

SDC methods minimize the disclosure risk to an acceptable level while releasing as much information as possible. There are two types of SDC methods; perturbative and non-perturbative methods. Perturbative methods falsify the data before publication by introducing a small amount of noise. For example, matrix masking involves transforming the original data $X \in \mathbb{R}^{n\times p}$ through pre-post multiplication and the possible addition of noise. That is, $X' = AXB + E$ where $A \in \mathbb{R}^{q\times n}$ and $B \in \mathbb{R}^{p\times k}$ matrices and E is the matrix noise [Fienberg and Slavkovic, [12]]. Non-perturbative methods reduce the amount of information released by suppression or aggregation of the data X and creation of synthetic data; see Reiter (2005) for generation of synthetic microdata and Slavkovic and Lee (2009) for creation of synthetic tables.

Privacy-preserving data mining methods (PPDM) aim to constuct efficient anonymization algorithms while maintaining privacy. For example, k-anonymity technique advocates release of data such that there at least $k-1$ records with same features as the records we are trying to protect. However, k-anonymity has been criticized as being necessary but sometimes not sufficient to protect privacy (see Xiao and Tao (2007) and Hundepool et al, [13]).

Both the PPDM and SDC methods focus on obtaining valid statistical results without compromising the privacy of the individuals whose data are stored in the databases. However, these schemes are often criticized for lacking formal privacy guarantees and offering secured privacy only against a specific kinds of attacks

[Smith [20], Vu and Slavkovic, [23]]. A concept called *differential privacy* [Dwork, [6]] tackles privacy from a different perspective. Differential privacy has emerged from the cryprographic community and provides a mathematically provable and rigorous formal definition of privacy. It seeks to guarantee that by changing one single entry of the database any information extracted from the database will remain close to what it was before, even in the presence of any external information [Hundepool et al., [13], Dwork [6]].

1.1 Our contribution

In this thesis we use the notion of *differential privacy* as the formal concept of privacy. Intuitively, differential privacy guarantees that a user does not gain any more knowledge about an individual, regardless if this individual supplied his actual or fake information in the database. Hence, differential privacy is a rigid guarantee since it is independent on any additional information available to the attacker. Some recent developments due to Smith [20], Dwork and Lei, [11], Wasserman et al., [24] and [Vu and Slavkovic, [23] try to relate differential privacy to the traditional statistical inference.

In our work, we build on work of Smith [20] and Vu and Slavkovic, [23] and we design differentially private estimators when the data are sampled from the normal distribution and perform classical hypothesis testing under the framework of differential privacy. Because of differential privacy, the proposed algorithms ensure that an attacker does not learn anything new about an individual than what he already knew regardless of the individuals data being in the database or not. We evaluate the usefulness for statistical inference via a number set of simulation studies by investigating the quality of the proposed differentially private estimators.

It is well known that the evaluation of sample size and power is a crucial element in the planning of a statistical study. This typically involves controlling the risks of making Type I and Type II errors by ensuring that the sample size is large enough to detect important differences in the effects of interest with high probability [Neter et al., [16]]. In our work we derive sample size adjustment factors for sample size calculation and power analysis under the framework of differential privacy.

1.2 Structure of this thesis

Chapter 2: In this chapter we provide an overview of differential privacy. First, we provide the definition of differential privacy and describe the mechanisms that are used to create differentially private algorithms. We also outline some applications of differential privacy in the context of classical statistical inference.

Chapter 3: In this chapter, we design differentially private estimators for the normal distribution and we establish some asymptotic properties following the work of Smith [20]. Moreover, we extend the work of Vu and Slavkovic, [23] on the calculation of sample size under differential privacy in classical hypothesis testing. We apply their method when the data are sampled from the normal distribution. We propose new differentially private test statistics and evaluate Type I and Type II errors rates for two hypothesis testing procedures for testing the mean of the normal distribution.

Chapter 4: In Chapter 4, we demonstrate that we can design better differentially private algorithms than that in Chapter 3. In this section, we present algorithms with instance-based additive noise. We show that the algorithms satisfy a stronger notion of privacy and add less noise (be more accurate) compared to the algorithms in chapter 3.

Chapter 5: We conclude this thesis by summarizing our results and discussing future research problems.

Chapter 2

Differential Privacy

2.1 Definition

The definition of differential privacy uses a bound on the change of the distribution of any output for two data sets $\mathbf{x}$ and $\mathbf{x}'$ that differ at most in one element. More formally, differential privacy ensures that by changing one element in the data set $\mathbf{x}$, the distribution of the output does not change very much. Intuitively, a differentially private mechanism describes that any adversary does not gain more knowledge about an individual, whether the individual provides his actual or fake information in the data set. Thus, differential privacy guarantees that the risk of an individual's privacy is the same regardless if one individual's data are in the database or not [Dwork [6]].

Definition 2.1.1 (Dwork [6])**.** A randomized function $\mathcal{K}$ is (ϵ, δ)-differentially private if for any two data sets $\mathbf{x}$ and $\mathbf{x}'$ that differ only on one element and for all $S \subseteq \text{range}(\mathcal{K})$,

$$P(\mathcal{K}(\mathbf{x}) \in S) \leq e^{\epsilon} P(\mathcal{K}(\mathbf{x}') \in S) + \delta.$$

When $\delta = 0$, we say the randomized function is ϵ-differentially private. ϵ is a privacy parameter that measures the amount of information leakage; small ϵ, means strong privacy guarantee.

As we will see in the next section, from the statistics perspective the data are fixed and the randomness comes from the disclosure method, that is from the privacy mechanism.

2.2 Achieving Differential Privacy

In this section, we describe mechanisms, due to Dwork [6], Nissim et al., [17], McSherry and Talwar, [15], Wasserman and Zhou, [24] that guarantee differential privacy for all real-valued functions f defined on the set of all data sets $\mathcal{D}$.

2.2.1 Laplace Mechanism and ℓ_1-Sensitivity

Dwork [6] proposed the Laplace mechanism that achieves ϵ-differential privacy for any real-valued function f defined on $\mathcal{D}$ that we want to release by adding Laplace noise proportional to the sensitivity of the function f. Sensitivity measures the maximum change in the value of f due to the change of one entry in the data set $\mathbf{x}$.

Definition 2.2.1 (Global Sensitivity, Dwork [6])**.** For every function $\mathbf{f} : \mathcal{D} \longmapsto \mathbb{R}^k$ the global sensitivity of $\mathbf{f}$, denoted by $GS_{\mathbf{f}}$ is:

$$GS_{\mathbf{f}} = \max||\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}')||_1$$

$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{D}$ differing in at most one element.

Suppose that we want to release the real-valued function f defined on the data $\mathbf{x}$. For example, in statistics, the function f can be a sufficient statistic or a point estimator. The Laplace mechanism K_f adds a random noise from the Laplace distribution with mean 0 and standard deviation $\frac{GS_f}{\epsilon}$ to the value of f, that is $f(\mathbf{x})$, and releases $K_f(\mathbf{x}) = f(\mathbf{x})+\text{Lap}(\frac{GS_f}{\epsilon})$. The Laplace distribution with mean 0 and standard deviation b has density at y given by $h(y) = \frac{1}{2b}e^{\frac{-|y|}{b}}$.

Extending to the multivariate case, if $\mathbf{f} = (f_1, ..., f_k)$ is a vector of real-valued functions, the Laplace mechanism $K_{\mathbf{f}}$ releases $K_{\mathbf{f}}(\mathbf{x}) = \mathbf{f}(\mathbf{x})+\text{Lap}(\frac{GS_{\mathbf{f}}}{\epsilon})^k$ where $\text{Lap}(b)^k$ denotes k i.i.d random variables from the Laplace distribution. [Dwork et al., [8]] showed that the Laplace Mechanism results in ϵ-differential privacy.

Theorem 2.2.1 (Dwork et al., [8])**.** For $\mathbf{f} : \mathcal{D} \longmapsto \mathbb{R}^k$, the Laplace mechanism $K_{\mathbf{f}}$ that adds independently generated noise from $\text{Lap}(\frac{GS_{\mathbf{f}}}{\epsilon})$ to each component is ϵ-differentially private.

The idea of preserving privacy by adding noise to the released true answer $f(\mathbf{x})$ is not new (for references, see Denning [5] and Adam and Wortmann, [1]). However, the K mechanism provides privacy guarantees and excellent accuracy for functions with low global sensitivity. In particular, the noise added to ensure differential privacy depends only on the sensitivity of the function and on the parameter ϵ. Both are independent of the database and the number of data it contains. Thus, for insensitive functions the mechanism introduce the least noise even if the database is very large [Dwork [9], [10]]. Examples of functions that can be released with less error are means, variances, histograms, contingency tables and the single value decomposition [Dwork and Smith, [8], Vu and Slavkovic, [23], Dwork et al., [7], Dwork and Lei, [11]].

2.2.2 Output Perturbation

The *output perturbation* technique releases a specified function f of the data $\mathbf{x}$ after the addition of a random noise while preserving differential privacy. The magnitude of the noise is determined by the global sensitivity of f and the privacy parameter ϵ [Dwork et al., [7]]. Suppose we would like to release a statistic f. Then the output perturbation mechanism A_f releases $A_f(\mathbf{x}) = f(\mathbf{x}) + Y$ where Y is a random variable drawn from a noise distribution with standard deviation proportional to the global sensitivity GS_f of f. The Laplace mechanism can be considered as a special case of output perturbation with Laplace noise distribution, and it leads to ϵ-differential privacy as stated in Theorem 2.2.1.

2.2.3 Smooth Sensitivity

The output perturbation framework described above works well for functions insensitive to changes of one entry in the database. For many functions, such as the median, this approach yields high noise. Nissim et al., [17] develop mechanisms by adding noise to the value of a real-valued function calibrated to the *local sensitivity*, rather than the global sensitivity [Nissim et al., [17], Dwork and Smith, [8]]. However, the local sensitivity depends on the database, hence the challenge is to ensure that the noise magnitude does not leak any information about the database. To prevent this, they calibrate the noise with the *smooth sensitivity* of

f on the database $\mathbf{x}$ that smooths the change in magnitude of noise. We define smooth sensitivity in Chapter 4.

2.2.4 Sample and Aggregate Framework

The idea of the *sample and aggregate* method (Figure 2.1) is that if $f(\mathbf{x})$ is well approximated by evaluating $f(\mathbf{x})$ on subsamples from $\mathbf{x}$, then we can replace f by another function whose sensitivity is lower and more efficiently computable [Nissim et al, [17]]. The first step in the sample and aggregate method is to partition the database $\mathbf{x}$ into m smaller data sets $\mathbf{x}_1, .., \mathbf{x}_m$ with $\frac{n}{m}$ points per database and evaluate f on all these data sets to get the outputs $z_1, .., z_m$. Finally, if g is an appropriately chosen function defined on $z_1, ..., z_m$ called the aggregation function, according to the output perturbation framework the output of this algorithm is $A(\mathbf{x}) = g(z_1, ..., z_m) + Y$. Nissim et al., [17] showed that the sample and aggregate mechanism is ϵ-differentially private.

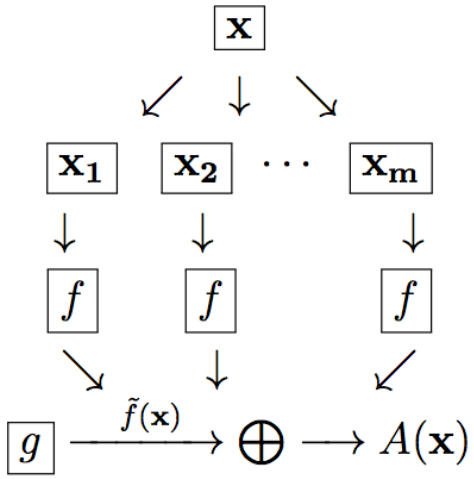


Figure 2.1: Sample and Aggregate Framework

2.2.5 Exponential Mechanism

McSherry et al., [15] proposed the exponential mechanism, a generic output release mechanism that also satisfies differential privacy. Unlike the output perturbation technique, the output of the exponential mechanism is not necessarily numeric.

This has led to some remarkable results such as density estimation with differential privacy [Wasserman and Zhou, 2007], a learning theory approach to non-interactive database privacy [Blum et al., 2011], high-dimensional sparse regression [Kifer et al., 2012] and differential privacy for functional data [Hall et al., 2012].

The exponential mechanism is defined by a score function $q : \mathcal{D} \times \mathcal{Z} :\longmapsto \mathbb{R}$ that assigns a real-valued score $q(\mathbf{x}, z)$ to any pair $(\mathbf{x}, z)$ from $\mathcal{D} \times \mathcal{Z}$ where $\mathcal{D}$ denotes the set of data sets and $\mathcal{Z}$ the range of z. Given $\mathbf{x}$ and a privacy parameter ϵ, the goal of the exponential mechanism is to release an output z in $\mathcal{Z}$ such that $q(\mathbf{x}, z)$ is maximized while guaranteeing ϵ-differential privacy. The algorithm releases z in $\mathcal{Z}$ according to the probability distribution proportional to $\exp(\frac{\epsilon q(\mathbf{x},z)}{2\Delta q})$. Here Δq, is the maximum difference in the score function when two databases differ in one entry.

2.3 Differential Privacy and Statistics

In this section, we briefly describe some recent work that relates differential privacy and statistical inference that are relevant for our problem (e.g., Smith [20], Vu and Slavkovic, [23], Wasserman and Zhou, [24], Wasserman [25]).

2.3.1 Differential Privacy and Maximum Likelihood Estimators

Smith [20] showed that for many parametric models there exists a differentially private point estimator which has the same limiting distribution as the maximum likelihood estimator. In particular, if $X = (X_1, ..., X_n)$ is a simple random sample drawn from $f(x|\theta)$, Smith [20] proposed the sample and aggregate technique combined with the bias corrected MLE to construct an estimator $\mathcal{T}^*$ that satisfies differential privacy such that its mean square error is $\frac{1+o(1)}{nI(\theta)}$; $I(\theta)$ denotes the Fisher information at θ:

$$I(\theta) = E_\theta[(\tfrac{\partial}{\partial\theta} \log f_\theta(X))^2].$$

More precisely, Smith [20] designed a differentially private algorithm that breaks the data $X = (X_1, ..., X_n)$ into k blocks of $\frac{n}{k}$ points each, computes the bias-

corrected MLE on each block, and releases the average of these estimates plus some small Laplace noise [Dwork and Smith [8]].

Theorem 2.3.1 (Smith [20])**.** Under appropriate regularity conditions, there exists a (randomized estimator) $\mathcal{T}^*$ which is asymptotically efficient and ϵ-differentially private where $\lim_{n\to\infty} \epsilon = 0$.

This is a remarkable theorem since the differentially private estimator guarantees privacy and asymptotically performs as well as the maximum likelihood estimator. Thus, leading to valid statistical inference. In this thesis, we evaluate the performance of a differentially private estimator for the mean of the normal random variable on the finite samples.

2.3.2 Differential Privacy and Hypothesis Testing

Privacy-preserving data analysis is also connected to sample size determination in the classical statistical hypothesis testing for clinical trials. Vu and Slavkovic [23] derived sample size adjustment factors for sample size calculation in order to produce the desired power subject to differential privacy for binomial random variables and contingency tables. More specifically, they proposed procedures that provide the lower bounds on the realististic sample size needed to achieve the same statistical power as the true sample size would if there was no infusion of noise for privacy protection for two types of basic hypothesis tests: (1) the test for a single proportion and (2) the Pearson χ^2 test of independence. These procedures can be used by data analysts for planning clinical trials while maintaining privacy and for measuring the loss of statistical efficiency (the reduced sample size) when privacy is deployed [Vu and Slavkovic, [23]]. In this thesis, we extend the work of Vu and Slavkovic, [23] on sample size determination for hypothesis testing with normal random variables.

2.3.3 Data Release Mechanism

The data release mechanism $Q_n(\cdot|X)$ proposed by Wasserman and Zhou, [24] is basically a conditional distribution for a sanitized data set $Z = (Z_1, ..., Z_k)$ given X that satisfies differential privacy. In other words, the data release mechanism is a

random mechanism that takes as an input a data set X and releases a sanitized data set Z according to the distribution $Q_n(\cdot|X)$ that satisfies differential privacy. From statistical perspective, we would like to be able to make any statistical inference based on Z as we would have based on X. In summary,

$$\text{input data set } X = (X_1, ..., X_n) \xrightarrow[\text{sanitize}]{Q_n(Z|X)} \text{output dataset } Z = (Z_1, ..., Z_k).$$

Wasserman [25] and Wasserman and Zhou, [24] used this data release mechanism for nonparametric density estimation. The idea is to compute a density estimate $\hat{f}_X$ based on the initial data set $X = (X_1, ..., X_n)$ and convert it into a differentially private estimator $\hat{f}^*$. Finally, sample $Z_1, ..., .Z_k$ from $\hat{f}^*$ and return $\hat{f}_Z$ as the nonparametric density estimate based on Z.

The release of the sanitized data set Z would inherit the differential privacy properties of the density estimator. Thus, the differentially private sample data could be used as the basis for any number of statistical analyses such as exploratory data analysis, model fitting, etc.

Chapter 3

Hypothesis Testing Under Differential Privacy Framework

3.1 Introduction

In order to improve the quality of research and of health care, researchers and clinicians need rapid access to multiple types of information, including clinical data and clinical literature databases, as well as clinical trials registries, and in some cases sharing of data sets and of results [Cao et al., [2]]. ClinicalTrials.gov has emerged as the largest clinical trial registry in the world. As the number and size of data registries grow, data mining tools enable rapid summarization and derivation of knowledge from the stored clinical information. While these data mining tools offer advantages, challenges associated with privacy invasion remain of concern [Vu and Slavkovic, [23]].

We design private point estimators based on the theory proposed by [Smith [20]] that satisfy differential privacy. The private point estimator satisfies differential privacy if for any pair of data sets that differ in only one component the probability distribution does not change very much. Intuitively, differential privacy ensures that an adversary does not gain knowledge about a specific individual, whether that individual provides actual or fake information to the data set.

The goal of this chapter is to design differentially private point estimators for the normal random variable with a focus on statistical elements relevant to the

design and analysis of clinical trial data; similar to what was proposed in Vu and Slavkovic, [23] for binary random variables. More specifically, we evaluate the private and non private estimators in terms of their bias and asymptotic efficiency. Furthermore, we derive sample size adjustment factors that are needed for sample size calculation and power analysis in hypothesis testing of the mean for normal distribution subject to privacy; this extends the work of Vu and Slavkovic, [23]. Finally, we propose a differentially private hypothesis testing procedure for the mean of a normal random variable such that both statistical power and differential privacy can be achieved.

3.2 Summary of previous work

The relationship between differential privacy and sample size determination was first studied by Vu and Slavkovic, [23] for binomial random variables and contingency tables. In their work, they used the output perturbation algorithm (see Vu and Slavkovic, [23]) to design a differentially private estimator for the binomial proportion. The algorithm adds Laplace random noise with mean 0 and standard deviation $\frac{1}{\epsilon N}$ to the sample proportion and outputs the differentially private point estimator; N here denotes the sample size. A Monte Carlo simulation compared the performance of this new private point estimator with the true maximum likelihood estimator. As expected, the simulations showed that by increasing the sample size N, the private estimator achieved better asymptotic efficiency. Similarly, increasing the value of the privacy level ϵ, that is allowing for more information leakage, they improved asymptotic efficiency of the private estimator. Their algorithm performed well because the parameter space for the binomial proportion is bounded and the diameter Λ of the parameter space is equal to 1.

Furthermore, Vu and Slavkovic [23] showed how to integrate the differential privacy framework with classical statistical hypothesis testing. In particular, they derived rules for sample size adjustment factors K that will enable researchers to determine the required sample size to achieve the same statistical power and significance level in the "private" analysis as they would have obtained in a non-private analysis for two types of basic hypothesis: (1) the test for a single proportion and (2) the Pearson χ^2 test of independence [Vu and Slavkovic, [23]]. The range of the

sample size adjustment factors was obtained for different values of privacy level ϵ and for a fixed finite sample size N. The χ^2 statistic of independence was calculated for a 2×2 contingency table where the Laplace noise was added to the cell probabilities under the saturated model. The simulations compared statistical power $1 - \beta$ for the χ^2 test of independence with confidence level $1 - \alpha$ given the original sample size N and the private sample size $N' = KN$. The results showed that ϵ-differentially private estimates achieved the desired privacy and had greater power because the corrected sample sizes are larger [Vu and Slavkovic, [23]].

3.2.1 Contribution of our work

Our work extends the work of Vu and Slavkovic, [23] to point estimation and hypothesis testing of the mean when the data are normally distributed. We design a differentially private point estimator for estimating the mean of the normal random variable. We provide two different private estimators depending on the noise distribution. In Section 3.3, Algorithm 1 satisfies ϵ-differential privacy with Laplace noise, and in Section 3.8, Algorithm 3 satisfies (ϵ, δ)-differential privacy with Gaussian noise. In Section 3.3.1, we study the effect of the diameter Λ of the parameter space on these private point estimators in terms of asymptotic efficiency. In Chapter 4, we introduce the sensitivity framework to design an algorithm that is not sensitive to the value of Λ. The simulations indicate that this algorithm outperforms the output perturbation algorithm for small samples regardless the value of ϵ and Λ.

In Section 3.4, we extend the method of Vu and Slavkovic, [23] for sample size determination when the data are normally distributed and the diameter of the parameter space Λ is not 1. We provide lower bounds on the "private" sample size needed to achieve the same statistical power as the true sample size, if there was no infusion of noise for statistical power.

In Section 3.5, we propose a new hypothesis testing procedure for the population mean based on the proposed private estimators (from Section 3.3). Based on the simulation studies, this new hypothesis testing procedure indicates that there is no significant Type I error inflation and that there is good statistical power.

3.3 A differentially private point estimator for the mean

Let $X = (X_1, ..., X_N) \sim N(\mu, \sigma^2)$ where σ^2 is known and $\mu \in [-\frac{\Lambda}{2}, \frac{\Lambda}{2}]$ where $\Lambda > 0$. Let $\hat{\mu}(X) = \bar{X}$ be the maximum likelihood estimator of μ. In Algorithm 1, as proposed by Vu and Slavkovic, [23], we add Laplace noise with standard deviation $\frac{GS_{\hat{\mu}}}{\epsilon}$ to the MLE; $GS_{\hat{\mu}}$ denotes the ℓ_1-global sensitivity of the MLE and ϵ the privacy level, in order to obtain a differentially private estimator of μ. Recall, smaller ϵ, means stronger privacy, but more noise addition. Here, $GS_{\hat{\mu}} = \frac{\Lambda}{N}$ and hence, a differentially private maximum likelihood estimator of μ is $\mu(X, R) = \bar{X} + R$, where $R \sim \text{Lap}(\frac{\Lambda}{\epsilon N})$.

Algorithm 1: An ϵ-Differentially Private Estimator [Vu and Slavkovic, [23]]

Input A data set $\mathbf{x} \in \mathbb{R}^N$, $\epsilon > 0$. Assume Λ is the range of $T(\mathbf{x})$ or the diameter of the parameter space.

1. Obtain the sufficient statistic $T(\mathbf{x})$ for μ.
2. Draw a random observation R from a Laplace distribution with mean 0 and standard deviation $\Lambda/\epsilon N$.
3. Return $\mu(\mathbf{x}, R) = \bar{x} + R$.

Thus, the differentially private estimator is $\mu(X, R) = \bar{X} + \text{Lap}(\frac{\Lambda}{\epsilon N})$.

Proposition 3.3.1 (Vu and Slavkovic, [23])**.** Algorithm 1 satisfies ϵ-differential privacy.

Proof. Let $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$ that differ only in one position, $y \in \mathbb{R}$ and $\epsilon > 0$. Then the random variable $\mu(\mathbf{x}, R) = \bar{x} + R$ is a Laplace random variable with density $h(y) = \frac{1}{2 \cdot \Lambda/N\epsilon} e^{\frac{-|y-\bar{x}|}{\Lambda/N\epsilon}}$. Thus,

$$\begin{aligned}
\frac{h_{\mu(\mathbf{x},R)}(y)}{h_{\mu(\mathbf{x}',R)}(y)} &= \frac{e^{-\frac{N\epsilon|y-\bar{x}|}{\Lambda}}}{e^{\frac{-N\epsilon|y-\bar{x'}|}{\Lambda}}} \\
&= e^{-\frac{N\epsilon|y-\bar{x}|}{\Lambda}} e^{\frac{N\epsilon|y-\bar{x'}|}{\Lambda}} \\
&= e^{-\frac{N\epsilon}{\Lambda}(|y-\bar{x}|-|y-\bar{x'}|)} \quad \text{since } |y-y'| \geq ||y|-|y'||
\end{aligned}$$

$$\leq e^{\frac{N\epsilon}{\Lambda}|\bar{x'}-\bar{x}|}$$
$$\leq e^{\frac{N\epsilon}{\Lambda}GS_{\hat{\theta}}}$$
$$\leq e^{\epsilon}$$

□

Alternatively, one can construct an ϵ-differentially private estimator using the sample and aggregate method [Smith [20]]. The algorithm is:

Algorithm 2: Sample and Aggregate Algorithm [Smith, [20]]

Input $\mathbf{x} = (x_1, x_2, ..., x_N) \in \mathbb{R}^N$, $\epsilon > 0$ and $k \in \mathbb{N}$.

1. Divide the database x into k disjoint smaller databases with t points per database, $t = \frac{N}{k}$. That is $x_i^* = (x_{(i-1)t}, ..., x_{it})$ for $i = 1, ..., k$.
2. Compute the MLE of μ for each block x_i^*, that is $z_i = \frac{\sum_{j=1}^{t} x_{ij}}{t}$.
3. Compute $\bar{z} = \frac{\sum_{i=1}^{k} z_i}{k}$.
4. Return $\mu(x, R) = \bar{z} + R$ where $R \sim \text{Lap}(\frac{GS_{\bar{z}}}{\epsilon})$ and $GS_{\bar{z}} = \frac{\Lambda}{kt\epsilon}$ is the global sensitivity of $\bar{z}$.

Thus, $\mu(X, R) = \bar{X} + R$ where $R \sim \text{Lap}(\frac{\Lambda}{kt\epsilon})$ with $kt = N$.

Both Algorithms 1 and 2 add the same amount of noise to the sample mean. Thus, for the sample mean of the normal distribution the two algorithms are equivalent. This implies that it is sufficient to use the Laplace Mechanism technique to get an ϵ-differentially private estimator.

3.3.1 Efficiency guarantees

To study the efficiency of an estimator $T(X)$ we use standardized mean square error (MSE), which is the mean square error of the estimator divided by the variance of the MLE. The mean square error of an estimator is defined as:

$$J_{T(X)}(\mu) = E[(T(X) - \mu)^2] \ .$$

If $T(X)$ is unbiased, its mean square error is equal to its variance, $J_{T(X)}(\mu) = \mathrm{Var}(T(X))$. Hence, the standardized mean square error is just the ratio of the variances and is given by:

$$\frac{J_{T(X)}(\mu)}{\mathrm{Var}(\hat{\mu})} \,. \tag{3.1}$$

Therefore, if the estimator $T(X)$ is asymptotically efficient, the standardized mean square error approaches to 1 as the sample size increases.

The noise added to the sample mean to obtain the ϵ-differentially private estimator $\mu(X, R) = \bar{X} + R$ does not contribute to the expectation but it does contribute to the variance. More specifically,

$$E(\mu(X,R)) = \mu, \qquad \mathrm{Var}(\mu(X,R)) = \tfrac{\sigma^2}{N} + \tfrac{2\Lambda^2}{N^2\epsilon^2} \,.$$

The next proposition shows that if we can find Λ in terms of ϵ and N, then the private statistic is a consistent estimator of μ.

Proposition 3.3.2. $\mu(X, R) \xrightarrow{P} \mu$ as long as $\Lambda = o(\epsilon N)$ and $\sigma^2 < \infty$.

Proof. The private statistic is an unbiased estimator of μ, so $\mu(X, R)$ is a consistent estimator of μ as long as $\mathrm{Var}(\mu(X, R)) \to 0$. Let $\delta > 0$, then by Chebyshev's inequality we have:

$$\begin{aligned} P(|\mu(X,R) - \mu| > \delta) &\leq \frac{\mathrm{Var}(\mu(X,R))}{\delta^2} \\ &= \frac{\sigma^2}{\delta^2 N} + \frac{2\Lambda^2}{\delta^2\epsilon^2 N^2} \,. \end{aligned}$$

If $\sigma^2 < \infty$ then $\frac{\sigma^2}{\delta^2 N} \to 0$ as $N \to \infty$. Thus, as long as $\Lambda = o(\epsilon N)$ the private estimator is a consistent estimator of μ.

□

Proposition 3.3.2 also implies that the private statistic converges in probability to the sample mean $\mu(X, R) \xrightarrow{P} \bar{X}$, provided $\Lambda = o(\epsilon N)$ as $N \to \infty$. That is, the limiting distribution of the private estimator is the same as the limiting

distribution of the maximum likelihood estimator. Smith, [20] proved that for any parametric model, under some regularity conditions, there exists a (randomized) estimator that is asymptotically efficient and differentially private. In the following proposition, we provide a particular version of the proof of [Smith, [20]] which is sufficient to show that our estimator $\mu(X, R)$ is asymptotically efficient.

Proposition 3.3.3. $\frac{\mu(X,R)-\mu}{\sigma/\sqrt{N}} \xrightarrow{D} N(0,1)$ as long as $\sigma^2 < \infty$ and $\Lambda = o(\epsilon\sqrt{N})$.

Proof. The mean square error of the private statistic is $J_{\mu(X,R)}(\mu) = \frac{1}{N}(\frac{1}{1/\sigma^2} + \frac{2\Lambda^2}{\epsilon^2 N})$. Hence, if $\Lambda = o(\epsilon\sqrt{N})$ as $N \to \infty$ we have:

$$J_{\mu(X,R)}(\mu) = \frac{1}{N}(\frac{1}{1/\sigma^2} + o(1))$$

□

This provides evidence that differential privacy can be consistent with statistically valid inference as initially shown by [Smith [20]].

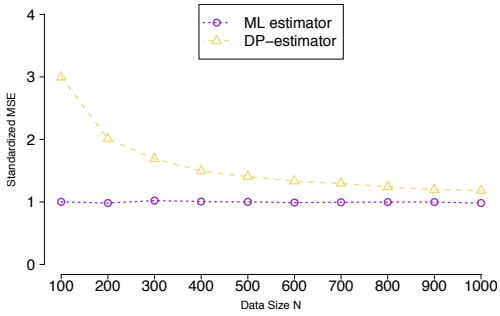
3.3.2 Simulation study I on statistical efficiency

The first simulation studies the efficiency and the performance of the private statistic for a finite sample. The simulation studies the performance of the private estimator using the standardized mean squared error for finite sample sizes as we vary ϵ and Λ. In the simulation, we draw $M = 10,000$ samples of N data points from a normal distribution with parameters $\mu = 0$ and $\sigma = 1$, then add Laplace noise $\text{Lap}(\frac{\Lambda}{N\epsilon})$ to the MLE to obtain the private statistic $\mu(X, R)$. We approximate its MSE by:

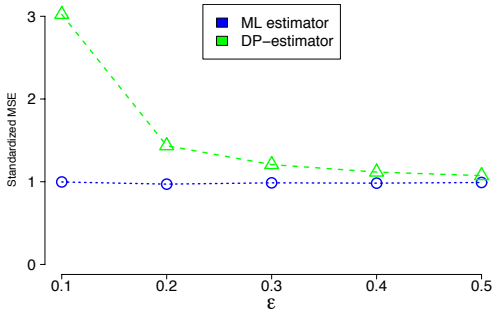
$$J_{\mu(X,R)}(\mu) = \frac{1}{M}\sum_{i=1}^{M}(\hat{\mu}(X, R) - \mu)^2 \ .$$

We consider four cases and present the results in Figure 3.1. First, we plot the standardized MSE of the private estimator $\mu(X, R)$ against the varying sample size N (from 100 to 1,000) for fixed values $\epsilon = 0.1$ and $\Lambda = 1$ (see Figure 3.1a). Figure 3.1b plots the standardized MSE against varying levels of ϵ from 0.05 to 0.5 for a fixed data size $N = 100$ and $\Lambda = 1$. The plot in Figure 3.1c, shows the behaviour of the standardized MSE against varying levels of Λ (from 1 to 21) for $\epsilon = 0.1$ and $N = 100$. Finally, in Figure 3.1d, we plot the standardized MSE against Λ and

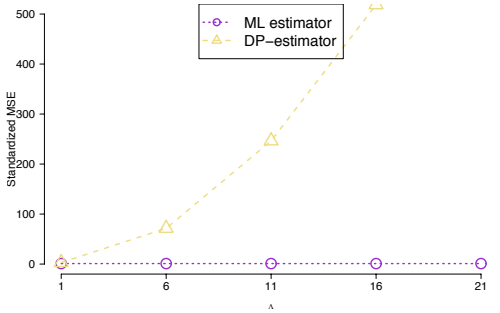
data size N for a fixed $\epsilon = 0.1$.



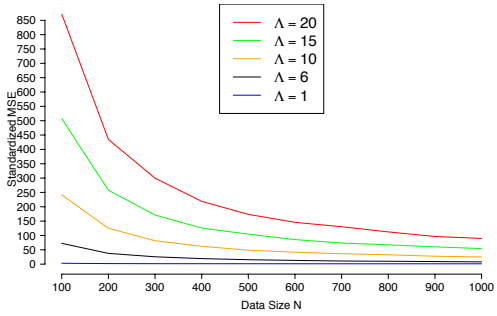
(a) Controlling for the sample size N with $\epsilon = 0.1$ and $\Lambda = 1$



(b) Controlling for ϵ with $\Lambda = 1$ and $N = 100$



(c) Controlling for Λ with $\epsilon = 0.1$ and $N = 100$



(d) Controlling for N and Λ with $\epsilon = 0.1$

Figure 3.1: MSE of MLE versus DP-estimator with $\mu = 0, \sigma^2 = 1$

Figure 3.1a confirms the asymptotic efficiency of the private estimator for small Λ and large N. As expected, when $\Lambda = 1$ and $\epsilon = 0.1$, as the sample size increases the standardized MSE of the private estimator approaches 1 since the private estimator is asymptotically efficient. From Figure 3.1b, we see that when we increase the value of the privacy parameter ϵ (i.e., less privacy guarantee, less noise), we achieve better asymptotic efficiency. On the other hand, when Λ increases as seen in Figure 3.1c, the standardized MSE of the private estimator increases monotonically. This shows that the performance of the private estimator degrades as

the diameter Λ increases. This is due to the increasing global sensitivity of the estimator as Λ increases, making the variance of the noise too high to guarantee the efficiency of the private estimator. However, Figure 3.1d shows that when we increase both the value of Λ and the sample size N the asymptotic efficiency of the private estimator is achieved indicating that as long as $\frac{\Lambda}{N\epsilon}$ is sufficiently small the differentially private estimator, $\mu(X, R)$, behaves much like the MLE. In fact, this confirms the theoretical results from Proposition 3.3.1 and 3.3.2 that the private statistic $\mu(X, R)$ and the MLE are asymptotically equivalent provided $\frac{\Lambda}{\epsilon N} = o(1)$.

3.4 Sample size determination under differential privacy framework

In the previous section, we designed an output perturbation algorithm (Algorithm 1) for point estimation with differential privacy guarantees. We compared the efficiency of the private estimator $\mu(X, R)$ with the MLE as we vary the sample size N, the privacy parameter ϵ and the diameter Λ. Propositions 3.3.1 and 3.3.2 and the simulation results show that the private estimator achieves asymptotic efficiency when $\frac{\Lambda}{N\epsilon}$ is sufficiently small. In this section, we use the differentially private estimator $\mu(X, R)$ in classical hypothesis testing problem and sample size determination. Specifically, we provide a framework for privacy preserving hypothesis testing and sample size determination for normal random variables. Our analysis is based on the work of Vu and Slavkovic, [23] which specifically studied the binomial distribution and contingency tables under such a setting.

Sample size determination is often an essential component in experimental design. Researchers are required to determine the finite sample size needed to detect important significant differences in the parameter of interest. In this section, we derive sample size adjustment factors than can be used for sample size evaluation under the differential privacy framework. When conducting a statistical test, two types of error must be considered: Type I and Type II errors with probabilities α and β respectively. The Type I error probability is the probability of rejecting the null hypothesis when it is true. The power of a hypothesis test calculated as $1 - \beta$ is the probability of rejecting the null hypothesis when it is false where β, the Type

II error, is the probability of failing to reject the null hypothesis when it is false [Casella and Berger, [3]]. All sample size calculations for planning clinical trials calculate a sample size by choosing a fixed Type I error rate (α) and specifying the desired power ($1-\beta$). In the following, we denote with N the original or true sample size calculated without accounting for privacy, and with N' the differentially private sample size. All expressions correspond to a specified null hypothesis H_0 and one test statistic, and depend on the chosen Type I error α and power $1-\beta$. In this section, we formulate the sample size determination problem in the context of differential privacy for testing the mean of the normal distribution. More particularly, the procedures provide the lower bounds of the private sample size needed to achieve the same statistical power as the true N similar to what Vu and Slavkovic, [23] showed for the proportion of binomial distribution.

3.4.1 Hypothesis test about μ with known variance

Suppose we are interested in testing the following hypothesis:

$$H_0 : \mu = \mu_0 \qquad H_1 : \mu = \mu_1 = \mu_0 + \gamma$$

where $\gamma = \mu_1 - \mu_0 > 0$ is the noncentrality parameter and μ_i, $i = 0, 1$ is the mean of a normal random variable under the null and alternative hypothesis, respectively.

Without privacy, the usual test statistic is $t = \frac{\bar{X}-\mu_0}{\sigma/\sqrt{N}} \sim N(0,1)$. The test of size α rejects H_0 if $\bar{X} > \mu_0 + \frac{\sigma}{\sqrt{N}} z_{1-\alpha}$. The required sample size in order to achieve Type I error probability α and power $1-\beta$ for the one - sided hypothesis is given by Van Belle [22], and Casella and Berger, [3] as:

$$N = \frac{(z_{1-\alpha}+z_{1-\beta})^2\sigma^2}{\delta^2} \ .$$

The private estimator obtained by Algorithm 1 is $\mu(X,R) = \bar{X} + \text{Lap}(\frac{\Lambda}{N\epsilon})$. Related to the work of Vu and Slavkovic, [23] we considered two cases for calculating the distribution of the private test statistic under H_0. The first case is based on the exact distribution of the private estimator, as the convolution of two independent random variables, and the second case is based on approximating the Laplace distribution by the normal distribution. For each case we show how to

determine the sample size required to achieve $1-\beta$ power and Type I error α for the one-sided test based on the proposed private statistic.

Normal-Laplace distribution. In the first case, the differentially private estimator $\mu(X,R) = \bar{X} + \text{Lap}(\frac{\Lambda}{N'\epsilon})$ has the Normal-Laplace distribution, $\mu(X,R) \sim \text{NL}(\mu, \frac{\sigma^2}{N'}, \frac{\epsilon N'}{\Lambda}, \frac{\epsilon N'}{\Lambda})$ since it is the convolution of a normal and a Laplace random variable [Reed [19]]. We would like to find the required sample size N', when the private statistic has the exact Normal-Laplace distribution. We obtain the exact value of $K = N'/N$ using a numerical method in R [18, 18] solving the following equation for N':

$$F_{H_0}^{-1}(1-\alpha/2) = F_{H_1}^{-1}(1-\beta) \tag{3.2}$$

where F_{H_0} is the CDF of $NL(\mu_0, \frac{\sigma^2}{N'}, \frac{\epsilon\Lambda}{N'}, \frac{\epsilon\Lambda}{N'})$ under H_0 and F_{H_1} is the CDF of $NL(\mu_0+\gamma, \frac{\sigma^2}{N'}, \frac{\epsilon\Lambda}{N'}, \frac{\epsilon\Lambda}{N'})$ under H_1. The R code is given in Appendix A.

Normal-Normal distribution. In this case, the Laplace noise distribution is approximated by a normal distribution which has mean 0 and variance $\frac{2\Lambda^2}{(\epsilon N')^2}$. Thus, the sampling distribution of $\mu(X,R)$, under H_0, is approximated by $N(\mu_0, \frac{\sigma^2}{N'} + \frac{2\Lambda^2}{(\epsilon N')^2})$, and under H_1 is approximated by $N(\mu_0+\gamma, \frac{\sigma^2}{N'} + \frac{2\Lambda^2}{(\epsilon N')^2})$. We call the approximated normal distribution the Normal-Normal distribution. This approach was first used by Vu and Slavkovic, [23] for the hypothesis testing of the binomial proportion. By specifying the Type I error α and power $1-\beta$ we would like to find the required sample size N' to account for the added noise when the private statistic has the Normal-Normal distribution.

The sample size N' is obtained by solving the following equations [Van Belle [22]]:

$$P(\mu(X,R) > c_\alpha | H_0) = \alpha$$
$$P(\mu(X,R) < c_\beta | H_1) = \beta$$

Letting $c_\alpha(N') = c_\beta(N')$, leads to the following equation:

$$\sqrt{\frac{\sigma^2}{N'} + \frac{2\Lambda^2}{\epsilon^2 N'^2}} z_{1-\alpha} + \mu_0 = \mu_0 + \gamma - z_{1-\beta}\sqrt{\frac{\sigma^2}{N'} + \frac{2\Lambda^2}{\epsilon^2 N'^2}}$$

and after some computation we obtain the quadratic equation in terms of N':

$$N'^2 - NN'^2 - 2\frac{(z_{1-\alpha} + z_{1-\beta})^2\Lambda^2}{\gamma^2\epsilon^2} = 0 \ .$$

Solving for N', the privacy-preserving sample size N' is given by:

$$N' = N(\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{8\gamma^2\Lambda^2}{\epsilon^2(z_{1-\alpha} + z_{1-\beta})^2\sigma^4}}) \tag{3.3}$$

and the sample size adjustment factor K is

$$K = \frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{8\gamma^2\Lambda^2}{\epsilon^2(z_{1-\alpha} + z_{1-\beta})^2\sigma^4}} \ . \tag{3.4}$$

The above formula (3.4) is used to determine the sample size N' under differential privacy by taking $N' = KN$ and is similar to the formula of the sample size adjustment factor for the binomial proportion of Vu and Slavkovic, [23]. In their formula, the diameter range Λ is equal to one since the parameter space of the binomial proportion is the interval $[0, 1]$.

For each of the two settings described above, given a fixed value of α and β γ we compute the value of K as we vary Λ and ϵ. Moreover, we examine the effect of adjusting the sample size N on Type I and Type II error rates using both the Normal-Laplace and Normal-Normal adjustment factors K, for testing the null hypothesis $H_0 : \mu = \mu_0$. The results given in Section 3.6, show that there is no significant Type I and Type II error inflation for either of the sample size adjustments for different values of ϵ when $\Lambda = 1$. However, when Λ increases and when $\epsilon = 0.1$ both cases displayed inflated Type I error, even when the proposed sample size adjustment is employed. Although the inflation is much smaller than when no adjustment is made at all.

3.5 A new hypothesis testing procedure

In this section, we propose a new hypothesis testing procedure for the mean of the normal distribution based on the differentially private estimator $\mu(X, R) = \bar{X} + \text{Lap}(\frac{\Lambda}{\epsilon N})$.

Instead of adjusting the original sample size to account for the added noise, we use the two distributions from the previous section, the Normal-Normal and the Normal-Laplace distributions, as the null distribution of the private statistic for testing the null hypothesis $H_0 : \mu = \mu_0$ against the alternative $H_1 : \mu > \mu_0$. Our simulations in Section 3.7 show an improvement in comparison to our results in Section 3.6 with respect to the Type I error performance of the private statistic as Λ increases.

Under privacy, the estimator of μ is $\mu(X, R) = \bar{X}+\text{Lap}(\frac{\Lambda}{\epsilon N})$ and we are interested in testing the null hypothesis $H_0 : \mu = \mu_0$ against the alternative $H_1 : \mu > \mu_0$. The test rejects H_0 when $\mu(X, R) > k$ where k is the critical value under the sampling distributon of the private test statistic. We consider two different tests according to the null sampling distribution:

Normal-Normal test: This test is based on the Normal-Normal null sampling distribution of $\mu(X, R) \sim N(\mu_0, \frac{\sigma^2}{N} + \frac{2\Lambda^2}{\epsilon^2 N^2})$. The test statistic is

$$t(X, R) = \frac{\mu(X,R)-\mu_0}{\sqrt{\frac{\sigma^2}{N}+\frac{2\Lambda^2}{\epsilon^2 N^2}}} \sim N(0, 1) \ .$$

Therefore, the test of size α rejects $H_0 : \mu = \mu_0$ when:

$$\mu(X, R) > \mu_0 + z_{1-\alpha}\sqrt{\frac{\sigma^2}{N} + \frac{2\Lambda^2}{\epsilon^2 N^2}}. \tag{3.5}$$

Normal-Laplace test: Under H_0, the private estimator $\mu(X, R)$ has the exact Normal-Laplace distribution $\text{NL}(\mu_0, \frac{\sigma^2}{N}, \frac{\epsilon\Lambda}{N}, \frac{\epsilon\Lambda}{N})$. The critical value k, is obtained by:

$$P(t(X, R) > k|H_0) = \alpha \tag{3.6}$$

where $t(X, R)$ is the private test statistic, k is the $1 - \alpha$ quantile of the Normal-Laplace distribution and is obtained computationally in R.

Obviously the new critical points in (3.5) and (3.6) depend on the privacy parameter ϵ and on the diameter of the parameter space Λ. A simulation study that compares the two versions of the hypothesis testing procedure (3.5) and (3.6) with the test that rejects H_0 when $\mu(X, R) > \mu_0 + z_{1-\alpha}\sqrt{\frac{\sigma^2}{N}}$ is provided in Section 3.7.

3.6 Simulation study II

This section presents results from two simulations studies. The first study, computes the sample size adjustment factors K for the Normal-Laplace (NL) and Normal-Normal (NN) case obtained by (3.2) and (3.3) respectively. The second study, investigates the effect of the sample size adjustment on the Type I error and Type II error rates. For each simulation, $10,000$ samples of size N were generated from the normal distribution with parameters $\mu_0 = 0$ and $\sigma^2 = 1$. We used a significance level of $\alpha = 0.05$, Type II $\beta = 0.1$ and $\beta = 0.4$ and noncentrality parameter $\gamma = 0.1$.

3.6.1 Simulation results on sample size adjustment factors K

Tables 3.1 to 3.4 and Figures 3.2 and 3.3 compare the sample size adjustment factors K for the Normal-Normal distribution (formula (3.5)) and the Normal-Laplace distribution at two different power levels of 0.6 and 0.9; we vary the privacy level ϵ from 0.1 to 0.5 and the length of the interval Λ from 1 to 10. For example, when we consider the Normal-Normal distribution, we can see from Table 3.2 that if we want privacy at the level of $\epsilon = 0.1$ we need a sample size which is 1.195 times the true sample size N to obtain a power of 0.9. Since the Normal-Laplace distribution has heavier tails than the Normal-Normal distribution, the approximated formula (3.4) underestimates the values of K, compared to the exact values of K. However, from the Tables 3.1 and 3.2, we can see that the differences are small; especially for increasing values of ϵ. Also, when $\Lambda = 1$, as the privacy parameter increases the sample size adjustment factor decreases to 1 for both power levels; thus there is no difference between the non-private sample size N, and the private sample size N'. Another observation is that since the non-private sample size increases as the required power increases, the sample size adjustment factor K decreases.

Figures 3.2a and 3.3a show that the K factor increases linearly with respect to Λ for both cases (Normal-Normal and Normal-Laplace) when the privacy parameter is fixed at $\epsilon = 0.1$. Thus, as Λ increases we need a larger sample size to account for the infusion of noise, as seen from Tables 3.3 and 3.4.

ϵ	K factor Normal-Normal	K factor Normal-Laplace
0.1	1.397	1.382
0.2	1.124	1.122
0.3	1.058	1.058
0.4	1.034	1.033
0.5	1.022	1.022

Table 3.1: Sample size adjustment factor K when $\Lambda = 1$, $\beta = 0.4$ and $N = 361$.

ϵ	K factor Normal-Normal	K factor Normal-Laplace
0.1	1.195	1.190
0.2	1.055	1.055
0.3	1.025	1.025
0.4	1.014	1.014
0.5	1.009	1.009

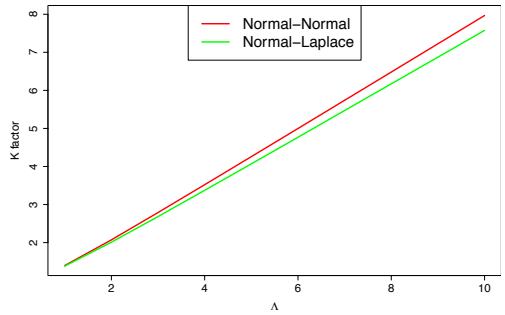
Table 3.2: Sample size adjustment K when $\Lambda = 1$, $\beta = 0.1$ and $N = 857$.

Λ	K factor Normal-Normal	K factor Normal-Laplace
1	1.195	1.190
2	1.588	1.557
3	2.034	1.968
4	2.498	2.398
5	2.968	2.834
6	3.442	3.277
7	3.920	3.722
8	4.398	4.170
9	4.878	4.620
10	5.358	5.07

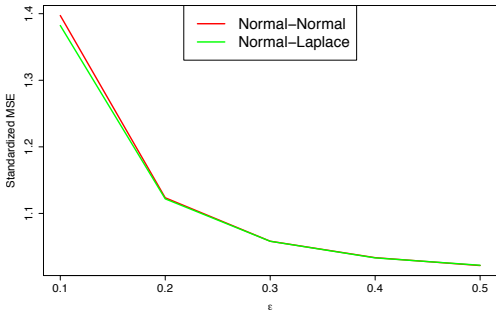
Table 3.3: Sample size adjustment factor K when $\epsilon = 0.1$, $\beta = 0.1$ and $N = 857$.

Λ	K factor Normal-Normal	K factor Normal-Laplace
1	1.397	1.382
2	2.072	2.011
3	2.790	2.684
4	3.522	3.374
5	4.259	4.069
6	4.998	4.767
7	5.739	5.471
8	6.481	6.172
9	7.224	6.873
10	7.967	7.576

Table 3.4: Sample size adjustment factor K when $\epsilon = 0.1$, $\beta = 0.4$ and $N = 361$.

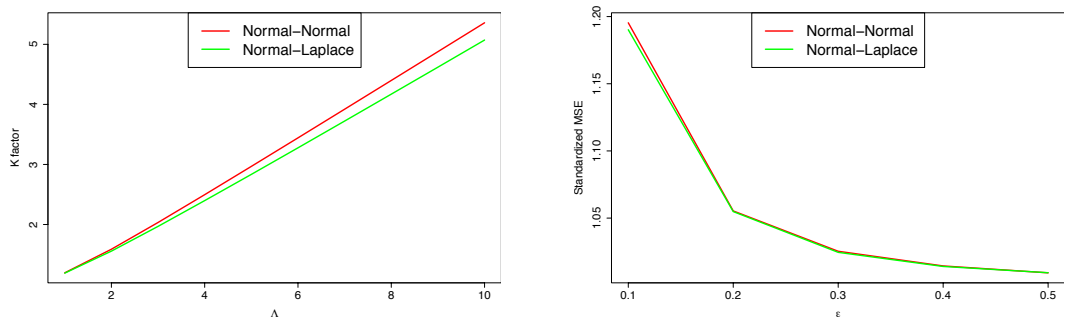


(a) Sample size adjustment factor K with $\epsilon = 0.1$ controlling for the effect of Λ.



(b) Sample size adjustment factor K with $\Lambda = 1$ controlling for the effect of ϵ.

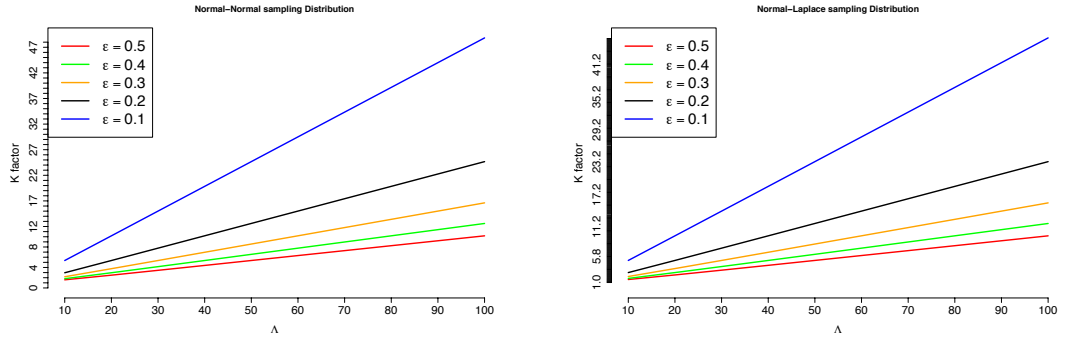
Figure 3.2: Sample size adjustment factor K with $\mu_0 = 0$, $\sigma^2 = 1$ for $\alpha = 0.05$ and $\beta = 0.4$.



(a) Sample size adjustment factor K with $\epsilon = 0.1$ controlling for the effect of Λ.

(b) Sample size adjustment factor K with $\Lambda = 1$ controlling for the effect of ϵ.

Figure 3.3: Sample size adjustment factor K with $\mu_0 = 0$, $\sigma^2 = 1$ for $\alpha = 0.05$ and $\beta = 0.1$.



(a) Sample size adjustment factor K for the Normal-Normal distribution.

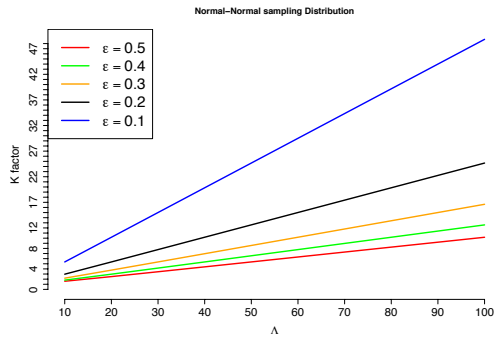
(b) Sample size adjustment factor for the Normal-Laplace distribution.

Figure 3.4: Sample size adjustment factor K controlling the effect of ϵ and Λ with $\mu_0 = 0$, $\sigma^2 = 1$ for $\alpha = 0.05$ and $\beta = 0.1$.

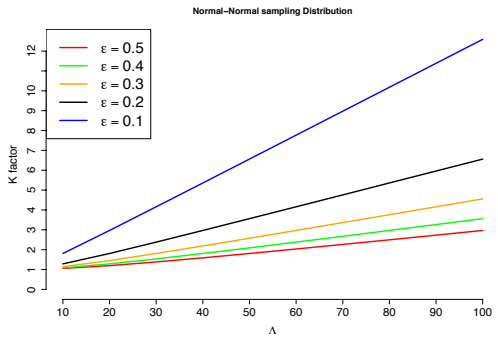
Figures 3.4a- 3.4b plot the range of the factor K derived by formula (3.4) with respect to ϵ and Λ. It is clear that when $\epsilon = 0.1$ and $\Lambda = 20$ we need a sample size about 10 times the true sample size $N = 857$ to achieve a test of size 0.05 and statistical power of 0.9 for both cases.

Figures 3.5a- 3.5d plot the range of the sample size adjustment factor K derived in (3.4) (Normal-Normal) for statistical power of 0.9 and sample size $N = 857$, for

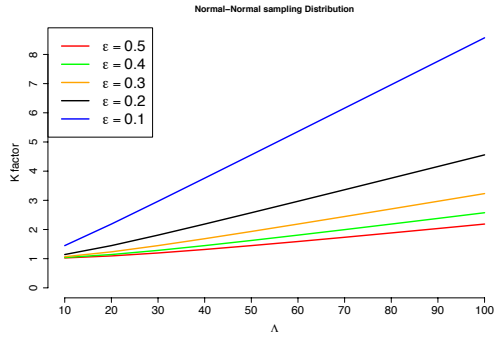
different values of the variance σ^2 as we vary ϵ from 0.1 to 0.5 and Λ from 10 to 100. We observe that with an increase in variance the range of the values of K decreases since the non-private sample size N increases.



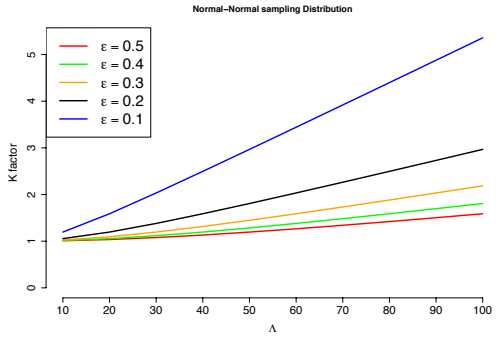
(a) Sample size adjustment K with $\mu_0 = 0$, $\sigma = 1$.



(b) Sample size adjustment factor K with $\mu_0 = 0$, $\sigma = 4$.



(c) Sample size adjustment factor K with $\mu_0 = 0$, $\sigma = 6$.



(d) Sample size adjustment factor K with $\mu_0 = 0$, $\sigma = 10$.

Figure 3.5: Sample size adjustment factor K (formula (3.4)) for $\alpha = 0.05$, $\beta = 0.1$ controlling for the effect of ϵ and Λ.

3.6.2 Simulation results on impact of K on the error rates

Tables 3.5, 3.6, 3.7 and 3.8 show the effect of the sample size adjustment for the Normal-Normal (3.4) and Normal-Laplace case (3.2) on the Type I error and Type II error rates of the hypothesis test that rejects H_0 when $\mu(X, R) > \mu_0 + z_{1-\alpha}\sqrt{\frac{\sigma^2}{N}}$. Note that these tables were based on a different simulated samples with the given

parameters. It is clear from Tables 3.5 and 3.6 that the significance level $\alpha = 0.05$ and the Type II error probability $\beta = 0.1$ are achieved when $\Lambda = 1$ and as ϵ increases from 0.1 to 0.5 since the null distribution of the private statistic reduces to that of the MLE (Proposition 3.3.3). It is clearly though, that there is a greater inflation of the Type I error and loss of power when $\epsilon = 0.1$ for all cases (Tables 3.5 and 3.6, first row). We also observe that even when there is no sample size adjustment (no correction, $K = 1$) the approximations of the Type I and Type II error rates are good, especially when ϵ increases. Table 3.7 shows that for larger values of Λ and when $\epsilon = 0.1$ the Type I error probability is larger than the significance level $\alpha = 0.05$ for both correction factors. This Type I error inflation is expected since the private statistic is asymptotically normal when $\frac{\Lambda}{\epsilon N}$ is sufficiently small. On the other hand, we can see from the Table 3.8 that since the private sample sizes become larger there is an increase in statistical power. However, this improvement in power is meaningful only if the Type I error rate is less or equal than the nominal significance level $\alpha = 0.05$ [Chen [4], Zimmerman et al., [26]]. With no correction and as Λ increases, we lose the asymptotic normality of the private statistic, resulting in inflated Type I and Type II error rates.

Figure 3.6 shows the inflation of Type I error probability when rejecting with the non-private classical critical value $\mu_0 + z_{1-\alpha}\sqrt{\frac{\sigma^2}{N}}$. This is because the larger values of Λ lead to the null sampling distribution of the private statistic with heavier tails.

ϵ	No correction ($K = 1$)	K Normal-Normal	K Normal-Laplace
0.1	0.0705	0.0656	0.062
0.2	0.054	0.0567	0.0538
0.3	0.0534	0.0478	0.052
0.4	0.0499	0.0493	0.052
0.5	0.0538	0.0507	0.0531

Table 3.5: Type I error when $\Lambda = 1$ for $\alpha = 0.05$ and $\beta = 0.1, N = 857$.

ϵ	No correction ($K = 1$)	K Normal-Normal	K Normal-Laplace
0.1	0.127	0.0785	0.0784
0.2	0.102	0.0903	0.093
0.3	0.101	0.0921	0.0979
0.4	0.105	0.0948	0.0942
0.5	1.021	0.0942	0.0948

Table 3.6: Type II error when $\Lambda = 1$ for $\alpha = 0.05$ and $\beta = 0.1$, N=857.

Λ	No correction ($K = 1$)	K Normal-Normal	K Normal-Laplace
1	0.0734	0.064	0.059
2	0.112	0.098	0.091
3	0.151	0.113	0.124
4	0.186	0.134	0.131
5	0.221	0.149	0.149
6	0.249	0.166	0.163
7	0.272	0.175	0.182
8	0.293	0.189	0.185
9	0.303	0.196	0.200
10	0.322	0.206	0.205

Table 3.7: Type I error when $\epsilon = 0.1$ for $\alpha = 0.05$ and $\beta = 0.1, N = 857$.

Λ	No correction ($K = 1$)	K Normal-Normal	K Normal-Laplace
1	0.123	0.0811	0.0795
2	0.165	0.0511	0.0565
3	0.212	0.0388	0.0438
4	0.238	0.0339	0.0382
5	0.267	0.0288	0.031
6	0.292	0.0245	0.0265
7	0.320	0.0235	0.0265
8	0.331	0.0217	0.0256
9	0.348	0.0215	0.0273
10	0.348	0.0199	0.023

Table 3.8: Type II error when $\epsilon = 1$ for $\alpha = 0.05$ and $\beta = 0.1, N = 857$.

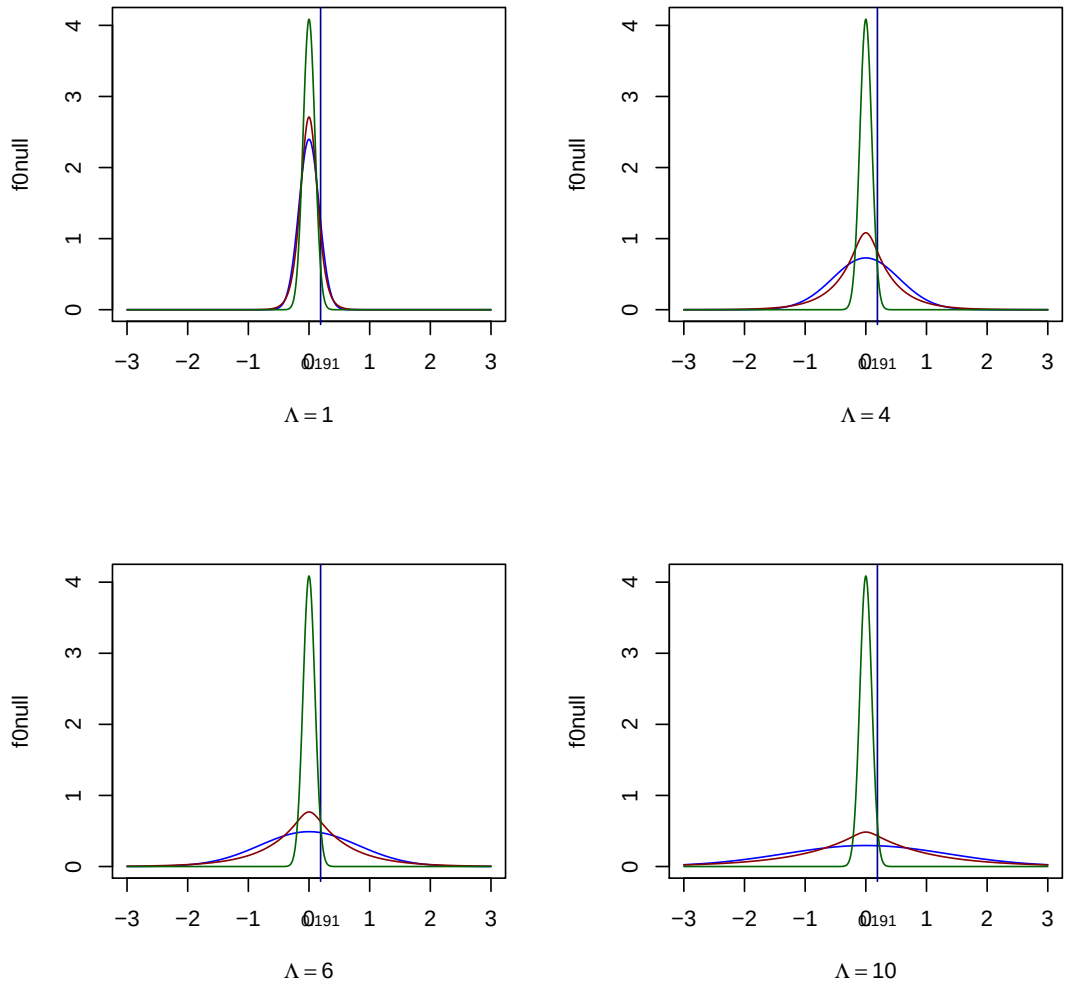


Figure 3.6: Effect of Λ on Type I error probability when $\mu_0 = 0$, $\sigma = 1$, $\alpha = 0.05$, $\epsilon = 0.1$, $N = 105$ rejecting with the non-critical value for Normal-Laplace (red) and Normal-Normal density (blue).

3.7 Simulation study III on new hypothesis tests

The next simulations compares the Type I and Type II error performance of the Normal-Normal test (3.5), and the Normal-Laplace test (3.6) with the test the rejects H_0 when $\mu(X, R) > \mu_0 + z_{1-\alpha}\sqrt{\frac{\sigma^2}{N}}$. In the simulation, we generated 10,000 samples of size $N = 857$ from the normal distribution with $\mu_0 = 0$ and $\sigma^2 = 1$.

For each simulation, we estimate the Type I and Type II error rates for different values of ϵ that vary from 0.1 to 0.5 and for different values of Λ from 1 to 10. We consider a significance level $\alpha = 0.05$, Type II error $\beta = 0.1$ with noncentrality parameter $\gamma = 0.1$. Note again that the Type I and Type II error rates are based on a different simulated samples with the given parameters.

The Type I error and Type II error probabilities of the two versions of the new hypothesis testing procedure with critical values (3.5) for Normal-Normal test and (3.6) for Normal-Laplace test, and of the test with the non-private critical value $\mu_0 + \frac{\sigma}{\sqrt{N}} z_{1-\alpha}$ (Normal test) are given in the Tables 3.9, 3.10, 3.11 and 3.12.

From Table 3.9, we can see that when $\Lambda = 1$, the Type I error rates of the Normal-Normal test and the Normal-Laplace test remain near the $\alpha = 0.05$ significance level even when $\epsilon = 0.1$ while the Type I error of the Normal test is slightly inflated when $\epsilon = 0.1$. Moreover, Table 3.10 shows that when $\Lambda = 1$ we have adequate approximations of the Type II error probability for the three testing procedures especially when ϵ increases from 0.1 to 05. Although the Normal test shows a slightly better power than the other two and this is likely due to the increasing variance of the new sampling distributions.

Tables 3.11 and 3.12 are the Type I and Type II error rates when ϵ is fixed at 0.1 and Λ increases from 1 to 10. Table 3.11 (first column) indicates that the Normal test gives inflated Type I error rates. This inflation of the Type I error is due to the fact that the private statistic no longer follows an asymptotically normal distribution when Λ is large relative to the sample size N. On the other hand, from Table 3.11 we can see that for the Normal-Normal test and the Normal-Laplace test, the Type I error rates tend to be near the significance level $\alpha = 0.05$ even for large values of Λ. In contrast, as shown in Table 3.12 when the privacy level is fixed at 0.1 large values of Λ lead to increased Type II error rates and small power for the three hypothesis testing procedures. But we know that increasing the sample size should improve power and achieve the desired Type II error probability of 0.1. Therefore, Table 3.13 shows Type II error probabilities after increasing the sample size using the sample size adjustment factors obtained in Section 3.4.1. We observe that after increasing the sample size to adjust for the noise, we can control both the desired Type I and Type II error probabilities as Λ increases and when $\epsilon = 0.1$.

Figure 3.7 shows that large values of Λ in combination with small sample size

and $\epsilon = 0.1$ lead to a decline in power.

ϵ	Normal	Normal-Normal	Normal-Laplace
0.1	0.0682	0.0498	0.0533
0.2	0.0555	0.0492	0.0516
0.3	0.0533	0.0504	0.0497
0.4	0.0515	0.047	0.0487
0.5	0.0512	0.0481	0.0513

Table 3.9: Effect of ϵ on Type I error rates when $\Lambda = 1$ for $\alpha = 0.05$ and $\beta = 0.1$ without adjusting the sample size N (No correction, $K = 1$).

ϵ	Normal	Normal-Normal	Normal-Laplace
0.1	0.1236	0.1626	0.1609
0.2	0.1128	0.111	0.1137
0.3	0.1032	0.0999	0.1058
0.4	0.0978	0.1027	0.1035
0.5	0.109	0.1018	0.1049

Table 3.10: Effect of ϵ on Type II error rates when $\Lambda = 1$ for $\alpha = 0.05$ and $\beta = 0.1$ without adjusting the sample size N (No correction, $K = 1$).

Λ	Normal	Normal-Normal	Normal-Laplace
1	0.0678	0.0535	0.0516
2	0.1093	0.0472	0.0473
3	0.1524	0.0468	0.0511
4	0.191	0.0461	0.0516
5	0.2216	0.0451	0.0493
6	0.2547	0.0468	0.049
7	0.2699	0.0468	0.0516
8	0.2991	0.0502	0.0535
9	0.303	0.0476	0.0488
10	0.3246	0.0483	0.0489

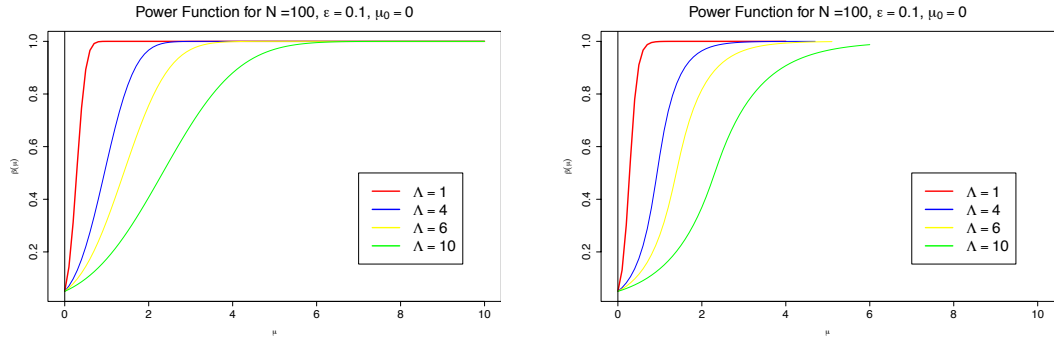
Table 3.11: Effect of Λ on Type I error rates when $\epsilon = 0.1$ for $\alpha = 0.05$ and $\beta = 0.1$ without adjusting the sample size N (No correction, $K = 1$).

Λ	Normal (No correction, $K = 1$)	Normal-Normal	Normal-Laplace
1	0.1247	0.1616	0.1565
2	0.1718	0.311	0.3172
3	0.2057	0.487	0.4821
4	0.2467	0.6375	0.6323
5	0.2678	0.7456	0.7334
6	0.2993	0.8053	0.7865
7	0.3149	0.8373	0.8247
8	0.3239	0.8574	0.8524
9	0.3454	0.8793	0.8744
10	0.3585	0.8821	0.8841

Table 3.12: Effect of Λ on Type II error rates when $\epsilon = 0.1$ for $\alpha = 0.05$ and $\beta = 0.1$ without adjusting the sample size N (No correction, $K = 1$).

Λ	Normal (No correction, $K = 1$)	Normal-Normal	Normal-Laplace
1	0.1277	0.0992	0.1029
2	0.1703	0.0961	0.0964
3	0.2031	0.0895	0.0985
4	0.2405	0.0872	0.0976
5	0.2709	0.0892	0.1041
6	0.2951	0.0839	0.0991
7	0.312	0.0877	0.0999
8	0.3207	0.0829	0.1027
9	0.3424	0.0821	0.1028
10	0.362	0.0868	0.1029

Table 3.13: Effect of Λ on Type II error rates when $\epsilon = 0.1$ for $\alpha = 0.05$ and $\beta = 0.1$ with $N = 857$ and private sample size $N' = KN$.



(a) *Power of Normal-Normal test* (b) *Power of Normal-Laplace test*

Figure 3.7: Power Functions of Normal-Normal test and Normal-Laplace test for $\mu_0 = 0$, $\epsilon = 0.1$ and $N = 100$.

3.8 Hypothesis test about μ with unknown variance

In this section, we study the effect of privacy on Type I and Type II error probabilities on two statistical procedures in rejecting the null hypothesis $H_0 : \mu = \mu_0$, when variance σ^2 is assumed to be unknown.

Without privacy, it is well known that the test statistic for the one sided test is the usual t-statistic [Casella and Berger, [3]]:

$$t = \frac{\sqrt{N}(\bar{X}-\mu_0)}{s}.$$

and the test of size α rejects the null hypothesis H_0 if $\bar{X} > \frac{s}{\sqrt{N}}t_{1-\alpha} + \mu_0$. The private test statistic is $T(X,R) = \frac{\mu(X,R)-\mu_0}{S_{\mu(X,R)}} \sim f_{priv}$ where $S_{\mu(X,R)}$ is the standard error of the private estimator $\mu(X,R)$.

The standard test rejects $H_0 : \mu = \mu_0$ against $H_1 : \mu > \mu_0$ when:

$$\mu(X,R) > \mu_0 + t_{1-\alpha}\frac{S}{\sqrt{N}}. \tag{3.7}$$

In Section 3.4.1 we established that we can approximate the sampling distribution of the private estimator $\mu(X,R)$ under H_0, by the Normal-Normal distribution, that is $\mu(X,R) \sim \mathrm{N}(\mu_0, \frac{\sigma^2}{N} + \frac{2\Lambda^2}{(\epsilon N)^2})$. Therefore, under H_0 the test statistic $t(X,R)$ follows the standard normal distribution:

$$t(X,R) = \frac{\mu(X,R)-\mu_0}{\sqrt{\frac{\sigma^2}{N} + \frac{2\Lambda^2}{\epsilon^2N^2}}} \sim N(0,1). \tag{3.8}$$

Also, since $S^2 \xrightarrow{P} \sigma^2$, then

$$\sqrt{\frac{\frac{S^2}{N} + \frac{2\Lambda^2}{\epsilon^2N^2}}{\frac{\sigma^2}{N} + \frac{2\Lambda^2}{\epsilon^2N^2}}} \xrightarrow{P} 1.$$

Then by Slutzky's theorem we have that the private statistic with unknown variance converges in distribution to $N(0,1)$,

$$\frac{\mu(X,R)-\mu_0}{\sqrt{\frac{S^2}{N} + \frac{2\Lambda^2}{\epsilon^2N^2}}} \xrightarrow{d} N(0,1)\ .$$

Thus, the Normal-Normal test with unknown σ^2, rejects $H_0 : \mu = \mu_0$ against $H_1 : \mu > \mu_0$ when:

$$\mu(X,R) > \mu_0 + z_{1-\alpha}\sqrt{\frac{S^2}{N} + \frac{2\Lambda^2}{\epsilon^2N^2}}. \tag{3.9}$$

3.8.1 Simulation study IV with unknown variance σ^2

In this simulation, the two test procedures (3.7) and (3.9) with critical points $t_1 = \mu_0 + t_{1-\alpha}\frac{S}{\sqrt{N}}$ and $t_2 = \mu_0 + z_{1-\alpha}\sqrt{\frac{S^2}{N} + \frac{2\Lambda^2}{\epsilon^2 N^2}}$, respectively, are studied. We simulated 10,000 samples from the Normal distribution with $\mu_0 = 0$ and sample variance $s^2 = 1.1$. For each simulation we assumed $\gamma = 0.1$, $\alpha = 0.05$, $\beta = 0.1$ and $N = 857$. The empirical results presented in Tables 3.14, 3.15, 3.16 and 3.17 are the Type I error and Type II error probabilities for the two testing procedures. Again, we estimated the Type I and Type II error rates based on different simulated samples as we varied ϵ from 0.1 to 0.5 and Λ from 1 to 10.

The simulation results when the variance is unknown agree with the results of the simulation study in Section 3.7 where the variance was known. More specifically, from Table 3.14 we can see that when $\Lambda = 1$, and as ϵ increases there is no significant inflation on the Type I error for either statistical test procedures. Additionaly, Table 3.15 shows that the power for the Normal-Normal test is always greater than that of the Normal test, except for $\epsilon = 0.5$. On the other hand, Table 3.16 shows that when Λ increases and $\epsilon = 0.1$, the Type I error for the Normal test is highly inflated, whereas the Type I error rate is maintained for the Normal-Normal test. Finally, Table 3.17 shows the effect of the sample size adjustment $N' = KN$, where K is the Normal-Normal sample size adjustment factor (see Section 3.41, formula (3.4)), when Λ increases and $\epsilon = 0.1$. It is clear that the sample size adjustment helps in increasing the power for the Normal-Normal test.

ϵ	Normal	Normal-Normal
0.1	0.0615	0.0492
0.2	0.0534	0.0545
0.3	0.0545	0.048
0.4	0.0474	0.0509
0.5	0.0483	0.0536

Table 3.14: Type I error rates when $s^2 = 1.1$, $\Lambda = 1$, $\alpha = 0.05$, $\beta = 0.1$ and $N = 857$.

ϵ	Normal	Normal-Normal
0.1	0.117	0.0898
0.2	0.1059	0.0972
0.3	0.097	0.0927
0.4	0.0948	0.0909
0.5	0.0936	0.0959

Table 3.15: Type II error rates when $s^2 = 1.1$, $\Lambda = 1$, $\alpha = 0.05$, $\beta = 0.1$ and $N = 857$.

Λ	Normal	Normal-Normal
1	0.0608	0.0432
2	0.1025	0.0423
3	0.149	0.0456
4	0.1822	0.0463
5	0.2198	0.0483
6	0.2398	0.0459
7	0.2696	0.0445
8	0.2845	0.0486
9	0.3078	0.0476
10	0.3158	0.0466

Table 3.16: Type I error rates when $s^2 = 1.1$, $\epsilon = 0.1$, $\alpha = 0.05$, $\beta = 0.1$ and $N = 857$.

Λ	Normal	Normal-Normal
1	0.1319	0.109
2	0.1726	0.0974
3	0.2161	0.094
4	0.2485	0.0955
5	0.2788	0.0961
6	0.305	0.0922
7	0.3287	0.0879
8	0.3408	0.0852
9	0.3484	0.0804
10	0.3652	0.082

Table 3.17: Type II error rates when $s^2 = 1.1$, $\epsilon = 0.1$, $\alpha = 0.05$, $\beta = 0.1, N = 857$ and $N' = KN$.

3.9 Gaussian noise

In this section, we revisit the point estimation and sample size determination problem for the normal distribution. We add noise to the maximum likelihood estimator from the normal distribution scaled according to the ℓ_2-global sensitivity. The perturbed private estimator $\mu(X, R)$ is (ϵ, δ)-differentially private [Nissim et al., [17]]. We relax the definition of privacy for simplicity of our calculations, with the hope of getting better utility. The output perturbation mechanism with Gaussian noise is described in Algorithm 3.

Algorithm 3: A Differentially Private Estimator with Gaussian noise.

Input A data set $\mathbf{x} \in \mathbb{R}^N$, $\epsilon > 0$. Assume that Λ is the range of $T(\mathbf{x})$ or the diameter of the parameter space.

1. Obtain the sufficient statistic $T(\mathbf{x})$ for μ.
2. Draw a random observation R from the Gaussian distribution with mean 0 and standard deviation $\sigma^* = \frac{\Lambda/N\sqrt{2ln(1/\delta)}}{\epsilon} > 0$.
3. Return $\mu(X, R) = \bar{x} + R$.

To prove that this algorithm is (ϵ, δ)-differentially private we need the Mills inequality.

Proposition 3.9.1 (Mills Inequality)**.** If $Z \sim N(0,1)$ then for every $t > 1$

$$P(|Z| > t) \leq e^{-t^2/2}\,.$$

Proposition 3.9.2. Algorithm 3 is (ϵ, δ)-differentially private [Thakurta [21]].

Proof. Let $\mathbf{x}, \mathbf{x}'$ be databases that differ only in one element. The global sensitivity of $\hat{\mu}$ in the ℓ_2-metric is $GS_{\hat{\mu}} = \frac{\Lambda}{N}$. The random variables $\mu(\mathbf{x}, R)$ and $\mu(\mathbf{x}', R)$ are normal random variables with means $\bar{x}$ and $\bar{x'}$ respectively and identical standard deviations σ^*. Let h_x and $h_{x'}$ be the corresponding density functions. Then for any $y \in \mathbb{R}$ we have:

$$\begin{aligned}
\frac{h_x(y)}{h_{x'}(y)} &= \frac{e^{\frac{-1}{2\sigma^{*2}}(y-f(x))^2}}{e^{\frac{-1}{2\sigma^{*2}}(y-f(x'))^2}} \\
&= e^{-\frac{1}{2\sigma^{*2}}((y-f(x))^2-(y-f(x'))^2)} \\
&= e^{\frac{1}{2\sigma^{*2}}|(y-f(x))^2-(y-f(x'))^2|} \\
&= e^{\frac{1}{2\sigma^{*2}}|f(x')-f(x)||y-f(x)+y-f(x')|} \\
\text{since } |f(x') - f(x)| \leq GS_{\hat{\mu}} & \\
&\leq e^{\frac{GS_{\hat{\mu}}}{2\sigma^{*2}}(|y-f(x')|+|y-f(x)|)}.
\end{aligned}$$

Thus

$$\frac{h_x(y)}{h_{x'}(y)} \leq e^{\frac{GS_{\hat{\mu}}}{2\sigma^{*2}}(|y-f(x')|+|y-f(x)|)}. \tag{3.10}$$

Hence, since $\frac{\mu(x,R)-f(x)}{\sigma^*} \sim N(0,1)$ using Mill's inequality we have

$$P(|\mu(x,R) - f(x)| > \sigma^* t)) \leq e^{-t^2/2}\,. \tag{3.11}$$

Letting $t = \sqrt{2ln\frac{1}{\delta}}$ we get $e^{-t^2/2} = \delta$ and in order to have $t > 1$ we choose $\delta \leq \frac{2}{\sqrt{\epsilon}}$. Let B_x be the set such that $B_x = [y \in \mathbb{R} : |y - f(x)| < \sigma^* t]$. We want the

noise to be in the set B_x with probability at least $1-\delta$. Thus replacing $t = \sqrt{2ln\frac{1}{\delta}}$ in (3.11) we get:

$$P(\mu(x,R) \in B_x^c) = P(|\mu(x,R) - f(x)| > \sigma^* \sqrt{2ln(1/\delta)}) \leq \delta \ . \tag{3.12}$$

Then from (3.10) we obtain:

$$\begin{aligned} \frac{h_x(y)}{h_{x'}(y)} &\leq e^{\frac{GS_{\hat{\mu}}}{2\sigma^{*2}} 2\sigma^* \sqrt{2ln(1/\delta)}} \\ &= e^{\frac{GS_{\hat{\mu}} \sqrt{2ln(1/\delta)}}{\sigma^*}} \ . \end{aligned} \tag{3.13}$$

with probability at least $1-\delta$. Solving for σ^* we get $\sigma^* \geq \frac{GS_{\hat{\mu}} \sqrt{2ln(1/\delta)}}{\epsilon}$. To complete the proof we show the following:

Let $S \subseteq \mathbb{R}$. Then using (3.12) and (3.13)

$$\begin{aligned} P(\mu(x,R) \in S) &= P(\mu(x,R) \in S, \mu(x,R) \in B_x)) + P(\mu(x,R) \in S, \mu(x,R) \in B_x^c)) \\ &\leq P(\mu(x,R) \in S, \mu(x,R) \in B_x)) + \delta \\ &\leq e^{\epsilon} P(\mu(x',R) \in S) + \delta \ . \end{aligned}$$

□

Comparing the private estimator of Algorithm 3 with the private estimator of Algorithm 1 we observe that the variance of the new estimator increases by a factor $ln(1/\delta)$.

3.9.1 Sample size determination with Gaussian noise

We are interested in testing the following hypothesis:

$$H_0 : \mu = \mu_0 \qquad H_1 : \mu = \mu_1 = \mu_0 + \gamma$$

where $\gamma = \mu_1 - \mu_0 > 0$ is the noncentrality parameter.

After adding Gaussian noise $N(0, (\frac{\Lambda/N'\sqrt{2ln(1/\delta)}}{\epsilon})^2)$, the distribution of the private statistic $\mu(X, R)$ under the null hypothesis is $N(\mu_0, (\frac{\Lambda/N'\sqrt{2ln(1/\delta)}}{\epsilon})^2)$ and under the alternative hypothesis the distribution is $N(\mu_0 + \gamma, (\frac{\Lambda/N'\sqrt{2ln(1/\delta)}}{\epsilon})^2)$ where $\delta = \frac{1}{N'}$. To achieve Type I error α and power $1 - \beta$ the private sample size N' is obtained by solving the following non-linear equation for N' :

$$N'^2 - NN' - \frac{2\Lambda^2(z_{1-\alpha} + z_{1-\beta})ln(N')}{\epsilon^2\gamma^2} = 0. \tag{3.14}$$

and the exact adjustment sample size factor K is obtained by dividing the private sample size N' by the true classical sample size N. The solution of the above equation is computed numerically in R [18, 18] and the code is given in the Appendix A.

3.9.2 A hypothesis testing procedure with Gaussian noise

Under differential privacy the statistic with Gaussian noise is $\mu(X, R) = \bar{X} + N(0, (\frac{\Lambda/N\sqrt{2ln(1/\delta)}}{\epsilon})^2)$. We considered two testing procedures to reject the null hypothesis $H_0 : \mu = \mu_0$.

The first statistical procedure is based on the fact that under the null hypothesis the asymptotic distribution of $\mu(X, R)$ is the standard normal distribution. Therefore, this test rejects H_0 using the non-private critical value $t_1 = \mu_0 + z_{1-\alpha}\frac{\sigma}{\sqrt{N}}$. That is,

$$\text{Reject } H_0 \text{ when } \mu(X, R) > t_1 = \mu_0 + z_{1-\alpha}\frac{\sigma}{\sqrt{N}}. \tag{3.15}$$

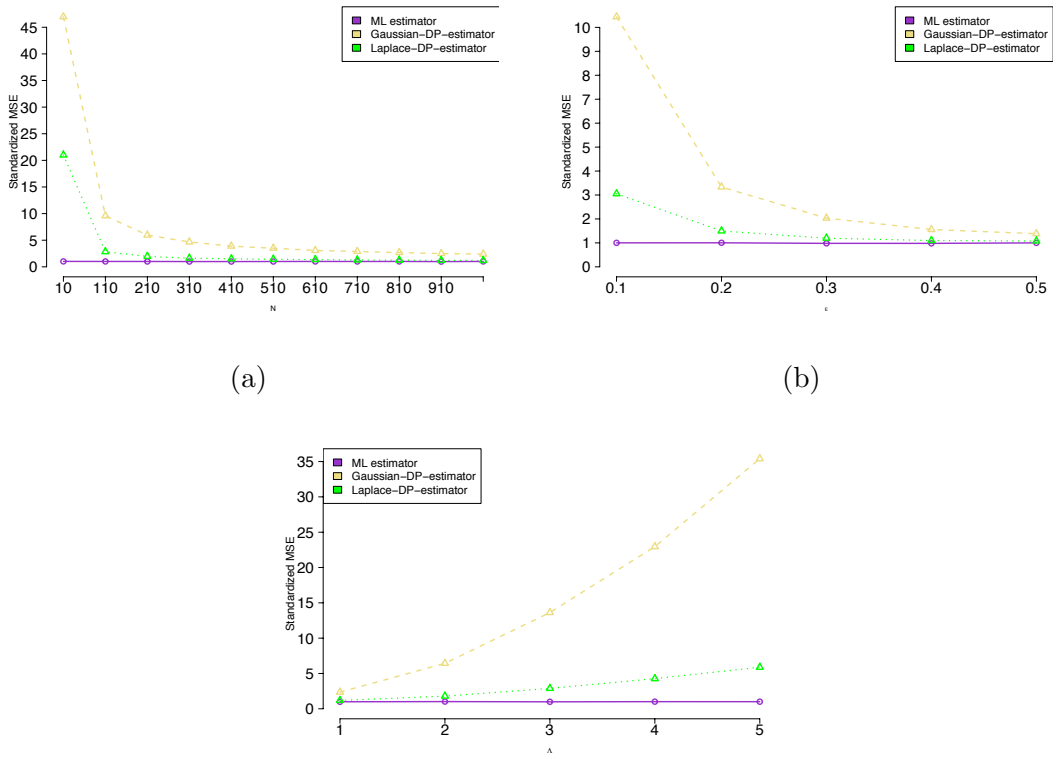
In the second case, the distribution of the private estimator $\mu(X, R)$ under the null hypothesis is the exact normal distribution with mean μ_0 and variance $\frac{\sigma^2}{N} + \frac{2\Lambda^2 ln(1/\delta)}{\epsilon^2 N^2}$ since it is the convolution of two independent normal random variables. Therefore, in the second case

$$\text{Reject } H_0 \text{ when } \mu(X, R) > t_2 = \mu_0 + z_{1-\alpha}\sqrt{\frac{\sigma^2}{N} + \frac{2\Lambda^2 ln(1/\delta)}{\epsilon^2 N^2}}. \tag{3.16}$$

3.9.3 Simulation study V with Gaussian noise

The first simulation investigates the statistical efficiency of the private estimator with Gaussian noise. The simulation demonstrates the interactive effect of the sample size N, the privacy parameter ϵ and diameter Λ on the standardized mean squared error and compares the efficiency of the private estimator with Gaussian noise to that of the maximum likelihood estimator and Algorithm 1 (Laplace noise). The second simulation computes the sample size adjustement factor (3.14) with changing values of ϵ and Λ and studies the effect of sample size adjustment $N' = KN$ on the Type I and Type II error rates for the test (3.15) with the classical critical value $\mu_0 + z_{1-\alpha}\frac{\sigma}{\sqrt{N}}$. The third simulation compares the Type I and Type II error performance for the two private hypothesis testing procedures (3.15) and (3.16) with critical values $t_1 = \mu_0 + z_{1-\alpha}\frac{\sigma}{\sqrt{N}}$ and $t_2 = \mu_0 + z_{1-\alpha}\sqrt{\frac{\sigma^2}{N} + \frac{2\Lambda^2 ln(1/\delta)}{\epsilon^2 N^2}}$, respectively. In all simulations, we draw $M = 10,000$ samples of N data points from a normal distribution with parameters $\mu = 0$ and $\sigma^2 = 1$. We add Gaussian noise to the maximum likelihood estimator and used $\delta = \frac{1}{N}$. To compute the sample size adjustment factor K, Type I and Type II error rates we used $\alpha = 0.05$, $\beta = 0.1$ and $\gamma = 0.1$. The privacy level ϵ varied from 0.1 to 0.5 and the range of the parameter space Λ varied from 1 to 10.

Figure 3.8 shows the effect of ϵ, Λ and the true sample size N on the standardized mean squared error for Algorithm 3 and Algorithm 1. We observe that Algorithm 1 outperforms Algorithm 3. As expected, the mean square error of the Gaussian private estimator is always greater than the mean squared of the Laplace estimator exactly by a factor of $ln(1/\delta)$.



(a) (b)

(c)

Figure 3.8: Standardized MSE of MLE versus Gaussian and Laplace private estimators.

Tables 3.18 and 3.19 compare the sample size correction factors K with Gaussian noise at two different powers $1-\beta = 0.6$ and 0.9. In comparison with the sample size adjustment factor K obtained with Laplace noise (see simulation study II, Tables 3.1-3.4), after adding Gaussian noise, the range of the sample size adjustment factor K increases. For example, in order to achieve statistical power of 0.6 and privacy $\epsilon = 0.1$ the private sample N is 2.503 times the true sample size with Gaussian noise while with Laplace noise the private sample size is 1.382 times the true sample size (Table 3.1). This is expected since Algorithm 3 outputs an estimator with greater variance.

ϵ		0.1	0.2	0.3	0.4	0.5
$\beta = 0.1$	K	1.905	1.313	1.155	1.092	1.061
$\beta = 0.4$	K	2.504	1.562	1.293	1.180	1.119

Table 3.18: Sample size adjustment factor K when $\Lambda = 1$ with Type II errors $\beta = 0.1$ and $\beta = 0.4$.

Λ		1	2	3	4	5	6	7	8
$\beta = 0.1$	K	2.504	4.582	6.750	8.972	11.220	13.512	15.819	18.144
$\beta = 0.4$	K	1.905	3.267	4.705	6.178	7.678	9.193	10.724	12.267

Table 3.19: Sample size adjustment factor K when $\epsilon = 0.1$ with Type II errors $\beta = 0.1$ and $\beta = 0.4$.

The next set of simulations provide Type I and Type II error rates of the test that rejects H_0 if $\mu(X, R) > \mu_0 + \frac{\sigma}{\sqrt{N}} z_{1-\alpha}$ with nominal level $\alpha = 0.05$ and power $1 - \beta = 0.9$, for the original sample size $N = 857$, and the adjusted sample size $N' = KN$ with Gaussian noise. From Table 3.20, we observe that when $\Lambda = 1$ and ϵ increases from 0.1 to 0.5, the test gives inflated Type I error rates when there is no correction of the true sample size N. Similarly, Table 3.21 shows that the test under no correction has low power. On the other hand, the sample size adjustment factors results in a decrease in Type I error and an increase in power as ϵ increases.

Tables 3.22 and 3.23 show the simulation results of Type I and Type II error rates for different values of Λ and when $\epsilon = 0.1$. As expected, due to the departure from normality, the Type I and Type II error rates are inflated as Λ increases with the true classical sample size N. Although, the sample size adjustment factors lead to an increase in power (Table 3.23, second row) there is no improvement in Type I error rates (Table 3.22, first row)

ϵ	0.1	0.2	0.3	0.4	0.5
No correction	0.148	0.0823	0.0634	0.0638	0.0542
K correction	0.1194	0.0764	0.0587	0.0594	0.0539

Table 3.20: Type I error rates with $\Lambda = 1$, $\alpha = 0.05$, $\beta = 0.1$, $N = 857$ when $\mu_0 = 0$ and $\sigma^2 = 1$.

ϵ	0.1	0.2	0.3	0.4	0.5
No correction	0.2131	0.1368	0.1151	0.1056	0.1083
K correction	0.0427	0.0663	0.0826	0.088	0.0889

Table 3.21: Type II error rates with $\Lambda = 1$, $\alpha = 0.05$, $\beta = 0.1$, $N = 857$ when $\mu_0 = 0$ and $\sigma^2 = 1$.

Λ	1	2	3	4	5	6	7	8
No correct.	0.153	0.270	0.337	0.371	0.400	0.411	0.425	0.429
K correct.	0.114	0.183	0.226	0.253	0.281	0.292	0.312	0.315

Table 3.22: Type I error rates with $\epsilon = 0.1$, $\alpha = 0.05$, $\beta = 0.1$, $N = 857$ when $\mu_0 = 0$ and $\sigma^2 = 1$.

Λ	1	2	3	4	5	6	7	8
No correct.	0.210	0.315	0.365	0.396	0.424	0.436	0.444	0.444
K correct.	0.044	0.021	0.016	0.011	0.010	0.007	0.007	0.008

Table 3.23: Type II error rates with $\epsilon = 0.1$, $\alpha = 0.05$, $\beta = 0.1$, $N = 857$ when $\mu_0 = 0$ and $\sigma^2 = 1$.

Type I error and Type II error rates for the two tests (3.15) and (3.16) are reported in Tables 3.24, 3.25, 3.26 and 3.27. The simulation results are similar to those with Laplace noise (Sections 3.6 and 3.7, Tables 3.9, 3.10, 3.11 3.12 and 3.13). The Type I error rates and Type II error rates for both tests were estimated using the true classical sample size N. Tables 3.24 and 3.25 show that for $\Lambda = 1$ and as ϵ increases, the two private tests with Laplace noise (Normal-Normal and Normal-Laplace tests) had better Type I and Type II error rates than the tests (3.15) and (3.16) with Gaussian noise.

From the first row of Table 3.26, we observe that when $\epsilon = 0.1$, the test (3.15) has large Type I error rates. On the other hand, increasing the value of Λ does not appear to cause the test (3.16) to give inflated Type I error rates as Λ increases. However, from Table 3.27 we observe that Λ affects the power for both tests. Both tests tend to give smaller power as the diameter Λ increases with the true sample size N. This suggests that adjusting the true sample size N using the sample size

adjustment factor (3.14) result in the increase of power for the test in (3.16). This is left to future explorations.

ϵ	0.1	0.2	0.3	0.4	0.5
t_1	0.1573	0.0778	0.062	0.0641	0.0575
t_2	0.0497	0.0546	0.0483	0.048	0.0483

Table 3.24: Type I error rates with $\Lambda = 1$, $\alpha = 0.05$, $\beta = 0.1$, $N = 857$ when $\mu_0 = 0$ and $\sigma^2 = 1$.

ϵ	0.1	0.2	0.3	0.4	0.5
t_1	0.215	0.138	0.121	0.112	0.198
t_2	0.432	0.204	0.144	0.123	0.114

Table 3.25: Type II error rates with $\Lambda = 1$, $\alpha = 0.05$, $\beta = 0.1$, $N = 857$ when $\mu_0 = 0$ and $\sigma^2 = 1$.

Λ	1	2	3	4	5	6	7	8
t_1	0.140	0.273	0.336	0.377	0.399	0.415	0.425	0.437
t_2	0.05	0.048	0.053	0.051	0.052	0.043	0.0.054	0.052

Table 3.26: Type I error rates with $\epsilon = 0.1$, $\alpha = 0.05$, $\beta = 0.1$, $N = 857$ when $\mu_0 = 0$ and $\sigma^2 = 1$.

Λ	1	2	3	4	5	6	7	8
t_1	0.216	0.322	0.363	0.394	0.427	0.441	0.442	0.449
t_2	0.429	0.716	0.812	0.853	0.879	0.896	0.9003	0.913

Table 3.27: Type II error rates with $\epsilon = 0.1$, $\alpha = 0.05$, $\beta = 0.1$, $N = 857$ when $\mu_0 = 0$ and $\sigma^2 = 1$.

3.10 Summary

In this chapter we designed two different differentially private algorithms for the mean of a normal random variable with Laplace noise (Algorithm 1) and Gaussian

noise (Algorithm 3). The simulation results, indicate that the output perturbation Algorithm 3 is of limited use since it performs poorly on the simulated data compared to Algorithm 1. In fact, we see that the MSE of Algorithm 3 is always greater than that of Algorithm 1.

We also estimate the effect of sample size adjustment on Type I and Type II error rates for the Normal test that rejects $H_0 : \mu = \mu_0$ against $H_1 : \mu > \mu_0$ with the classical critical value $\mu_0 + \frac{\sigma}{\sqrt{N}} z_{1-\alpha}$. The results show that when $\Lambda = 1$ and with sample size adjustment $N' = KN$ to adjust for the added noise, the test has estimated Type I error rates near the nominal α level and good statistical power for varying values of ϵ. This empirical result, supports the theory in Section 3.3.1 that the private estimator $\mu(X, R)$ is asymptotically normal, given that the sample size is large enough relative to the range of the diameter space Λ. However, when Λ increases, with a moderate sample size N and a fixed value of ϵ, the private statistic is not asymptotically normal. Thus, the Normal test displays inflated Type I error rates and small statistical power. But using the proposed sample size adjustment factors we observe an increase in power and stable Type I error.

We also propose two hypothesis testing procedures and compare them in terms of their achieved Type I error and Type II error rates. The simulations show that both the proposed Normal-Normal and the Normal-Laplace tests outperform the Normal test in Type I error performance for a moderate true sample size N (no sample size adjustment). More specifically, the Type I error rates stay near the nominal α level as Λ increases. Moreover, the results also indicate the use of the proposed sample size adjustment factors result in improvement in statistical power. Thus the best results are obtained by using both the new critical test values and the sample size adjustment factors.

Finally, the simulation study V shows that the Type I performance of the hypothesis test based on Algorithm 3 is always lower than the error performance of the hypothesis test based on Algorithm 1. This also holds for the case when sample size adjustment is applied.

Chapter 4

Smooth Sensitivity

In Chapter 3, we showed that to ensure differential privacy in point estimation and hypothesis testing, is sufficient to perturb the value of the maximum likelihood estimate with random noise (from e.g., the Gaussian or Laplace distribution) of magnitude proportional to the global sensitivity of the MLE, $GS_{\hat{\mu}} = \frac{\Lambda}{N}$. However, the simulation results showed that as Λ increases the MLE has high global sensitivity, making the differential private estimator useless for statistical inference unless we increase the sample size significantly in certain settings. In this chapter, we suggest a new algorithm based on [Nissim et al., [17]] that allows one to release the MLE with instance-based additive noise. That is, the noise magnitude is determined not only by the MLE, but also by the dataset itself. To ensure that the noise magnitude does not leak any information about the database, we calibrate the noise magnitude to the *smooth sensitivity* of the MLE on the database $\mathbf{x}$-a measure of the variability of the MLE in the neighborhood of $\mathbf{x}$ as proposed by [Nissim et al., [17]].

4.1 Our contribution

In Section 4.2, we define the smooth bound and smooth sensitivity, and we show that adding noise proportional to the smooth sensitivity is (ϵ, δ)-differentially private. The results of this section are drawn from [Nissim et al., [17]]. In Section 4.2.2, we give the instance-based algorithm (Algorithm 4) for the maximum likelihood estimator of the mean of a normal random variable that satisfies (ϵ, δ)-

differential privacy. In Section 4.3, we evaluate the accuracy of Algorithm 4 and compare its performance with that of Algorithms 1 and 3 derived in Chapter 3. The simulation results show that Algorithm 4 is more accurate than Algorithms 1 and 3 even for large values of Λ and small sample size.

4.2 A differentially private estimator with instance - based noise

Nissim et al., [17] introduce the idea of instance-dependent noise. They defined the *local sensitivity* LS_f of a function f at a particular data $\mathbf{x}$. However, they show that adding noise proportional to the local sensitivity does not satisfy differential privacy, as the magnitude of the noise itself reveals information about the database. Therefore, they define a class of smooth upper bounds S_f to LS_f, such that adding noise proportional to S_f is differentially private. Moreover, they define the *smooth sensitivity* function S_f^* that is optimal, in the sense that $S_f(\mathbf{x}) > S_f^*(\mathbf{x})$ for every other smooth S_f and show that algorithms that use smooth sensitivity perform well.

4.2.1 Smooth bounds and smooth sensitivity

Definition 4.2.1 (Local Sensitivity, Nissim et al., [17])**.** For $f : \mathcal{D}^n \rightarrow \mathbb{R}^d$ and $\mathbf{x} \in \mathcal{D}^n$, the local sensitivity of f at $\mathbf{x}$ is

$$LS_f(\mathbf{x}) = \max_{\mathbf{y}:d(\mathbf{x},\mathbf{y})=1} ||f(\mathbf{x}) - f(\mathbf{y})||$$

Note that the global sensitivity is $GS_f = max_{\mathbf{x}} LS_f(\mathbf{x})$. An algorithm that releases f with noise with magnitude proportional to $LS_f(\mathbf{x})$ on input $\mathbf{x}$, is not in general differential private, since the magnitude of the noise can leak information [Nissim et al., [17]]. Hence, the goal is to add instance-based noise with a smaller magnitude than the worst case noise GS_f/ϵ, and yet satisfy differential privacy. Nissim et al., [17] propose the instance-based noise framework that expands the applicability of output perturbation. The new framework, instead of using the local sensitivity, relases $f(\mathbf{x})$ with noise with magnitude according to a smooth upper

bound on the local sensitivity, namely, a function S that is an upper bound on $LS_f(\mathbf{x})$ at all points and such that $\ln(S(\cdot))$ has low global sensitivity. This bound is called the *smooth sensitivity* of f. We say that S is ϵ-smooth if $GS_{ln(S(\cdot))} \leq \epsilon$ [Nissim et al., [17]].

Definition 4.2.2 (A Smooth Bound, Nissim et al., [17])**.** For $\beta > 0$ a function $S : \mathcal{D}^n \to \mathbb{R}^+$ is a β-smooth upper bound on the local sensitivity of f if it satisfies the following requirements:

$$\begin{aligned} \forall \mathbf{x} \in \mathcal{D}^n : \quad & S(\mathbf{x}) \geq LS_f(\mathbf{x}) \\ \forall \mathbf{x}, \mathbf{y}, d(\mathbf{x}, \mathbf{y}) = 1 : \quad & S(\mathbf{x}) \leq e^{\beta} S(\mathbf{y}) \end{aligned}$$

Definition 4.2.3 (Smooth Sensitivity, Nissim et al., [17])**.** For $\beta > 0$, the β-smooth sensitivity of f is

$$S^*_{f,\beta}(\mathbf{x}) = \max_{\mathbf{y} \in \mathcal{D}^n} (LS_f(\mathbf{y}) \cdot e^{-\beta d(\mathbf{x},\mathbf{y})}).$$

The smooth sensitivity $S^*_{f,\beta}$, is the smallest function that satisfies Definition 4.2.2.

Lemma 4.2.1 (Nissim et al., [17])**.** $S^*_{f,\beta}$ is a β-smooth upper bound on LS_f. In addition, $S^*_{f,\beta}(\mathbf{x}) \leq S(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{D}^n$ for every β-smooth upper bound S on LS_f.

Note that the constant function $S(x) = GS_f$ also satisfies Definition 4.2.2.

Next, we show that by adding noise drawn from a noise distribution to the value of $f(\mathbf{x})$ with standard deviation proportional to the smooth sensitivity yields an (ϵ, δ) - differentially private mechanism. The noise to be added is proportional to $S^*_{f,\beta}/\alpha$, where $S^*_{f,\beta}$ is the smooth sensitivity of f and α and β depend on the privacy parameters ϵ and δ.

Consider the largest local sensitivity attained when up to k entries of $\mathbf{x}$ are modified. The local sensitivity of f at distance k is:

$$LS^{(k)}(\mathbf{x}) = \max_{\mathbf{y} \in \mathcal{D}^n : d(\mathbf{x},\mathbf{y}) \leq k} LS_f(\mathbf{y}).$$

Now the smooth sensitivity can be expressed in terms of $LS^{(k)}$ [Nissim et al., [17]] :

$$S^*_{f,\beta} = \max_{k=0,...,n} e^{-k\beta} LS^{(k)}(\mathbf{x}). \tag{4.1}$$

Thus, to compute the smooth sensitivity of f at $\mathbf{x}$, it suffices to understand $LS^{(k)}$. In addition, one can further break down the expression for smooth sensitivity by considering data $\mathbf{y}$ that differ from $\mathbf{x}$ at distance at most 1 [Karwa et al., [14]].

Theorem 4.2.1 (Nissim et al., [17])**.** Let f be a real-valued function and let $S^*_{f,\beta}$ be its β-smooth sensitivity. The mechanism $\mathcal{A}(\mathbf{x}) = f(\mathbf{x}) + \frac{S^*_{f,\beta}(\mathbf{x})}{\alpha} \cdot Z$, where Z is a random variable sampled from a noise distribution, is (ϵ, δ)-differentially private.

4.2.2 Maximum likelihood estimator

In this section, we compute the smooth sensitivity of the maximum likelihood estimator of the mean μ of the normal random variable and we present an algorithm that uses the smooth sensitivity framework of [Nissim et al., [17]] to release the private maximum likelihood estimator.

Let $X = (X_1, .., X_N) \sim N(\mu, \sigma^2)$ where $\sigma > 0$ is known and suppose $X_i \in [-\Lambda, \Lambda], i = 1, ..., N$. Let $\hat{\mu}$ be the sample mean, the maximum likelihood estimator of μ.

Proposition 4.2.1. The smooth sensitivity of the MLE is

$$S^*_{\hat{\mu},\beta} = \max_{k=0...N} e^{-k\beta} \Big| \frac{X_N + ... + X_{N-k+1} - k\hat{\mu}(x)}{N-k} \Big|. \tag{4.2}$$

Proof. Let $\hat{\mu}(\mathbf{x})$ be the MLE of μ at database $x = (x_1, ..., x_N)$. Assume that the database entries are sorted in the nondecreasing order: $x_1 \leq ... \leq x_N$. The local sensitivity at distance 1 is maximized by deleting the maximum entry x_N of the database. Thus, $LS^{(1)}(x) = \left| \frac{X_N - \hat{\mu}(x)}{N-1} \right|$. The local sensitivity of $\hat{\mu}$ at distance k is obtained analogously:

$$LS^{(k)} = \Big| \frac{X_N + ... + X_{N-k+1} - k\hat{\mu}(x)}{N-k} \Big|$$

□

We do not have a closed form of the smooth sensitivity of the MLE but it can be evaluated numerically in R [18, 18]. Algorithm 4 releases $\mathcal{A}(x)$ adding small amount of noise sampled either from the Laplace or the Gaussian distribution.

Algorithm 4: A Differentially Private estimator with Smooth Sensitivity.

Input A data set $\mathbf{x} \in \mathbb{R}^N$, parameters ϵ, δ and Λ.

1. Set $\beta = \epsilon/2\ln(1/\delta)$.
2. Compute $S^*_{\hat{\mu},\beta} = \max_{k=0...N} e^{-k\beta} A^{(k)}(x)$.
3. Either sample R from the Laplace distribution with mean 0 and standard deviation $\frac{S^*_{\hat{\mu},\beta}}{\epsilon/2}$ or sample R from the Gaussian distribution with mean 0 and standard deviation $\frac{S^*_{\hat{\mu},\beta}}{\epsilon/\sqrt{\ln(1/\delta)}}$.
4. Return $\mathcal{A}(\mathbf{x}) = \hat{\mu}(\mathbf{x}) + R$.

Lemma 4.2.2. Algorithm 4 is (ϵ, δ)-differentially private.

The proof of Lemma 4.2.2 is based on the Theorem 4.2.1 [Nissim et al., [17]].

4.3 Simulation study

To evaluate the performance of the private estimator with smooth sensitivity and compare its performance with the estimators from Sections 3.3 and 3.8 we have conducted a number of simulations studies. We evaluate the quality of the instance-dependent Algorithm 4 using the standardized MSE. We generate $M = 10,000$ samples from the normal distribution with $\mu = 0$ and $\sigma = 1$ and $X_i \in [-\Lambda, \Lambda]$, $i = 1, ..., N$, for various values of Λ and ϵ.

The first simulation compares Algorithms 1 and 3 with the instance-dependent Algorithm 4 for different values of ϵ and N when $\Lambda = 1$. The second simulation evaluates the quality of the estimators of Algorithm 4 as we vary the privacy parameter ϵ and N for two different values of $\Lambda = 1$, $\Lambda = 1$ and $\Lambda = 10$. We generate tables and graphs with ϵ varying from 0.1 to 0.5 in steps of 0.1 and N between 10 and 100 in steps of 10.

Remark: The instance - based Algorithm 4 and Algorithm 3 satisfy a different definition of privacy from Agorithm 1. Algorithm 1 satisfies ϵ - differential privacy while Algorithms 3 and 4 satisfy (ϵ, δ)-differential privacy.

4.3.1 Evaluation of Algorithm 4

As we see from Table 4.1 and Figure 4.1a the performance of Algorithm 4 improves as the sample size N increases when $\epsilon = 0.1$ and when $\Lambda = 1$. Table 4.2 shows a similar performance of the instance-dependent Algorithm 4 when $\Lambda = 10$. Consequently, Algorithm 4 is not sensitive to the value of Λ. Moreover, Table 4.3 and Figure 4.1 show that even for a high privacy level ($\epsilon = 0.1$) Algorithm 4 performs well when $N = 100$ and $\Lambda = 1$.

N	MLE	LapS	GausS
10	0.897	3.612	2.503
20	0.990	1.737	1.552
30	1.014	1.252	1.214
40	1.002	1.237	1.214
50	0.973	1.208	1.203
60	1.038	1.180	1.184
70	0.938	1.107	1.114
80	1.018	1.087	1.096
90	1.009	1.081	1.091
100	0.930	1.058	1.067

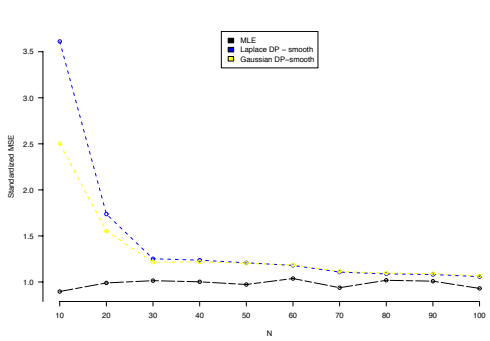
Table 4.1: Standardized MSE for MLE, the instance-dependent Algorithm 4 when $\epsilon = 0.1$ and $\Lambda = 1$. LapS is Algorithm 4 with Laplace noise and GausS is Algorithm 4 with Gaussian noise.

N	MLE	LapS	GausS
10	0.826	4.485	3.006
20	0.947	1.429	1.321
30	1.002	1.217	1.185
40	0.974	1.300	1.277
50	0.893	1.278	1.272
60	0.918	1.200	1.202
70	1.05	1.124	1.132
80	1.018	1.122	1.134
90	0.965	1.119	1.134
100	1.020	1.093	1.107

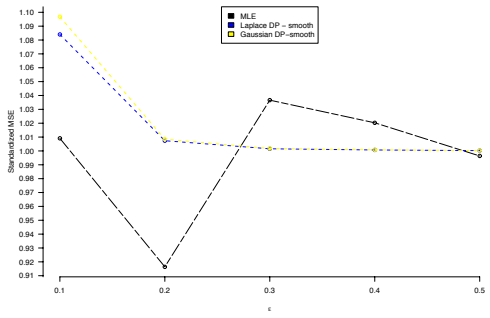
Table 4.2: Standardized MSE for MLE, the instance-dependent Algorithm 4 with Laplace noise (LapS) and with Gaussian noise (GausS) when $\epsilon = 0.1$ and $\Lambda = 10$.

ϵ	MLE	LapS	GausS
0.1	0.975	1.071	1.081
0.2	1.046	1.003	1.004
0.3	0.981	1.001	1.001
0.4	1.002	1.000	1.000
0.5	0.981	1.000	1.000

Table 4.3: Standardized MSE for MLE, the instance-dependent Algorithm 4 with Laplace noise (LapS) and with Gaussian noise (GausS) when $N = 100$ and $\Lambda = 1$.



(a) Standardized MSE controlling for N, at privacy level $\epsilon = 0.1$ and $\Lambda = 1$.



(b) Standardized MSE controlling for $\epsilon = 0.1$, with N=100 and $\Lambda = 1$.

Figure 4.1: Evaluation of the instance-dependent algorithm 4.

4.3.2 Comparison of Algorithm 4 with Algorithms 1 and 3

As we can see from Tables 4.4 and 4.5 and Figure 4.2 the instance-based Algorithm 4 always performs better than Algorithms 1 (Laplace noise with global sensitivity, GS_L) and 3 (Gaussian noise with global sensitivity, GS_G) where the noise magnitude is calibrated according to the global sensitivity. The result reflects the fact that the smooth sensitivity $S^*_{\hat{\mu},\beta}$ of the MLE is always smaller than the global sensitivity $\frac{GS_{\hat{\mu}}}{\epsilon}$. We also observe that Algorithm 3 always has the worst performance on the data. Finally, we do not provide simulation results for varying levels of Λ since we saw in Sections 3.3 and 3.8 that the performance of Algorithms 1 and 3 degrades as Λ increases.

Given these results on statistical efficiency, we anticipate a better hypothesis test performance. However, we also observed that with using the new critical values along with the sample size adjustment factors, we do well in terms of both the Type I and Type II error and the efficiency. But the advantage of the smooth sensitivity is then that there is no need for the additional sample size adjustment, and thus the cost of the study maybe more managable. This is an interesting problem to explore.

N	MLE	LapS	GausS	GS_L	GS_G
10	0.896	3.612	2.503	62.829	570.467
20	0.990	1.737	1.552	33.632	392.027
30	1.014	1.252	1.214	23.136	302.160
40	1.002	1.237	1.219	17.745	248.087
50	0.973	1.208	1.203	14.465	211.7016
60	1.038	1.180	1.184	12.259	185.393
70	0.938	1.107	1.114	10.674	165.393
80	1.018	1.087	1.096	9.480	149.639
90	1.008	1.081	1.091	8.548	136.866
100	0.930	1.058	1.067	7.80	126.283

Table 4.4: MSE of the instance-dependent algorithm 4 with Laplace noise (LapS) and Gaussian noise (GausS) and algorithms with global sensitivity with Laplace (GS_L) and Gaussian random noise (GS_G) when $\epsilon = 0.1$ and $\Lambda = 1$.

ϵ	MLE	LapS	GausS	GS_L	GS_G
0.1	0.975	1.071	1.081	7.870	127.548
0.2	1.046	1.003	1.004	2.717	32.637
0.3	0.981	1.001	1.001	1.763	15.061
0.4	1.002	1.000	1.000	1.429	8.909
0.5	0.981	1.000	1.000	1.275	6.062

Table 4.5: MSE of the instance-dependent algorithm 4 with Laplace noise (LapS) and Gaussian noise (GausS) and algorithms with global sensitivity with Laplace (GS_L) and Gaussian (GS_G) random noise when $N = 100$ and $\Lambda = 1$.

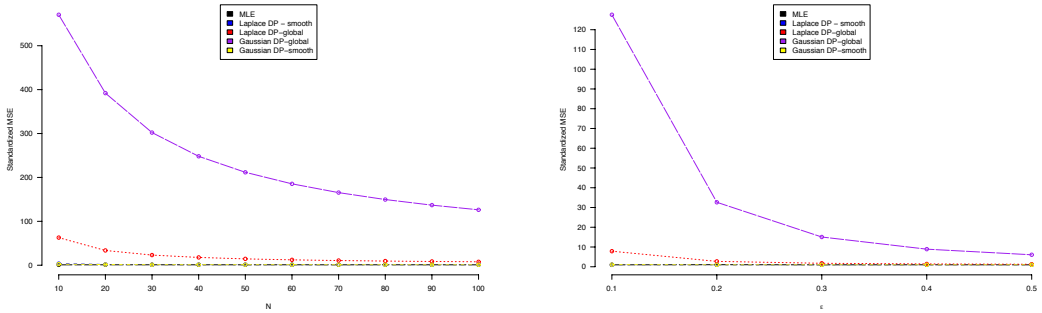


Figure 4.2: Comparison of the instance-dependent algorithm 4 and algorithms with global sensitivity.

Chapter 5

Concluding Remarks

5.1 Summary and Conclusions

In this thesis, we integrated two important areas of statistical inference, point estimation and hypothesis testing for normal random variables with differential privacy.

First, we showed that if we add Laplace noise with standard deviation $\frac{GS_{\hat{\mu}}}{\epsilon}$ to the maximum likelihood estimator of the mean, differential privacy is preserved. The statistical efficiency of the private estimator depends on the privacy parameter ϵ and on the the diameter of the parameter space Λ. A Monte Carlo simulation study showed that if $\Lambda = 1$ and the sample size is small the private estimator is efficient when ϵ increases. The simulation results also showed that for large values of Λ the private estimator is as good as the MLE provided that the sample size is sufficiently large. In general, the private estimator of the mean has the same distribution with the MLE as long as $\Lambda = o(\epsilon N)$.

Furthermore, in our work we proposed sample size adjustment factors K for computing the required sample size needed to achieve the same statistical power as the true sample size would without the infusion of noise for privacy protection. The results of the simulation studies showed no large differences in Type I and Type II error performance when $\Lambda = 1$ between the test that rejects H_0 using the non-private critical value (3.5) and the Normal-Normal test (3.6) and Normal-Laplace test. In particular, all tests gave good approximations for the Type I and Type II error rates even when there was no correction for the true sample

size and for high amount of privacy. On the other hand, with fixed $\epsilon = 0.1$ and increasing diameter Λ, the test with non-private critical value displayed inflated Type I error probabilities even when we adjusted the true sample size with the standard Normal based test. On the other hand, the simulations results showed that the Normal-Normal test and the Normal-Laplace test displayed Type I error rates near the nominal level α for large values of Λ and there was improvement in power with using the adjusted sample size.

Therefore, based on the simulation results we suggest using rejection region $\mu(X, R) > z_{1-\alpha}\sqrt{\frac{\sigma^2}{N}}$ when $\Lambda = 1$ and N is small and reject using the private critical values when Λ is large and N is also large. When N is not large, the sample size adjustment factors should be used in the addition to the private critical values. Thus when Λ is large, the best results are obtained by using both the new critical test values and the sample size adjustment factor.

Finally, we also provided an algorithm that uses the smooth sensitivity framework and demonstarted that the output of the estimator is actually better than the previous algorithms even for small sample size and high level of privacy and remains insensitive to the choice of Λ. This is a bit more complex algorithm but we do not anticipate higher computing cost than from the output perturbation algorithm.

5.2 Future Research

In this thesis, we evaluate the performance of the proposed methodology only with the simulated data. The next step would be to run additional analysis on some real-life data.

In Chapter 4, we saw that using the smooth sensitivity we could design a differentially private algorithm that is better than the output perturbation algorithms because it is more robust to the range of the parameter space Λ. The proposed method is promising for incorporating differential privacy to any parametric model with unbounded space. The question is, can we extend the smooth sensitivity framework to construct a differentially private algorithm for any parametric model where the parameter space is unbounded? Another open research question is whether there exists a hypothesis testing precedure for the normal

model and for any parametric model based on the smooth sensitivity framework with better statistical power.

In the work of Vu and Slavkovic [23] on the binomial proportion, the parameter space was bounded, and the output perturbation algorithm performed well. From our analysis, we learned that we can successfully incorporate differential privacy with utility guarantees for the maximal likelihood estimator for the normal distribution. An interesting problem to investigate would be whether the sample correlation for a random sample drawn from the bivariate normal distribution can be made differentially private. The sample correlation is bounded as it takes values in $[-1, 1]$. Would the output perturbation style algorithms from Chapter 3 work well in this setting too or would we need utilize the smooth sensitivity in order to gain a better utility ?

Appendix A

A.1 Code for Simulation Study I on statistical efficiency

```
########################################
#mean square error for the mean. effect of data size
########################################

library('normalp')
startDataSize = 100
endDataSize = 1000
stepDataSize = 100
numSteps=1 +
ceiling((endDataSize - startDataSize) / stepDataSize)
numSteps

simSize = 10000  #simulation size
mu1=0
sigma=1

Lambda = 1
alpha = 1
epsilon = .1

laplaceVar = rep(0,numSteps)
mseML = rep(0,numSteps)
```

```
mseEP = rep(0,numSteps)
mseEP1=rep(0,numSteps)
mseEP2=rep(0,numSteps)

for (j in 1:numSteps) {
dataSize = startDataSize + (j - 1)*stepDataSize
k = (dataSize^(alpha))

muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i]=mean(rnorm(1,mu1,sd=sigma/sqrt(dataSize))) #mle of the mean
muEP[i]=muML[i] + rnormp(1, mu=0, sigmap=(Lambda/(k*epsilon)),p=1)
##private mle}
laplaceVar[j] = 2 * ( Lambda / (k*epsilon) )^2
mseML[j] = sum( (muML- mu1)^2 ) / simSize
mseEP[j] = sum( (muEP- mu1)^2 ) / simSize

#standardised MSE
varML = (sigma^2)/ dataSize
mseML[j] = mseML[j] / varML
mseEP[j] = mseEP[j] /varML
}

mseML
mseEP

postscript("NormalMseByDataSize.ps")
ltys = c("22", "44", "13", "1343", "73", "2262", "12223242",
"F282", "F4448444", "224282F2", "F1")

lineWD=3
textSize = 1.5
legendSize = 1.5
```

```
axisSize = 2
symbolSize = 1.5

range = range(0,4, 1)

plot(mseML, type="o", col="darkorchid", ylim=range, axes=FALSE,
ann=FALSE, cex=symbolSize, pch=1, lty =ltys[1], lwd=lineWD)
lines(mseEP, type="o", cex=symbolSize, pch=2, lty=ltys[2],
col="lightgoldenrod", lwd=lineWD)
alphas = rep(0,numSteps)
for(i in 1:numSteps)
alphas[i] = startDataSize + (i - 1)*stepDataSize
axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,range[2],1), cex.axis=axisSize)
title(xlab="Data Size N", cex.lab=textSize)
title(ylab="Standardized MSE", cex.lab=textSize)

legend("top",c("ML estimator","DP-estimator"), cex=legendSize,
col=c("darkorchid","lightgoldenrod"), pch=c(1,2,3,6),
lty=c(ltys[1],ltys[4]), lwd=lineWD)

dev.off()

########################################
#mean square error for the mean. Effect of Lambda
########################################
library('normalp')

dataSize=100
simSize=10000
mu1=0
sigma=1
alpha = 1
epsilon = .1
startLambda = 1
```

```
endLambda = 20
stepLambda = 5
numSteps =1
+ ceiling( (endLambda - startLambda)/stepLambda)
numSteps

laplaceVar = rep(0,numSteps)
laplaceVar
mseML = rep(0,numSteps)
mseEP = rep(0,numSteps)

for (j in 1:numSteps) {
Lambda = startLambda + (j - 1)*stepLambda
k = (dataSize^(alpha))
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnormp(1, mu=0, sigmap=(Lambda/(k*epsilon)),p=1)
##private mle }
laplaceVar[j] = 2 * ( Lambda / (k*epsilon) )^2
mseML[j] = sum( (muML- mu1)^2 ) / simSize
mseEP[j] = sum( (muEP- mu1)^2 ) / simSize
#mseEP1[j]=sum( (muEP1- mu1)^2 ) / simSize
#mseEP2[j]=sum( (muEP2- mu1)^2 ) / simSize}

muEP
mseEP
varML = (sigma^2) / dataSize
mseML = mseML/varML
mseEP = mseEP/varML

mseML
```

```
mseEP

ltys = c("22", "44", "13", "1343", "73", "2262", "12223242",
"F282", "F4448444", "224282F2", "F1")

postscript("NormalMseByLambda.ps")

lineWD=3
textSize = 1.5
legendSize = 1.5
axisSize = 1.5
symbolSize = 1.5

range = range(0, 25,500)

plot(mseML, type="o", col="darkorchid", ylim=range, axes=FALSE,
ann=FALSE, cex=symbolSize, pch=1, lty =ltys[1], lwd=lineWD)
lines(mseEP, type="o", cex=symbolSize, pch=2, lty=ltys[2],
col="lightgoldenrod", lwd=lineWD)

alphas = rep(0,numSteps)
for(i in 1:numSteps)
alphas[i] =  startLambda + (i - 1)*stepLambda
axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,range[2],100), cex.axis=axisSize)
title(xlab=expression(Lambda))
title(ylab="Standardized MSE", cex.lab=textSize)

legend(1,400 ,c("ML estimator","DR-estimator"), cex=legendSize,
col=c("darkorchid","lightgoldenrod"),
pch=c(1,2), lty=c(ltys[1],ltys[2],ltys[3],ltys[4]), lwd=lineWD)

########################################
#MSE controlling the privacy parameter for
#a given sample size and Lambda=1
```

```
########################################

library('normalp')

dataSize = 100
simSize = 10000
Lambda = 1
alpha = 1
mu1=0
sigma=1

startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numSteps = 1 +
ceiling( (endEpsilon - startEpsilon) / stepEpsilon)
numSteps

laplaceVar = rep(0,numSteps)
laplaceVar
mseML = rep(0,numSteps)
mseEP = rep(0,numSteps)

for (j in 1:numSteps){
epsilon = startEpsilon + (j - 1)*stepEpsilon
k = (dataSize^(alpha))

muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(dataSize,mu1,sd=sigma))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(k*epsilon)),p=1) ##private mle }
```

```
laplaceVar[j] = 2 * ( Lambda / (k*epsilon) )^2
mseML[j] = sum( (muML- mu1)^2 ) / simSize
mseEP[j] = sum( (muEP- mu1)^2 ) / simSize}

varML = (sigma^2)/ dataSize
mseML = mseML / varML   ##Relative efficiency
mseEP = mseEP /varML

varML
mseML
mseEP

postscript("NormalMseByEpsilon.ps")

ltys = c("22", "44", "13", "1343", "73", "2262", "12223242",
"F282", "F4448444", "224282F2", "F1")
lineWD=3
textSize = 1.5
legendSize = 2
axisSize = 2
symbolSize = 3

range = range(0,mseML, mseEP)

plot(mseML, type="o", col="blue", ylim=range,
axes=FALSE, ann=FALSE, cex=symbolSize,
pch=1, lty =ltys[1], lwd=lineWD)
lines(mseEP, type="o", cex=symbolSize, pch=2,
lty=ltys[2], col="green", lwd=lineWD)

alphas = rep(0,numSteps)
for(i in 1:numSteps)
alphas[i] = startEpsilon + (i - 1)*stepEpsilon

axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
```

```
axis(2, las=1, at=seq(0,range[2],1), cex.axis=axisSize)

title(xlab=expression(epsilon),cex.lab=3)
title(ylab="Standardized MSE", cex.lab=textSize)

legend("top",c("ML estimator","DR-estimator"),
cex=legendSize,
fill=c("blue","green"))

dev.off()

##################################################
#Normal mean MSE controlling Lambda and Sample Size for fixed
#epsilon
##################################################

library('normalp')
mu1=0
sigma=1
sigma2=sigma^2
alpha = 1
epsilon = .1
startLambda = 1
endLambda = 20
stepLambda = 5
numStepsLambda=1
+ ceiling((endLambda - startLambda) /stepLambda)
numStepsLambda

startDataSize = 100
endDataSize = 1000
stepDataSize = 100
numStepsDS=1
+ceiling( endDataSize - startDataSize)/stepDataSize)
numStepsDS
```

```
varML = matrix(0,nrow=numStepsLambda,ncol=numStepsDS)
mseML = matrix(0,nrow=numStepsLambda,ncol=numStepsDS)
mseEP = matrix(0,nrow=numStepsLambda,ncol=numStepsDS)
mseEP1=matrix(0,nrow=numStepsLambda,ncol=numStepsDS)
mseEP2=matrix(0,nrow=numStepsLambda,ncol=numStepsDS)

simSize=10000

for (i in 1:numStepsLambda) {
Lambda = startLambda + (i - 1)*stepLambda
for (j in 1:numStepsDS){
dataSize=startDataSize+(j-1)*stepDataSize

muML=rep(0,simSize)
muEP =rep(0,simSize)
muEP1=rep(0,simSize)
muEP2=rep(0,simSize)

for(k in 1:simSize) {
muML [k]= mean(rnorm(1,mu1,sd=sigma/sqrt(dataSize)))
muEP[k] = muML[k] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1) ##private mle }
#laplaceVar[i,j] = 2 * ( Lambda / (k*epsilon) )^2
mseML[i,j] = sum( (muML- mu1)^2 ) / simSize
mseEP[i,j] = sum( (muEP- mu1)^2 ) / simSize

varML[i,j] = (sigma^2)/ dataSize
mseML[i,j] = mseML[i,j] / varML[i,j]
mseEP[i,j] = mseEP[i,j] /varML[i,j]}
}
mseML
mseEP
postscript('MSEnormalLambdabyN.ps')
```

```
axisSize=1
textSize=1.5
legendSize=1.5
range=range(0,mseEP)

matplot(mseEP[5,],type='l',col='red',ylim=range,ann=FALSE,
axes=FALSE)
matlines(mseEP[4,],type='l',col='green',cex=3)
matlines(mseEP[3,],type='l',col='orange',cex=3)
matlines(mseEP[2,],type='l',col='black',cex=3)
matlines(mseEP[1,],type='l',col='blue',cex=3)
alphas = rep(0,numStepsDS)
for(i in 1:numStepsDS)
alphas[i] = startDataSize + (i - 1)*stepDataSize
axis(1, at=1:numStepsDS, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,range[2],50), cex.axis=axisSize)
title(xlab="Data Size N", cex.lab=textSize)
title(ylab="Standardized MSE", cex.lab=textSize)
legend("top",c(expression(paste(Lambda==20)),
expression(paste(Lambda==15)),
expression(paste(Lambda==10)),expression(paste(Lambda==6)),
expression(paste(Lambda==1))), cex=legendSize,
col=c("red","green","orange","black","blue"),lty=1)
dev.off()
```

A.2 Code for Simulation Study II

```
#######################################################
#Adjustment Sample Size Correction when Sampling from
#Normal distribution with bounded space, known ####variance
#(Effect of epsilon)
#######################################################

#Case 1: Normal - Normal Approximation.
#The sampling distribution of the e-statistic is Normal
```

```
# Equation for computing p-quantile of GNL
solve.CVal.eq<-function(N,delta,alpha,beta,
sigma_sq,epsilon,Lambda)
{
b = Lambda/ (epsilon*N)
x1=quant.1.GNL(1-alpha, 0, sigma_sq/N,
alpha = 1/b, beta = 1/b,1)
x2=quant.1.GNL(beta, delta, sigma_sq/N,
alpha = 1/b, beta = 1/b,1)
x1 - x2
}
#To solve for the sample size based on the
# Normal Laplace Distributions
solve.CVal<-function(Nstart,Nend,delta,alpha,beta,sigma_sq,
epsilon,Lambda)
{
out<-uniroot(solve.CVal.eq,interval=c(Nstart, Nend),
delta=delta,alpha=alpha,beta=beta,
sigma_sq=sigma_sq,epsilon=epsilon, Lambda=Lambda,
tol=1e-2)
ifelse(out$f.root>-5,out$root,0)
}

#Diameter space
Lambda=1
# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu = 0
# The difference from the null hypothesis
delta = .1
# The standard deviation of the normal
```

```
sigma=1
N = (qnorm(1-alpha) + qnorm(1-beta))^2 * (sigma^2) * delta^{-2}
 #the truth sample size
N = ceiling(N)
N

startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numEpsilonSteps= 1+
ceiling((endEpsilon-startEpsilon)/stepEpsilon)

K1 = matrix(nrow = 1, ncol = numEpsilonSteps)

for (k in 1:numEpsilonSteps ){
epsilon = startEpsilon+ (k - 1)*stepEpsilon
K1[k] = .5 + .5 * sqrt(1 + (8 * delta^2*Lambda^2) /
(epsilon^2 * (qnorm(1-alpha) +
qnorm(1-beta))^2 * sigma^2) )
#the sample size correction factor
N1 = ceiling(N * K1[k])
}

K1
N1
##############################################
#Case 2: Normal-Laplace. The sampling distribution of
##############################################

# the statistic is Normal-Laplace.
#Diameter space
Lambda=1
# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
```

```
beta = .1
# The null hypothesis
mu = 0
# The difference from the null hypothesis
delta = .1
# The standard deviation of the normal
sigma=1
sigma2=sigma^2
N = (qnorm(1-alpha) + qnorm(1-beta))^2 * (sigma^2) * delta^{-2}
#the truth sample size
N = ceiling(N)
startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numEpsilonSteps = 1 +
ceiling((endEpsilon - startEpsilon)/stepEpsilon)
postfixFileName = paste("-alpha=",alpha,"-beta=",beta,"-mean=",
mu,"-delta=",delta,"-epsilonS=",startEpsilon,"
-epsilonE=",endEpsilon,"-epsilonStep=",stepEpsilon)

K2 = matrix(nrow = 1, ncol = numEpsilonSteps)
for (k in 1:numEpsilonSteps ){
epsilon = startEpsilon+ (k - 1)*stepEpsilon
K1[k] = .5 + .5 * sqrt(1 + (8 * delta^2*Lambda^2) /
(epsilon^2 * (qnorm(1-alpha)
+ qnorm(1-beta))^2 * sigma^2) )  #the sample size correction factor
N1 = ceiling(N * K1[k])
start = N
end = N1 * 2.0
N2 = solve.CVal(start,end,delta,alpha,beta,
sigma_sq=sigma2,epsilon,Lambda)
N2 = ceiling(N2)
K2[k] = N2 / N
}
```

```
K2
N2
write.table(K1,file='K1byEpsilon01')
write.table(K2,file='K2byEpsilon01')

ltys = c("22", "44", "13", "1343", "73", "2262", "12223242",
"F282",
"F4448444", "224282F2", "F1")
lineWD=3
textSize = 1.5
legendSize = 2
axisSize = 2
symbolSize = 3

range = range(0,K1, K2)

postscript('KfactorByEpsilon01.ps')
epsilon=seq(startEpsilon,endEpsilon,stepEpsilon)
plot(epsilon,K1,type="l",col="red",ann=FALSE, lwd=lineWD,
cex.axis=1.5)
lines(epsilon,K2,type="l",col="green",lwd=lineWD)
title(xlab=expression(epsilon),cex.lab=1.5)
title(ylab="Standardized MSE", cex.lab=textSize)
legend("top",c("Normal-Normal","Normal-Laplace"),
cex=legendSize, col=c("red","green"),lty=c(1,1),lwd=c(3,3))

dev.off()

#######################################################
#######################################################
#Adjustment Sample Size Correction when Sampling from Normal
#distribution with bounded space, known
#variance  (Effect of Lambda)
#################################################
#  Case 1: Normal - Normal Approximation. The sampling
```

```
#  distribution of the e-statistic is Normal
################################################

#Equation for computing p-quantile of GNL
solve.CVal.eq<-function(N,delta,alpha,beta,sigma_sq,
epsilon,Lambda){
b = Lambda / (epsilon*N)
x1=quant.1.GNL(1-alpha,0, sigma_sq/N,alpha=1/b,beta=1/b,1)
x2=quant.1.GNL(beta,delta, sigma_sq/N,alpha=1/b,beta=1/b,1)
x1 - x2}

#To solve for the sample size based on the
#Normal Laplace Distribution
solve.CVal <- function(Nstart,Nend,delta,alpha,beta,
sigma_sq,epsilon,Lambda){
out<-uniroot(solve.CVal.eq,interval=c(Nstart, Nend),delta=delta,
alpha=alpha,beta=beta,
sigma_sq=sigma_sq,epsilon=epsilon, Lambda=Lambda,
tol=1e-2)ifelse(out$f.root>-5,out$root,0)
}

#epsilon
epsilon=0.1
# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .4
# The null hypothesis
mu = 0
# The difference from the null hypothesis
delta = .1
# The standard deviation of the normal
sigma=1
N = (qnorm(1-alpha) + qnorm(1-beta))^2 * (sigma^2) * delta^{-2}
#the truth sample size
```

```
N = ceiling(N)
N

startLambda = 1
endLambda = 10
stepLambda = 1
numLambdaSteps = 1
+ ceiling( (endLambda - startLambda) / stepLambda)

K1 = matrix(nrow = 1, ncol = numLambdaSteps)

for (k in 1:numLambdaSteps ) {
Lambda = startLambda+ (k - 1)*stepLambda

K1[k] = .5 + .5 * sqrt(1 + (8 * delta^2*Lambda^2) /
(epsilon^2 * (qnorm(1-alpha)
+ qnorm(1-beta))^2 * sigma^2) )
#the sample size correction factor
N1 = ceiling(N * K1[k])
}

K1
N1

##################################################
##################################################
#Case 2: Normal-Laplace. The sampling distribution of the
#epsilon  statistic is Normal-Laplace.
##################################################

epsilon=0.1
# The confidence level alpha - Type I error
alpha = .05

# The power 1 - beta where beta - Type II error
```

```
beta = .4

# The null hypothesis
mu = 0
# The difference from the null hypothesis
delta = .1
# The standard deviation of the normal

sigma=1
sigma2=sigma^2
N = (qnorm(1-alpha) + qnorm(1-beta))^2 * (sigma^2) * delta^{-2}
 #the truth sample size
N = ceiling(N)

startLambda = 1
endLambda = 10
stepLambda = 1
numLambdaSteps = 1 +
ceiling( (endLambda - startLambda) / stepLambda)

K2 = matrix(nrow = 1, ncol = numLambdaSteps)

for (k in 1:numLambdaSteps) {
Lambda = startLambda+ (k - 1)*stepLambda

K1[k]=.5 + .5 * sqrt(1 + (8 * delta^2*Lambda^2)/
(epsilon^2 * (qnorm(1-alpha)
+ qnorm(1-beta))^2 * sigma^2) )  #the sample size correction factor
N1 = ceiling(N * K1[k])

start = N
end = N1 * 2.0
N2 = solve.CVal(start,end,delta,alpha,beta,
sigma_sq=sigma2,epsilon,Lambda)
N2 = ceiling(N2)
```

```
K2[k] = N2 / N
}

K2
N2

lineWD=3
textSize = 1.5
legendSize = 2
axisSize = 2
symbolSize = 3

postscript('KfactorByLambda04.ps')
Lambda=seq(startLambda,endLambda,stepLambda)
plot(Lambda,K1,type="l",col="red",ann=FALSE,lwd=lineWD,
cex.axis=1.5)
lines(Lambda,K2,type="l",col="green")
title(xlab=expression(Lambda),cex.lab=1.5)
title(ylab="K factor",cex.lab=1.5)
legend("top",c("Normal-Normal","Normal-Laplace"),
fill=c("red","green"),
lty=c(1,1),cex=legendSize)

dev.off()

##########################################
##########################################
# Plot of sample size adjustment factor
# K controlling both the effect of Lambda and epsilon
##########################################
##########################################
library('normalp')

# Equation for computing p-quantile of GNL
solve.CVal.eq <- function(N,delta,alpha,beta,
```

```
sigma_sq,epsilon,Lambda){
b = Lambda / (epsilon*N)
 x1 = quant.1.GNL(1-alpha, 0, sigma_sq/N, alpha = 1/b,
beta = 1/b, 1)
x2 = quant.1.GNL(beta, delta, sigma_sq/N, a
lpha = 1/b, beta = 1/b, 1)
x1 - x2}

# To solve for the sample size based on
the Normal Laplace Distributions #

solve.CVal <- function(Nstart,Nend,delta,alpha,beta,
sigma_sq,epsilon,Lambda)
 {
 out<-uniroot(solve.CVal.eq,interval=c(Nstart, Nend),
delta=delta,alpha=alpha,beta=beta,
sigma_sq=sigma_sq,epsilon=epsilon, Lambda=Lambda,
tol=1e-2) ifelse(out$f.root>-5,out$root,0)
 }

# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu = 0
# The difference from the null hypothesis
delta = .1
# The standard deviation of the normal
sigma=10
sigma2=sigma^2
N = (qnorm(1-alpha) +
qnorm(1-beta))^2 * (sigma^2) * delta^{-2}
#the truth sample size
N = ceiling(N)
```

```
startLambda = 10
endLambda = 100
stepLambda = 10
numStepsLambda = 1
+ ceiling( (endLambda - startLambda) / stepLambda)
numStepsLambda

startEpsilon = 0.1
endEpsilon = 0.5
stepEpsilon = 0.1
numStepsEpsilon = 1
+ ceiling( (endEpsilon - stepEpsilon) / stepEpsilon)
numStepsEpsilon

K1 = matrix(NA, nrow=numStepsLambda,ncol = numStepsEpsilon)
K2 = matrix(NA, nrow=numStepsLambda,ncol = numStepsEpsilon)
N1 = matrix(NA, nrow=numStepsLambda,ncol = numStepsEpsilon)
N2 = matrix(NA, nrow=numStepsLambda,ncol = numStepsEpsilon)

for (i in 1:numStepsLambda){
Lambda = startLambda + (i - 1)*stepLambda
for (j in 1:numStepsEpsilon){
epsilon=startEpsilon+(j-1)*stepEpsilon
K1[i,j] = .5 + .5 * sqrt(1 + (8 * delta^2*Lambda^2) /
(epsilon^2 * (qnorm(1-alpha) + qnorm(1-beta))^2 * sigma^2) )
 #the sample size correction factor

N1[i,j] = ceiling(N * K1[i,j])
start = N
end = N1[i,j] * 2.0
N2[i,j] = solve.CVal(start,end,delta,alpha,beta,
sigma_sq=sigma2,epsilon,Lambda)
N2[i,j] = ceiling(N2[i,j])
K2[i,j] = N2[i,j] / N  }
```

```
}

K1

axisSize=1.5
textSize=1.5
legendSize=2
postscript("Kfactor-NN-sigma=10.ps")
range=range(0,K1)

matplot(K1[,5],type='l',col='red',ylim=range,
ann=FALSE,axes=FALSE,lwd=3)
matlines(K1[,4],type='l',col='green',lwd=3)
matlines(K1[,3],type='l',col='orange',lwd=3)
matlines(K1[,2],type='l',col='black',lwd=3)
matlines(K1[,1],type='l',col='blue',lwd=3)

alphas = rep(0,numStepsLambda)
for(i in 1:numStepsLambda)
alphas[i] = 0  + (i)*stepLambda
axis(1, at=1:numStepsLambda, lab=alphas, cex.axis=axisSize)
axis(2,  at=seq(0,range[2],1), cex.axis=axisSize)
title(xlab=expression(Lambda), cex.lab=textSize)
title(ylab="K factor", cex.lab=textSize)
title("Normal-Normal sampling Distribution")
legend("topleft",c(expression(paste(epsilon==0.5)),
expression(paste(epsilon==0.4)),
expression(paste(epsilon==0.3)),
expression(paste(epsilon==0.2)),
expression(paste(epsilon==0.1))), cex=legendSize,
col=c("red","green","orange","black","blue"),lty=1,lwd=c(3,3))

dev.off()

################################################
```

```
#Plot of Normal - Laplace K factor
################################################

postscript("Kfactor-NL-sigma=4.ps")
range=range(0,K2)

matplot(K2[,5],type='l',col='red',ylim=range,
ann=FALSE,axes=FALSE,lwd=3)
matlines(K2[,4],type='l',col='green',lwd=3)
matlines(K2[,3],type='l',col='orange',lwd=3)
matlines(K2[,2],type='l',col='black',lwd=3)
matlines(K2[,1],type='l',col='blue',lwd=3)

alphas = rep(0,numStepsLambda)
for(i in 1:numStepsLambda)
alphas[i] = startLambda + (i - 1)*stepLambda
axis(1, at=1:numStepsLambda, lab=alphas, cex.axis=axisSize)
axis(2,  at=seq(1,range[2],0.2), cex.axis=axisSize)
title(xlab=expression(Lambda), cex.lab=textSize)
title(ylab="K factor", cex.lab=textSize)
title("Normal-Laplace sampling Distribution")
legend("topleft",c(expression(paste(epsilon==0.5))
,expression(paste(epsilon==0.4)),expression(paste(epsilon==0.3)),
expression(paste(epsilon==0.2)),expression(paste(epsilon==0.1))),
cex=legendSize,
col=c("red","green","orange","black","blue"),lty=1,lwd=c(3,3))

dev.off()

################################################
#Type I and Type II probabilities for Normal effect of epsilon
#size=0.05,power=1-0.1,delta=0.1,mu0=1,sigma=1,Lambda=1
# with adjusted corrected factors
################################################
```

```
library("normalp")

# Equation for computing p-quantile of GNL
solve.CVal.eq <- function(N,delta,alpha,beta,
sigma_sq,epsilon,Lambda){
b = Lambda / (epsilon*N)
x1 = quant.1.GNL(1-alpha/2, mu0, sigma_sq/N,
alpha = 1/b, beta = 1/b, 1)
x2 = quant.1.GNL(beta, mu0+delta, sigma_sq/N,
alpha = 1/b, beta = 1/b, 1)
x1 - x2
}
#To solve for the sample size based on the
#Normal Laplace Distributions

solve.CVal <- function(Nstart,Nend,delta,alpha,beta,
sigma_sq,epsilon,Lambda){
out<-uniroot(solve.CVal.eq,interval=c(Nstart, Nend),delta=delta,
alpha=alpha,beta=beta,
sigma_sq=sigma_sq,epsilon=epsilon, Lambda=Lambda,
tol=1e-2)
ifelse(out$f.root>-5,out$root,0)}

Lambda =1

# The confidence level alpha - Type I error
alpha = .05

# The power 1 - beta where beta - Type II error
beta = .1

# The null hypothesis
mu0 =0
# The difference from the null hypothesis
delta = .1
```

```
# The sd and variance of the sampling distribution
sigma =1
sigma2=sigma^2

N0 = (qnorm(1-alpha/2)
+ qnorm(1-beta))^2 * sigma2 * delta^{-2}
#non private sample size
N0 = ceiling(N0)

startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numEpsilonSteps = 1
+ceiling( (endEpsilon - startEpsilon) / stepEpsilon)
simSize = 10000

K1 = matrix(nrow = 1, ncol = numEpsilonSteps)
#K corrected factors for Normal-Normal
K2 = matrix(nrow = 1, ncol = numEpsilonSteps)
#K corrected factors for Normal-Laplace
alphaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
#alpha size for non -private size
betaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
betaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN2 = matrix(nrow = 1, ncol = numEpsilonSteps)
betaN2 = matrix(nrow = 1, ncol = numEpsilonSteps)

for (k in 1:numEpsilonSteps){
#Compute the corrected factors for each
#sampling distribution of the e-private statistic

epsilon = startEpsilon + (k - 1)*stepEpsilon

#K when Normal - Normal
```

```
K1[k] = .5
+ .5 * sqrt(1 + (8 * delta^2 * Lambda^2)/ (epsilon^2 * (qnorm(1-alpha/2)
+ qnorm(1-beta))^2 * sigma2) )
N1 = ceiling(N0 * K1[k])

#K when Normal Laplace

start = N0
end = N1 * 2.0
N2 = solve.CVal(start,end,delta,alpha,beta,
sigma_sq=sigma2,epsilon,Lambda)
N2 = ceiling(N2)
K2[k] = N2 / N0

##################################################
#Computation the type I error and type II error
#using the e-private statistic and the non - private sample size
##################################################

# N0
dataSize = N0
rejectionValue = mu0
+qnorm(1-alpha/2) * sqrt(sigma2/ dataSize)

# alpha N0
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(dataSize,mean=mu0,sd=sigma))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)}
```

```
alphaN0[k] = mean(muEP > rejectionValue)

# beta N0
mu1 = mu0 + delta
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
 #the e-private sample statistic}

betaN0[k] = 1 - mean(muEP > rejectionValue)

#alpha and beta for normal normal

# N1
dataSize = N1
rejectionValue = mu0
+qnorm(1-alpha/2) * sqrt(sigma2/ dataSize)

# alpha N1
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)
for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)}
alphaN1[k] = mean(muEP > rejectionValue)

# beta N1
mu1 = mu0 + delta
muML = rep(NA, simSize)
```

```
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
#the e-private sample statistic}
betaN1[k] = 1 - mean(muEP > rejectionValue)

#Type I & II error based on K2 for Normal-Laplace
# N2

dataSize = N2
rejectionValue = mu0 +
qnorm(1-alpha/2) * sqrt(sigma2/ dataSize)

#alpha N2
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
}
alphaN2[k] = mean(muEP > rejectionValue)

#beta N2
mu1 = mu0 + delta
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
```

```
muEP[i] = muML[i] +
rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
##the e-private sample statistic}

betaN2[k] = 1 - mean(muEP > rejectionValue) }

alphaN0
betaN0

alphaN1
betaN1

alphaN2
betaN2

##################################################
# null private sampling distributions when
#rejecting using the non -private statistics and
#without increasing the sample size
##################################################
library('normalp')

millsratio=function(x)
(1-pnorm(q=x))/dnorm(x=x);


NL.pdf=function(x,mu,sigma2,alpha,beta){
sigma=sqrt(sigma2);
z=(x-mu)/sigma;
temp1=alpha*beta/(alpha+beta);
temp2=dnorm(z);
temp3=millsratio(alpha*sigma - z)
+millsratio(beta*sigma + z)
temp=temp1*temp2*temp3}
```

```
z=seq(-3,3,0.01)
mu0=0
delta=0.1
mu1=mu0+delta
N=105
sigma2=1
Lambda=1
alpha=0.05
epsilon=0.1
sigma=sqrt( (sigma2/N)+((2*Lambda^2)/(epsilon^2*N^2)) );
#sd for Normal-Normal

NPcr=mu0 + qnorm(1-alpha/2) * sqrt(sigma2/N)

pdf('NullSampling.pdf')

par(mfrow=c(2,2))
f0null=dnorm(x=z,mean=mu0,sd=sigma)
f1null=NL.pdf(x=z,mu=mu0,sigma2=sigma2/N,
alpha=(epsilon*N)/Lambda,beta=(epsilon*N)/Lambda)
f3null=dnorm(x=z,mu0,sd=sqrt(sigma2/N))

M=max(range(f0null),range(f1null),range(f3null))
### Normal-Normal null sampling distribution
f0null=dnorm(x=z,mean=mu0,sd=sigma)
plot(z,f0null,col='blue',type='l',ylim=c(0,M),
xlab=expression(Lambda==1))
abline(v=NPcr,col='darkblue')
axis(side=1,at=NPcr,labels=round(NPcr,3),
col='darkblue',cex.axis=0.7, tck=-.01)

#Normal - Laplace sampling distributions
f1null=NL.pdf(x=z,mu=mu0,sigma2=sigma2/N,
alpha=(epsilon*N)/Lambda,beta=(epsilon*N)/Lambda)
lines(z,f1null,type='l',col='darkred',ylim=c(0,M))
```

```
# MLE sampling distribution
f3null=dnorm(x=z,mu0,sd=sqrt(sigma2/N))
lines(z,f3null,type='l',col='darkgreen',ylim=c(0,M))

z=seq(-3,3,0.01)
mu0=0
delta=0.1
mu1=mu0+delta
N=105
sigma2=1
Lambda=4
alpha=0.05
epsilon=0.1
sigma=sqrt( (sigma2/N)+((2*Lambda^2)/(epsilon^2*N^2)) );
#sd for Normal-Normal

NPcr=mu0 + qnorm(1-alpha/2) * sqrt(sigma2/N)

f0null=dnorm(x=z,mean=mu0,sd=sigma)
f1null=NL.pdf(x=z,mu=mu0,sigma2=sigma2/N,
alpha=(epsilon*N)/Lambda,beta=(epsilon*N)/Lambda)
f3null=dnorm(x=z,mu0,sd=sqrt(sigma2/N))

M=max(range(f0null),range(f1null),range(f3null))

###Normal-Normal null sampling distribution
f0null=dnorm(x=z,mean=mu0,sd=sigma)
plot(z,f0null,col='blue',type='l',ylim=c(0,M),
xlab=expression(Lambda==4))
abline(v=NPcr,col='darkblue')
 axis(side=1,at=NPcr,labels=round(NPcr,3),
col='darkblue',cex.axis=0.7, tck=-.01)

### Normal - Laplace sampling distributions
```

```
f1null=NL.pdf(x=z,mu=mu0,sigma2=sigma2/N,
alpha=(epsilon*N)/Lambda,beta=(epsilon*N)/Lambda)
lines(z,f1null,type='l',col='darkred',ylim=c(0,M))


### MLE sampling distribution
f3null=dnorm(x=z,mu0,sd=sqrt(sigma2/N))
lines(z,f3null,type='l',col='darkgreen',ylim=c(0,M))

z=seq(-3,3,0.01)
mu0=0
delta=0.1
mu1=mu0+delta
N=105
sigma2=1
Lambda=6
alpha=0.05
epsilon=0.1
sigma=sqrt( (sigma2/N)+((2*Lambda^2)/(epsilon^2*N^2)) );
#sd for Normal-Normal

NPcr=mu0 + qnorm(1-alpha/2) * sqrt(sigma2/N)

f0null=dnorm(x=z,mean=mu0,sd=sigma)
f1null=NL.pdf(x=z,mu=mu0,sigma2=sigma2/N,
alpha=(epsilon*N)/Lambda,beta=(epsilon*N)/Lambda)
f3null=dnorm(x=z,mu0,sd=sqrt(sigma2/N))

M=max(range(f0null),range(f1null),range(f3null))

#Normal-Normal null sampling distribution
f0null=dnorm(x=z,mean=mu0,sd=sigma)
plot(z,f0null,col='blue',type='l',ylim=c(0,M),
xlab=expression(Lambda==6))
abline(v=NPcr,col='darkblue')
```

```
axis(side=1,at=NPcr,labels=round(NPcr,3),
col='darkblue',cex.axis=0.7, tck=-.01)

#Normal - Laplace sampling distributions
f1null=NL.pdf(x=z,mu=mu0,sigma2=sigma2/N,
alpha=(epsilon*N)/Lambda,beta=(epsilon*N)/Lambda)
 lines(z,f1null,type='l',col='darkred',ylim=c(0,M))

# MLE sampling distribution
f3null=dnorm(x=z,mu0,sd=sqrt(sigma2/N))
lines(z,f3null,type='l',col='darkgreen',ylim=c(0,M))

z=seq(-3,3,0.01)
mu0=0
delta=0.1
mu1=mu0+delta
N=105
sigma2=1
Lambda=10
alpha=0.05
epsilon=0.1
sigma=sqrt( (sigma2/N)+((2*Lambda^2)/(epsilon^2*N^2)) );
#sd for Normal-Normal

NPcr=mu0 + qnorm(1-alpha/2) * sqrt(sigma2/N)

f0null=dnorm(x=z,mean=mu0,sd=sigma)
f1null=NL.pdf(x=z,mu=mu0,sigma2=sigma2/N,
alpha=(epsilon*N)/Lambda,beta=(epsilon*N)/Lambda)
f3null=dnorm(x=z,mu0,sd=sqrt(sigma2/N))

M=max(range(f0null),range(f1null),range(f3null))

#Normal-Normal null sampling distribution
f0null=dnorm(x=z,mean=mu0,sd=sigma)
```

```
plot(z,f0null,col='blue',type='l',ylim=c(0,M),
xlab=expression(Lambda==10))
abline(v=NPcr,col='darkblue')
axis(side=1,at=NPcr,labels=round(NPcr,3),
col='darkblue',cex.axis=0.7, tck=-.01)

#Normal - Laplace sampling distributions
f1null=NL.pdf(x=z,mu=mu0,sigma2=sigma2/N,
alpha=(epsilon*N)/Lambda,beta=(epsilon*N)/Lambda)
lines(z,f1null,type='l',col='darkred',ylim=c(0,M))

# MLE sampling distribution
f3null=dnorm(x=z,mu0,sd=sqrt(sigma2/N))
lines(z,f3null,type='l',col='darkgreen',ylim=c(0,M))

dev.off();
```

A.3 Code for Simulation Study III on new hypothesis tests

```
#######################################################
#######################################################
#Power and size for Normal rejecting using
#the private rejection critical value - effect of epsilon
#######################################################
library("normalp")

# Equation for computing p-quantile of GNL
solve.CVal.eq <- function(N,delta,alpha,beta,
sigma_sq,epsilon,Lambda){
b = Lambda / (epsilon*N)
x1 = quant.1.GNL(1-alpha, mu0, sigma_sq/N,
alpha = 1/b, beta = 1/b, 1)
```

```
x2 = quant.1.GNL(beta, mu0+delta, sigma_sq/N,
alpha = 1/b, beta = 1/b, 1)
x1 - x2}

# To solve for the sample size based
on the Normal Laplace Distributions

solve.CVal <- function(Nstart,Nend,delta,alpha,
beta,sigma_sq,epsilon,Lambda){
out<-uniroot(solve.CVal.eq,interval=c(Nstart, Nend),
delta=delta,alpha=alpha,beta=beta,
sigma_sq=sigma_sq,epsilon=epsilon, Lambda=Lambda,
tol=1e-2)
ifelse(out$f.root>-5,out$root,0)}

Lambda = 1

# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu0 =0
# The difference from the null hypothesis
delta = .1
# The sd and variance of the sampling distribution
sigma =1
sigma2=sigma^2
N0 = (qnorm(1-alpha)
+ qnorm(1-beta))^2 * sigma2 * delta^{-2}
#non private sample size
N0 = ceiling(N0)

startEpsilon = .1
endEpsilon = .5
```

```
stepEpsilon = .1
numEpsilonSteps = 1
+ ceiling( (endEpsilon - startEpsilon) / stepEpsilon)
simSize = 10000

alphaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
#alpha size for non -private size
betaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
betaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN2 = matrix(nrow = 1, ncol = numEpsilonSteps)
betaN2 = matrix(nrow = 1, ncol = numEpsilonSteps)

for (k in 1:numEpsilonSteps ) {
epsilon = startEpsilon + (k - 1)*stepEpsilon

#Compute the type I error and type II error
#using the e-private statistic and the non - private sample size

# N0
dataSize = N0
rejectionValue = mu0 +
qnorm(1-alpha) * sqrt(sigma2/ dataSize)

# alpha N0

mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)
for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN0[k] = mean(muEP > rejectionValue)
```

```
#beta N0
mu1 = mu0 + delta

muML = rep(NA, simSize)
 muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu1,
sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)
##the e-private sample statistic}
betaN0[k] = 1 - mean(muEP > rejectionValue)

#alpha and beta for normal normal

dataSize = N0
rejectionValue = mu0 +
qnorm(1-alpha) * sqrt(sigma2/ dataSize+
(2*Lambda^2)/(epsilon^2*dataSize^2))

# alpha N1
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)}
alphaN1[k] = mean(muEP > rejectionValue)

#beta N1
mu1 = mu0 + delta
```

```
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
#the e-private sample statistic}
betaN1[k] = 1 - mean(muEP > rejectionValue)

#Type I & II error  Normal-Laplace

dataSize = N0
b = Lambda / (epsilon*dataSize)
rejectionvalue=quant.1.GNL(1-alpha, mu=mu0,
sigma2/dataSize, alpha = 1/b, beta = 1/b, 1)
#rejectionValue = mu0
+ qnorm(1-alpha) * sqrt(sigma2/ dataSize)

# alpha N2
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN2[k] = mean(muEP > rejectionvalue)

# beta N2
mu1 = mu0 + delta
muML = rep(NA, simSize)
muEP = rep(NA, simSize)
```

```
for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
#the e-private sample statistic}

betaN2[k] = 1 - mean(muEP > rejectionvalue)}

alphaN0
betaN0

alphaN1
betaN1

alphaN2
betaN2

##############################################
#Power and size for Normal controlling Lambda
#with private rejection value without adjusting for the noise
##############################################

library("normalp")

# Equation for computing p-quantile of GNL

solve.CVal.eq <- function(N,delta,alpha,beta,
sigma_sq,epsilon,Lambda){
b = Lambda / (epsilon*N)
x1 = quant.1.GNL(1-alpha/2, mu0, sigma_sq/N,
alpha = 1/b, beta = 1/b, 1)
x2 = quant.1.GNL(beta, mu0+delta, sigma_sq/N,
alpha = 1/b, beta = 1/b, 1)
x1 - x2}
```

```
# To solve for the sample size based on the
#Normal Laplace Distributions #

solve.CVal <- function(Nstart,Nend,delta,
alpha,beta,sigma_sq,epsilon,Lambda){
out<-uniroot(solve.CVal.eq,interval=
c(Nstart, Nend),delta=delta,alpha=alpha,beta=beta,s
igma_sq=sigma_sq,epsilon=epsilon, Lambda=Lambda,
tol=1e-2)
ifelse(out$f.root>-5,out$root,0)}

epsilon = 0.1
#The confidence level alpha - Type I error
alpha = .05
#The power 1 - beta where beta - Type II error
beta = .1
#The null hypothesis
mu0 =0
# The difference from the null hypothesis
delta = .1
# The sd and variance of the sampling distribution
sigma =1
sigma2=sigma^2
N0 = (qnorm(1-alpha/2)
+ qnorm(1-beta))^2 * sigma2 * delta^{-2}
#non private sample size
N0 = ceiling(N0)

startLambda = 1
endLambda = 10
stepLambda = 1
numLambdaSteps = 1
+ ceiling( (endLambda - startLambda) / stepLambda)
simSize = 10000
```

```
alphaN0 = matrix(nrow = 1, ncol = numLambdaSteps)
#alpha size for non -private size
betaN0 = matrix(nrow = 1, ncol = numLambdaSteps)
alphaN1 = matrix(nrow = 1, ncol = numLambdaSteps)
betaN1 = matrix(nrow = 1, ncol = numLambdaSteps)
alphaN2 = matrix(nrow = 1, ncol = numLambdaSteps)
betaN2 = matrix(nrow = 1, ncol = numLambdaSteps)

for (k in 1:numLambdaSteps ){
Lambda = startLambda + (k - 1)*stepLambda
#Compute the type I error and type II error
#using the e-private statistic and the non - private sample size

#N0
dataSize = N0
rejectionValue = mu0
+qnorm(1-alpha) * sqrt(sigma2/ dataSize)

# alpha N0
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN0[k] = mean(muEP > rejectionValue)

#beta N0
mu1 = mu0 + delta
muML = rep(NA, simSize)
muEP = rep(NA, simSize)
```

```
for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
 muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
##the e-private sample statistic}
betaN0[k] = 1 - mean(muEP > rejectionValue)

#alpha and beta for normal normal

dataSize = N0
rejectionValue = mu0 +
qnorm(1-alpha/2) * sqrt(sigma2/ dataSize
+(2*Lambda^2)/(epsilon^2*dataSize^2))

# alpha
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN1[k] = mean(muEP > rejectionValue)

# beta
dataSize=N0
rejectionValue=mu0 +
qnorm(1-alpha/2) * sqrt(sigma2/ dataSize+
(2*Lambda^2)/(epsilon^2*dataSize^2))

mu1 = mu0 + delta
muML = rep(NA, simSize)
```

```
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,
sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)
#the e-private sample statistic}

betaN1[k] = 1 - mean(muEP > rejectionValue)

#Type I & II error for Normal-Laplace

dataSize = N0
b = Lambda / (epsilon*dataSize)
rejectionValue=quant.1.GNL(1-alpha/2, mu=mu0,
sigma2/dataSize, alpha = 1/b, beta = 1/b, 1)

# alpha
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN2[k] = mean(muEP > rejectionValue)

# beta
dataSize=N0
b = Lambda / (epsilon*dataSize)
rejectionValue=quant.1.GNL(1-alpha/2, mu=mu0,
sigma2/dataSize, alpha = 1/b, beta = 1/b, 1)
```

```
mu1 = mu0 + delta

muML = rep(NA, simSize)
muEP = rep(NA, simSize)
for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)
#the e-private  sample statistic}

betaN2[k] = 1 - mean(muEP > rejectionValue)
}

alphaN0
betaN0

alphaN1
betaN1

alphaN2
betaN2

##############################################
#Power and size for Normal controlling Lambda with
#private rejection value using adjusted sample size
# corrected factor to increase power.
##############################################

library("normalp")

# Equation for computing p-quantile of GNL
solve.CVal.eq <- function(N,delta,alpha,beta
,sigma_sq,epsilon,Lambda){
b = Lambda / (epsilon*N)
```

```
x1 = quant.1.GNL(1-alpha/2, mu0,
sigma_sq/N, alpha = 1/b, beta = 1/b, 1)
x2 = quant.1.GNL(beta, mu0+delta,
sigma_sq/N, alpha = 1/b, beta = 1/b, 1)
x1 - x2}

#To solve for the sample size based on the
#Normal Laplace Distributions #

solve.CVal <- function(Nstart,Nend,delta,
alpha,beta,sigma_sq,epsilon,Lambda){
out<-uniroot(solve.CVal.eq,interval=
c(Nstart, Nend),delta=delta,alpha=alpha,
beta=beta,sigma_sq=sigma_sq,
epsilon=epsilon, Lambda=Lambda,
tol=1e-2)
ifelse(out$f.root>-5,out$root,0)}

epsilon = 0.1
# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu0 =0
# The difference from the null hypothesis
delta = .1
# The sd and variance of the sampling distribution
sigma =1
sigma2=sigma^2
N0 = (qnorm(1-alpha/2)
+ qnorm(1-beta))^2 * sigma2 * delta^{-2}
#non private sample size
N0 = ceiling(N0)
```

```
startLambda = 1
endLambda = 10
stepLambda = 1
numLambdaSteps = 1
+ ceiling( (endLambda - startLambda) / stepLambda)
simSize = 10000

K1 = matrix(nrow = 1, ncol = numLambdaSteps)
#K corrected factors for Normal-Normal
K2 = matrix(nrow = 1, ncol = numLambdaSteps)
#K corrected factors for Normal-
alphaN0 = matrix(nrow = 1, ncol = numLambdaSteps)
 #alpha size for non -private size
betaN0 = matrix(nrow = 1, ncol = numLambdaSteps)
alphaN1 = matrix(nrow = 1, ncol = numLambdaSteps)
betaN1 = matrix(nrow = 1, ncol = numLambdaSteps)
alphaN2 = matrix(nrow = 1, ncol = numLambdaSteps)
betaN2 = matrix(nrow = 1, ncol = numLambdaSteps)

for (k in 1:numLambdaSteps ){
Lambda = startLambda + (k - 1)*stepLambda
#K when Normal - Normal

K1[k] = .5 + .5 * sqrt(1 +
(8 * delta^2 * Lambda^2)/
(epsilon^2 * (qnorm(1-alpha/2)
+ qnorm(1-beta))^2 * sigma2) )
N1 = ceiling(N0 * K1[k])

start = N0
end = N1 * 2.0
N2 = solve.CVal(start,end,delta,alpha,beta,
sigma_sq=sigma2,epsilon,Lambda)
N2 = ceiling(N2)
K2[k] = N2 / N0
```

```
#Compute the type I error and type II
#error using the e-private statistic and
#the non - private sample size

#N0
dataSize = N0
rejectionValue = mu0 +
qnorm(1-alpha/2) * sqrt(sigma2/ dataSize)

#alpha N0
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,
sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN0[k] = mean(muEP > rejectionValue)

mu1 = mu0 + delta
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,
sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)
##the e-private sample statistic}

betaN0[k] = 1 - mean(muEP > rejectionValue)
```

```
#alpha and beta for normal normal

dataSize = N1
rejectionValue = mu0 +
qnorm(1-alpha/2) * sqrt(sigma2/ dataSize
+(2*Lambda^2)/(epsilon^2*dataSize^2))

# alpha
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN1[k] = mean(muEP > rejectionValue)

#beta
dataSize=N1
rejectionValue=mu0 +
qnorm(1-alpha/2) *
 sqrt(sigma2/ dataSize
+(2*Lambda^2)/(epsilon^2*dataSize^2))

mu1 = mu0 + delta
muML = rep(NA, simSize)
 muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)
```

```
 ##the e-private sample statistic
}

betaN1[k] = 1 - mean(muEP > rejectionValue)

#Type I & II error for Normal-Laplace

dataSize = N2
b = Lambda / (epsilon*dataSize)
rejectionValue=quant.1.GNL(1-alpha/2, mu=mu0,
sigma2/dataSize, alpha = 1/b, beta = 1/b, 1)

#alpha
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
 rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN2[k] = mean(muEP > rejectionValue)

#beta
dataSize=N2
b = Lambda / (epsilon*dataSize)
rejectionValue=quant.1.GNL(1-alpha/2, mu=mu0,
sigma2/dataSize, alpha = 1/b, beta = 1/b, 1)

mu1 = mu0 + delta

muML = rep(NA, simSize)
muEP = rep(NA, simSize)
```

```
for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
##the e-private   sample statistic}
betaN2[k] = 1 - mean(muEP > rejectionValue)
}

alphaN0
betaN0

alphaN1
betaN1

alphaN2
betaN2
```

A.4 Code for Simulation Study IV with Gaussian noise

```
##############################################
##############################################
#Power and size for Normal rejecting using the private
#rejection critical value - effect of epsilon
#with unknown variance, adjusting for the sample size.
##############################################

library("normalp")
Lambda = 1

# The confidence level alpha - Type I error
alpha = .05
```

```
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu0 =0
# The difference from the null hypothesis
delta = .1
N0=857
sigma=1
data=rnorm(N0,mean=mu0,sd=sigma)
s2=var(data)
sigma=s2
startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numEpsilonSteps = 1
+ ceiling( (endEpsilon - startEpsilon) / stepEpsilon)
simSize = 10000

K1=matrix(nrow=1,ncol=numEpsilonSteps)
alphaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
#alpha size for non -private size
betaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
betaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)

for (k in 1:numEpsilonSteps ) {
epsilon = startEpsilon + (k - 1)*stepEpsilon
K1[k] = .5 + .5 * sqrt(1 + (8 * delta^2 * Lambda^2)/
(epsilon^2 * (qnorm(1-alpha) + qnorm(1-beta))^2 * sigma2) )
N1 = ceiling(N0 * K1[k])

##########################################
##########################################
#Compute the type I error and type II error using the
#e-private statistic and the non - private sample size
```

```
#and the non private critical value.
##########################################
# N1
dataSize = N1

#NormalNormal

rejectionValue = mu0 +
qnorm(1-alpha)*sqrt(s2/ dataSize+
(2*Lambda^2)/(epsilon^2*dataSize^2))

#alpha N1
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN1[k] = mean(muEP > rejectionValue)

# beta N1

mu1 = mu0 + delta

rejectionValue = mu0 +
qnorm(1-alpha)*sqrt(s2/ dataSize+
(2*Lambda^2)/(epsilon^2*dataSize^2))
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
 muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
```

```
 muEP[i] = muML[i]
+rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
 ##the e-private sample statistic}

betaN1[k] = 1 - mean(muEP > rejectionValue)

#Type I & II error  Normal

dataSize = N0
rejectionValue = mu0
+ qt(1-alpha,df=dataSize-1)*sqrt(var(data)/dataSize)

# alpha N0
mean = mu0

 muML = rep(NA, simSize)
 muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN0[k] = mean(muEP > rejectionValue)

#beta N0

dataSize = N0
rejectionValue = mu0 +
qt(1-alpha,df=dataSize-1)*sqrt(var(data)/dataSize)
mu1 = mu0 + delta

muML = rep(NA, simSize)
muEP = rep(NA, simSize)
```

```
for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
#the e-private  sample statistic}

betaN0[k] = 1 - mean(muEP > rejectionValue) }

alphaN1
betaN1

alphaN0
betaN0

########################################

#Power and size for Normal rejecting
#using the private rejection critical value -
#effect of epsilon with unknown variance,
#adjusting for the sample size.
########################################

library("normalp")
Lambda = 1
# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu0 =0
# The difference from the null hypothesis
delta = .1
N0=857
sigma=1
data=rnorm(N0,mean=mu0,sd=sigma)
```

```
s2=var(data)
sigma=s2

startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numEpsilonSteps = 1 +
ceiling( (endEpsilon - startEpsilon) / stepEpsilon)
simSize = 10000

K1=matrix(nrow=1,ncol=numEpsilonSteps)
alphaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
 ##alpha size for non -private size
betaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
betaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)

for (k in 1:numEpsilonSteps){
epsilon = startEpsilon + (k - 1)*stepEpsilon

K1[k] = .5 + .5 * sqrt(1 + (8 * delta^2 * Lambda^2)/
(epsilon^2 * (qnorm(1-alpha) + qnorm(1-beta))^2 * sigma2) )
N1 = ceiling(N0 * K1[k])

#Compute the type I error and type II error
#using the e-private statistic and the non - private
#sample size and the non private critical value.

#N1
dataSize = N1

#NormalNormal

rejectionValue = mu0 +
qnorm(1-alpha)*sqrt(s2/ dataSize+
```

```
(2*Lambda^2)/(epsilon^2*dataSize^2))

#alpha N1

mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1) }

alphaN1[k] = mean(muEP > rejectionValue)

# beta N1

mu1 = mu0 + delta

rejectionValue = mu0 +
qnorm(1-alpha)*sqrt(s2/ dataSize+
(2*Lambda^2)/(epsilon^2*dataSize^2))

muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+ rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
#the e-private sample statistic}

betaN1[k] = 1 - mean(muEP > rejectionValue)

########################################
```

```
#Type I & II error  Normal

dataSize = N0
rejectionValue = mu0 +
qt(1-alpha,df=dataSize-1)*sqrt(var(data)/dataSize)

# alpha N0
mean = mu0

muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i]
+rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alphaN0[k] = mean(muEP > rejectionValue)

# beta N0

dataSize = N0
rejectionValue = mu0 +
qt(1-alpha,df=dataSize-1)*sqrt(var(data)/dataSize)
mu1 = mu0 + delta
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
##the e-private sample statistic}

betaN0[k] = 1 - mean(muEP > rejectionValue)
```

```
}

alphaN1
betaN1

alphaN0
betaN0

########################################
########################################
#Power and size for Normal rejecting using the
#private rejection critical value - effect of
#epsilon with unknown variance and adjusting the sample size.
########################################
library("normalp")

epsilon = 0.1
# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu0 =1
# The difference from the null hypothesis
delta = .1
sigma2=1
sigma=sqrt(sigma2)
N0=857
data=rnorm(N0,mean=mu0,sd=sigma)
s2=var(data)
S=sqrt(s2)

startLambda = 1
endLambda = 10
stepLambda = 1
```

```
numLambdaSteps = 1
+ ceiling( (endLambda - startLambda) / stepLambda)
simSize = 10000

K1=matrix(nrow=1,ncol=numLambdaSteps)
alphaN0 = matrix(nrow = 1, ncol = numLambdaSteps)
##alpha size for non -private size
betaN0 = matrix(nrow = 1, ncol = numLambdaSteps)
# alpha1N0 = matrix(nrow = 1, ncol = numLambdaSteps)
# beta1N0 = matrix(nrow = 1, ncol = numLambdaSteps)
alpha2N0 = matrix(nrow = 1, ncol = numLambdaSteps)
beta2N0 = matrix(nrow = 1, ncol = numLambdaSteps)

for (k in 1:numLambdaSteps ){
Lambda = startLambda + (k - 1)*stepLambda
K1[k] = .5 + .5 * sqrt(1 + (8 * delta^2 * Lambda^2)/
(epsilon^2 * (qnorm(1-alpha) + qnorm(1-beta))^2 * sigma2) )
N1 = ceiling(N0 * K1[k])

########################################
########################################
#Compute the type I error and type II
#error using the e-private statistic and the
#non - private sample size and the non private critical value.
########################################
########################################
# N0
dataSize = N1
rejectionValue = mu0 +
qnorm(1-alpha)* sqrt(s2/ dataSize+
(2*Lambda^2)/(epsilon^2*dataSize^2))
# rejectionValue = mu0 +
qt(1-alpha,df=dataSize-1)*sqrt(var(data)/dataSize)

# alpha N0
```

```
mean = mu0
muML = rep(NA, simSize)
 muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
}
alphaN0[k] = mean(muEP > rejectionValue)

# beta N0

dataSize=N1
rejectionValue = mu0 +
qnorm(1-alpha)* sqrt(s2/ dataSize
+(2*Lambda^2)/(epsilon^2*dataSize^2))
mu1 = mu0 + delta

muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
##the e-private sample statistic }

betaN0[k] = 1 - mean(muEP > rejectionValue)

#####################################################
dataSize = N1
rejectionValue = mu0
+ qnorm(1-alpha)*sqrt(s2/dataSize)
```

```
# alpha N1
mean = mu0

muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)}

alpha2N0[k] = mean(muEP > rejectionValue)

# beta N1
dataSize=N1
rejectionValue = mu0 + qnorm(1-alpha)*sqrt(s2/dataSize)

mu1 = mu0 + delta

muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnormp(1, mu=0, sigmap=(Lambda/(dataSize*epsilon)),p=1)
#the e-private     sample statistic }

beta2N0[k] = 1 - mean(muEP > rejectionValue) }

alphaN0
betaN0

alpha2N0
beta2N0
```

A.5 Code for Simulation Study V

```
########################################
########################################
#Mean square error of the DP mle of mu
# with Gaussian Noise-effect of datasize
########################################

library('normalp')

simSize = 10000
Lambda = 1
mu1=0
sigma=1
epsilon=0.1


startDataSize= 10
endDataSize = 1000
stepDataSize = 100
numSteps = 1
+ ceiling( (endDataSize - startDataSize) / stepDataSize)
numSteps

noiseVar = rep(0,numSteps)
noiseVar
mseML = rep(0,numSteps)
mseEP = rep(0,numSteps)
mseEP1 = rep(0,numSteps)

for (j in 1:numSteps){
dataSize = startDataSize + (j - 1)*stepDataSize
```

```
delta=1/dataSize

muML = rep(NA, simSize)
muEP = rep(NA, simSize)
muEP1 = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(dataSize,mu1,sd=sigma))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
#gaussian private mle
muEP1[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1) ##laplace private mle }

noiseVar[j]=((Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))^2
mseML[j] = sum( (muML- mu1)^2 ) / simSize
mseEP[j] = sum( (muEP- mu1)^2 ) / simSize
mseEP1[j] = sum( (muEP1- mu1)^2 ) / simSize

varML = (sigma^2)/ dataSize
mseML[j] = mseML[j] / varML   ##Relative efficiency
mseEP[j] = mseEP[j] /varML
mseEP1[j] = mseEP1[j] /varML
}
varML
mseML
mseEP
mseEP1

ltys = c("22", "44", "13", "1343", "73", "2262", "12223242",
"F282", "F4448444", "224282F2", "F1")

postscript("MseByDataSizeG.ps")

lineWD=3
```

```
textSize = 2
legendSize = 1.5
axisSize = 2
symbolSize = 1.5

range = range(0,mseML, mseEP)

plot(mseML, type="o", col="darkorchid", ylim=range,
axes=FALSE, ann=FALSE, cex=symbolSize,
pch=1, lty =1, lwd=lineWD)
lines(mseEP, type="o", cex=symbolSize, pch=2, lty=2,
col="lightgoldenrod", lwd=lineWD)
lines(mseEP1, type="o", cex=symbolSize, pch=2, lty=3,
col="green", lwd=lineWD)

alphas = rep(0,numSteps)
for(i in 1:numSteps)
alphas[i] = startDataSize + (i - 1)*stepDataSize
axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,range[2],5), cex.axis=axisSize)
title(xlab=expression(N))
title(ylab="Standardized MSE", cex.lab=textSize)
legend("topright",c("ML estimator","Gaussian-DP-estimator",
"Laplace-DP-estimator"), cex=legendSize,
fill=c("darkorchid","lightgoldenrod",'green'),lwd=c(3,3,3))

dev.off()

########################################
########################################
#Mean square error of the DP mle of mu
# with Gaussian Noise-effect of Lambda
########################################
########################################
dataSize = 1000
```

```
simSize = 10000
delta=1/dataSize
mu1=0
sigma=1
epsilon=0.1
startLambda = 1
endLambda = 10
stepLambda = 1
numSteps = 1
+ ceiling( (endLambda - startLambda / stepLambda))
numSteps
noiseVar = rep(0,numSteps)
noiseVar
mseML = rep(0,numSteps)
mseEP = rep(0,numSteps)
mseEP1 = rep(0,numSteps)

for (j in 1:numSteps) {
Lambda = startLambda+ (j - 1)*stepLambda
N = dataSize
muML = rep(NA, simSize)
muEP = rep(NA, simSize)
muEP1 = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(N,mu1,sd=sigma))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(N*epsilon)))
#Gaussian private mle
muEP1[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(dataSize*epsilon)),p=1)
##laplace private mle}

noiseVar[j] =  ((Lambda*(sqrt(2*log(1/delta)))/(N*epsilon)))^2
mseML[j] = sum( (muML- mu1)^2 ) / simSize
```

```
mseEP[j] = sum( (muEP- mu1)^2 ) / simSize
mseEP1[j] = sum( (muEP1- mu1)^2 ) / simSize}

varML = (sigma^2)/ dataSize
mseML = mseML / varML   ##Relative efficiency
mseEP = mseEP /varML
mseEP1 = mseEP1 /varML

varML
mseML
mseEP
mseEP1

postscript("MseByLambdaG.ps")

lineWD=3
textSize = 1.5
legendSize = 1.5
axisSize = 2
symbolSize = 1.5


range = range(0,mseEP1, mseEP)

plot(mseML, type="o", col="darkorchid", ylim=range,
axes=FALSE, ann=FALSE,
cex=symbolSize, pch=1, lty =1, lwd=lineWD)
lines(mseEP, type="o", cex=symbolSize, pch=2, lty=2,
col="lightgoldenrod", lwd=lineWD)
lines(mseEP1, type="o", cex=symbolSize, pch=2, lty=3,
col="green", lwd=lineWD)

alphas = rep(0,numSteps)
for(i in 1:numSteps)
alphas[i] = startLambda+ (i - 1)*stepLambda
```

```
axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,range[2],5), cex.axis=axisSize)
title(xlab=expression(Lambda))
title(ylab="Standardized MSE", cex.lab=textSize)
legend("topleft",c("ML estimator","Gaussian-DP-estimator",
"Laplace-DP-estimator"), cex=legendSize,
fill=c("darkorchid","lightgoldenrod","green"))

dev.off()
########################################
#Mean square error of the DP mle of mu with
#Gaussian Noise-effect of epsilon
########################################
dataSize = 100
simSize = 10000
Lambda = 1
mu1=0
sigma=1
delta=1/dataSize
alpha=1


startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numSteps = 1
+ceiling( (endEpsilon - startEpsilon) / stepEpsilon))
numSteps

noiseVar = rep(0,numSteps)
noiseVar
mseML = rep(0,numSteps)
mseEP = rep(0,numSteps)
mseEP1 = rep(0,numSteps)
```

```
for (j in 1:numSteps) {
epsilon = startEpsilon + (j - 1)*stepEpsilon
N = dataSize
k = (dataSize^(alpha))

muML = rep(NA, simSize)
muEP = rep(NA, simSize)
muEP1=rep(NA,simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(N,mu1,sd=sigma))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(N*epsilon)))
#private mle
muEP1[i] = muML[i] + rnormp(1, mu=0,
sigmap=(Lambda/(k*epsilon)),p=1) ##private mle}

noiseVar[j] =  ((Lambda*(sqrt(2*log(1/delta)))/(N*epsilon)))^2
mseML[j] = sum( (muML- mu1)^2 ) / simSize
mseEP[j] = sum( (muEP- mu1)^2 ) / simSize
mseEP1[j] = sum( (muEP1- mu1)^2 ) / simSize
}
varML = (sigma^2)/ dataSize
mseML = mseML / varML
#Relative efficiency
mseEP = mseEP /varML
mseEP1=mseEP1/varML

varML
mseML
mseEP
mseEP1

postscript("MseByEpsilonG.ps")
```

```
ltys = c("22", "44", "13", "1343", "73", "2262", "12223242",
 "F282", "F4448444", "224282F2", "F1")
lineWD=3
textSize = 1.5
legendSize = 1.5
axisSize = 2
symbolSize = 1.5


range = range(0,mseML, mseEP)
plot(mseML, type="o", col="darkorchid", ylim=range,
axes=FALSE, ann=FALSE, cex=symbolSize,
pch=1, lty =1, lwd=lineWD)
lines(mseEP, type="o", cex=symbolSize, pch=2, lty=2,
col="lightgoldenrod", lwd=lineWD)
lines(mseEP1, type="o", cex=symbolSize, pch=2, lty=3,
col="green", lwd=lineWD)
alphas = rep(0,numSteps)
for(i in 1:numSteps)
alphas[i] = startEpsilon + (i - 1)*stepEpsilon
axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,range[2],1), cex.axis=axisSize)
title(xlab=expression(epsilon))
title(ylab="Standardized MSE", cex.lab=textSize)
legend("topright",c("ML estimator","Gaussian-DP-estimator",
"Laplace-DP-estimator"), cex=legendSize,
fill=c("darkorchid","lightgoldenrod","green"))

dev.off()

################################################################
#Adjustement sample size factors for Gaussian
#added noise with respect  to Lambda for epsilon=0.1
################################################################
```

```
K.factor=function(x,dataSize,Lambda,alpha,beta,epsilon,gamma){
za=qnorm(1-alpha)
zb=qnorm(1-beta)
x^2-dataSize*x+
(-2*Lambda^2*(za+zb)^2*log(x))/(epsilon^2*gamma^2)}
root.Kfactor=function(start,end,dataSize,
Lambda,alpha,beta,epsilon,gamma){
r=uniroot(K.factor,interval=c(start,end),
dataSize=dataSize,Lambda=Lambda,alpha=alpha,
beta=beta,epsilon=epsilon,gamma=gamma,tol=1e-2)
# ifelse(r$f.root>-5,r$root,0)}

alpha=0.05
beta=0.1
epsilon=0.1
gamma=0.1
sigma=1
N = (qnorm(1-alpha) +
qnorm(1-beta))^2 * (sigma^2) * gamma^{-2}
N = ceiling(N)
startLambda = 1
endLambda = 10
stepLambda = 1
numLambdaSteps = 1
+ ceiling( (endLambda - startLambda) / stepLambda)

K2 = matrix(nrow = 1, ncol = numLambdaSteps)

for (k in 1:numLambdaSteps ) {
Lambda = startLambda + (k - 1)*stepLambda
dataSize=N
start=dataSize
end=dataSize*1000

N2=root.Kfactor(start,end,dataSize,
```

```
Lambda,alpha,beta,epsilon,gamma)$root
N2=ceiling(N2)
K2[k]=N2/dataSize}
K2
################################################
################################################
#Type I and Type II probabilities for adding Normal noise.
#Effect of epsilon  size=0.05, power=1-0.1,delta=0.1,mu0=1,
#sigma=1,Lambda=1 with adjusted corrected factors
################################################
Lambda = 1
# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu0 =0
# The difference from the null hypothesis
gamma = .1
# The sd and variance of the sampling distribution
sigma =1
sigma_sq=sigma^2

N0 = (qnorm(1-alpha) +
qnorm(1-beta))^2 * sigma^2 * gamma^{-2}
#non private sample size
N0 = ceiling(N0)

startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numEpsilonSteps = 1
+ ceiling( (endEpsilon - startEpsilon) / stepEpsilon)
simSize = 10000
```

```
K1 = matrix(nrow = 1, ncol = numEpsilonSteps)
#K corrected factor
alphaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
##alpha size for non -private size
betaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
betaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)

for (k in 1:numEpsilonSteps) {
#Compute the corrected factors for each
#sampling distribution of the e-private statistic

epsilon = startEpsilon + (k - 1)*stepEpsilon

dataSize=N0
start=dataSize
end=dataSize*1000
N1=root.Kfactor(start,end,dataSize,
Lambda,alpha,beta,epsilon,gamma)$root
N1=ceiling(N1)
K1[k]=N1/dataSize

dataSize = N0
rejectionValue = mu0 + qnorm(1-alpha) * sqrt(sigma_sq/ dataSize)
delta=1/dataSize

# alpha N0
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)
for(i in 1:simSize){
muML[i] = mean(rnorm(dataSize,mean=mu0,sd=sigma))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle}
```

```
alphaN0[k] = mean(muEP > rejectionValue)

# beta N0
mu1 = mu0 + gamma
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

 for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))
/(dataSize*epsilon))) ##gaussian private mle }
betaN0[k] = 1 - mean(muEP > rejectionValue)

#alpha and beta for correction

dataSize = N1
rejectionValue = mu0 +
qnorm(1-alpha) * sqrt(sigma_sq/ dataSize)
delta=1/dataSize

# alpha N1
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)
for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,
sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
#gaussian private mle }

alphaN1[k] = mean(muEP > rejectionValue)
```

```
# beta N1
mu1 = mu0 + gamma
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle }

betaN1[k] = 1 - mean(muEP > rejectionValue) }

alphaN0
betaN0

alphaN1
betaN1

epsilon = 0.1
# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu0 =0
# The difference from the null hypothesis
gamma= .1
# The sd and variance of the sampling distribution
sigma =1
sigma_sq=sigma^2
N0 = (qnorm(1-alpha)
+ qnorm(1-beta))^2 * sigma_sq *gamma^{-2}
#non private sample size
N0 = ceiling(N0)
```

```
startLambda = 1
endLambda = 10
stepLambda = 1
numLambdaSteps = 1
+ ceiling( (endLambda - startLambda) / stepLambda)
simSize = 10000

K1 = matrix(nrow = 1, ncol = numLambdaSteps)
#K corrected factor
alphaN0 = matrix(nrow = 1, ncol = numLambdaSteps)
#alpha size for non -private size
betaN0 = matrix(nrow = 1, ncol = numLambdaSteps)
alphaN1 = matrix(nrow = 1, ncol = numLambdaSteps)
betaN1 = matrix(nrow = 1, ncol = numLambdaSteps)


for (k in 1:numLambdaSteps){
#Compute the corrected factors
# for each sampling distribution of the e-private statistic

Lambda = startLambda + (k - 1)*stepLambda

#K

dataSize=N0
start=dataSize
end=dataSize*1000
N1=root.Kfactor(start,end,dataSize,
Lambda,alpha,beta,epsilon,gamma)$root
N1=ceiling(N1)
K1[k]=N1/dataSize

#Compute the type I error and type II error
# using the e-private statistic and the non - private sample size
```

```
#N0
dataSize = N0
rejectionValue = mu0
+ qnorm(1-alpha) * sqrt(sigma_sq/ dataSize)
delta=1/dataSize
# alpha N0
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,
sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle
}
alphaN0[k] = mean(muEP > rejectionValue)
#MONTE CARLO ESTIMATOR OF THE ALPHA

# beta N0
mu1 = mu0 + gamma
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,
sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnorm(1, mean=0,
 sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle }

betaN0[k] = 1 - mean(muEP > rejectionValue)
```

```
#alpha and beta with adjusted sample size

# N1
dataSize = N1
rejectionValue = mu0
+ qnorm(1-alpha) * sqrt(sigma_sq/ dataSize)
delta=1/dataSize

# alpha N1
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)
for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle }

alphaN1[k] = mean(muEP > rejectionValue)

# beta N1
mu1 = mu0 + gamma
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize)  {
muML[i] = mean(rnorm(1,mean=mu1,
sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle}
betaN1[k] = 1 - mean(muEP > rejectionValue)
}
```

```
alphaN0
betaN0

alphaN1
betaN1

########################################
########################################
# Private rejection region without correction
########################################
########################################

Lambda = 1
# The confidence level alpha - Type I error
alpha = .05
# The power 1 - beta where beta - Type II error
beta = .1
# The null hypothesis
mu0 =0
# The difference from the null hypothesis
gamma = .1
# The sd and variance of the sampling distribution
sigma =1
sigma_sq=sigma^2
N0 = (qnorm(1-alpha) +
qnorm(1-beta))^2 * sigma_sq * gamma^{-2}
##non private sample size
N0 = ceiling(N0)
startEpsilon = 0.1
endEpsilon = 0.5
stepEpsilon = 0.1
numEpsilonSteps = 1
+ ceiling( (endEpsilon - startEpsilon) / stepEpsilon)
simSize = 10000
```

```
K1 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
##alpha size for non -private size
betaN0 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
betaN1 = matrix(nrow = 1, ncol = numEpsilonSteps)
alphaN2 = matrix(nrow = 1, ncol = numEpsilonSteps)
betaN2 = matrix(nrow = 1, ncol = numEpsilonSteps)

for (k in 1:numEpsilonSteps )
{epsilon = startEpsilon + (k - 1)*stepEpsilon

#Compute the type I error and type II error
#using the e-private statistic and the non - private sample size

# N0
dataSize = N0
rejectionValue = mu0 +
 qnorm(1-alpha) * sqrt(sigma_sq/ dataSize)
delta=1/dataSize

# alpha N0
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
#gaussian private mle }

alphaN0[k] = mean(muEP > rejectionValue)
```

```
# beta N0
mu1 = mu0 + gamma
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle}

betaN0[k] = 1 - mean(muEP > rejectionValue)

#alpha and beta  with private rejection regions

dataSize = N0
delta=1/dataSize
rejectionValue = mu0
+ qnorm(1-alpha) * sqrt(sigma_sq/ dataSize
+(2*Lambda^2*(log(1/delta)))/(epsilon^2*dataSize^2))

# alpha
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle }

alphaN1[k] = mean(muEP > rejectionValue)
```

```
# beta
dataSize=N0
delta=1/dataSize
rejectionValue=mu0 +
qnorm(1-alpha) * sqrt(sigma_sq/ dataSize+
(2*Lambda^2*(log(1/delta)))/(epsilon^2*dataSize^2))

mu1 = mu0 + gamma
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize) {
muML[i] = mean(rnorm(1,mean=mu1,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] +
rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
#gaussian private mle }

betaN1[k] = 1 - mean(muEP > rejectionValue)

##################################
# Using the corrector sample size factor
##################################
dataSize=N0
start=dataSize
end=dataSize*1000
N1=root.Kfactor(start,end,dataSize,Lambda,
alpha,beta,epsilon,gamma)$root
N1 = ceiling(N1)
K1[k]=N1/dataSize

dataSize = N1
delta=1/dataSize
rejectionValue = mu0
+ qnorm(1-alpha) * sqrt(sigma_sq/ dataSize
```

```
+(2*Lambda^2*(log(1/delta)))/(epsilon^2*dataSize^2))

# alpha
mean = mu0
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu0,sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnorm(1, mean=0, s
d=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle }

alphaN2[k] = mean(muEP > rejectionValue)

# beta
dataSize=N1
delta=1/dataSize
rejectionValue=mu0 +
qnorm(1-alpha) * sqrt(sigma_sq/ dataSize+
(2*Lambda^2*(log(1/delta)))/(epsilon^2*dataSize^2))

mu1 = mu0 + gamma
muML = rep(NA, simSize)
muEP = rep(NA, simSize)

for(i in 1:simSize){
muML[i] = mean(rnorm(1,mean=mu1,
sd=sigma/sqrt(dataSize)))
muEP[i] = muML[i] + rnorm(1, mean=0,
sd=(Lambda*(sqrt(2*log(1/delta)))/(dataSize*epsilon)))
##gaussian private mle }

betaN2[k] = 1 - mean(muEP > rejectionValue)  }
```

```
alphaN0
betaN0
alphaN1
betaN1
alphaN2
betaN2
```

A.6 Code for Chapter 4

```
#######################################################
#######################################################
# First set of simulations that evaluate the
#smooth sensitivity algorithms  controlling epsilon for a Lambda=1 and N=100
#######################################################
library('normalp')
library(msm)

set.seed(1234);

simSize = 1000
dataSize=100
Lambda=10
mu0=0
sigma=1


startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numSteps = 1
 + ceiling( (endEpsilon - startEpsilon) / stepEpsilon)
numSteps
```

```
smooth=matrix(0,numSteps)
local=matrix(0,dataSize)

mseML = rep(0,numSteps)
mseEP = rep(0,numSteps)
mseEP3 = rep(0,numSteps)

for (j in 1:numSteps){
epsilon = startEpsilon + (j - 1)*stepEpsilon
ML = rep(NA, simSize)
for(i in 1:simSize){
smoothrandom=rep(0,simSize)
data=rtnorm(dataSize,mean=mu0,sd=sigma,
lower=-Lambda,upper=Lambda)

data=sort(data)
data=matrix(data)
N=length(data)
delta=1/(N)
beta=(epsilon/2)*(log(1/delta))
alphaL=epsilon/2
alphaG=epsilon/(sqrt(log(1/delta)))

ML[i] = mean(data)
for (k in 1:N-1){
local[k]=(exp(-beta*k))*abs((sum(data[N-k+1:k])-k*ML[i])/(N-k))}
local

smoothrandom[i]=max(local)}
smooth[j]=sum((smoothrandom)^2)/simSize
mseML[j] = sum( (ML- mu0)^2 ) / simSize

tsigma=sigma^2*(1+
(-2*Lambda*dnorm(Lambda))/(2*pnorm(Lambda)-1))
varML=tsigma/dataSize
```

```
mseEP[j]=  (varML+(2*smooth[j])/(alphaL)^2 )/varML
#Laplace with smooth sensitivity
mseEP3[j]=(varML+(2*smooth[j])/(alphaG)^2)/varML
#Gaussian with smooth sensitivity
}

mseML=mseML/varML
mseML
mseEP
mseEP3

smooth

postscript("MsebyEpsilonLocal.ps")

#plots

 ltys = c("22", "44", "13", "1343", "73", "2262",
"12223242", "F282", "F4448444", "224282F2", "F1")
lineWD=2
textSize = 1
legendSize = 1
axisSize = 1
symbolSize = 1

range=range(mseML,mseEP,mseEP3)
plot(mseML, type="o", cex=symbolSize, pch=1,
lty=ltys[8], col="black", lwd=lineWD,ylim=range,
axes=FALSE, ann=FALSE)
 lines(mseEP, type="o", cex=symbolSize, pch=1,
lty=ltys[2], col="blue", lwd=lineWD)
lines(mseEP3, type="o", cex=symbolSize, pch=1,
lty=ltys[2], col="yellow", lwd=lineWD)
```

```
alphas = rep(0,numSteps)
for (i in 1:numSteps)
alphas[i] = startEpsilon + (i - 1)*stepEpsilon
axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,2, 0.01), cex.axis=axisSize)
title(xlab=expression(epsilon),cex.lab=1)
title(ylab="Standardized MSE", cex.lab=textSize)
legend("top", c("MLE","Laplace DP - smooth",
"Gaussian DP-smooth"), cex=legendSize, fill=c("black","blue","yellow"))

 dev.off()

##########################################################
# First set of simulations that
# Evaluate the smooth sensitivity algorithms
# controlling the samplesize for a Lambda=1 and e=0.1
##########################################################
library('normalp')
library(msm)

set.seed(1234);

simSize = 1000
Lambda=1
mu0=0
sigma=1
epsilon=0.1
alpha=1
startDataSize = 10
endDataSize = 100
stepDataSize = 10
numSteps = 1
+ ceiling( (endDataSize - startDataSize) / stepDataSize)
numSteps
```

```
smooth=matrix(0,numSteps)

varML=rep(0,numSteps)
mseML = rep(0,numSteps)
mseEP = rep(0,numSteps)
mseEP3 = rep(0,numSteps)

for (j in 1:numSteps){
dataSize = startDataSize + (j - 1)*stepDataSize
k = (dataSize^(alpha))
local=matrix(0,dataSize)
ML = rep(NA, simSize)

for(i in 1:simSize){
smoothrandom=rep(0,simSize)
data=rtnorm(dataSize,mean=mu0,
sd=sigma, lower=-Lambda,upper=Lambda)

data=sort(data)

data=matrix(data)
N=length(data)
delta=1/(N)
beta=(epsilon/2)*(log(1/delta))
alphaL=epsilon/2
alphaG=epsilon/(sqrt(log(1/delta)))

ML[i] = mean(data)

for (k in 1:N-1){
local[k]=(exp(-beta*k))*abs((sum(data[N-k+1:k])-k*ML[i])/(N-k))}
local

smoothrandom[i]=max(local)}
smooth[j]=sum((smoothrandom)^2)/simSize
```

```
mseML[j] = sum( (ML- mu0)^2 ) / simSize
tsigma=sigma^2*(1
+(-2*Lambda*dnorm(Lambda))/(2*pnorm(Lambda)-1))
varML[j]=tsigma/k
mseEP[j]=  (varML[j]+(2*smooth[j])/(alphaL)^2 )/varML [j]
#Laplace with smooth sensitivity
mseEP3[j]=(varML[j]+(2*smooth[j])/(alphaG)^2)/varML[j]
#Gaussian with smooth sensitivity}

mseML=mseML/varML
mseML
mseEP
mseEP3

smooth

postscript("MseByNSmooth.ps")

# plots

ltys = c("22", "44", "13", "1343", "73", "2262",
"12223242", "F282", "F4448444", "224282F2", "F1")
lineWD=2
textSize = 1
legendSize = 1
axisSize = 1
symbolSize = 1

range=range(mseML,mseEP,mseEP3)
plot(mseML, type="o", cex=symbolSize, pch=1,
lty=ltys[8], col="black", lwd=lineWD,ylim=range, axes=FALSE, ann=FALSE)
lines(mseEP, type="o", cex=symbolSize, pch=1,
lty=ltys[2], col="blue", lwd=lineWD)
lines(mseEP3, type="o", cex=symbolSize, pch=1,
lty=ltys[2], col="yellow", lwd=lineWD)
```

```
alphas = rep(0,numSteps)
for (i in 1:numSteps)
alphas[i] = startDataSize + (i - 1)*stepDataSize

axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,range[2],0.5), cex.axis=axisSize)
title(xlab=expression(N),cex.lab=1)
title(ylab="Standardized MSE", cex.lab=textSize)
legend("top", c("MLE","Laplace DP - smooth",
"Gaussian DP-smooth"), cex=legendSize,
fill=c("black","blue","yellow"))

dev.off()

################################################################
################################################################
# simulations that compare the smooth
# sensitivity algorithms with the GD-algorithms
# as we varying epsilon
################################################################
library('normalp')
library(msm)
set.seed(1234);
simSize = 1000
dataSize=100
Lambda=1
mu0=0
sigma=1
startEpsilon = .1
endEpsilon = .5
stepEpsilon = .1
numSteps = 1
+ ceiling( (endEpsilon - startEpsilon) / stepEpsilon)
numSteps
```

```
smooth=matrix(0,numSteps)
local=matrix(0,dataSize)

mseML = rep(0,numSteps)
mseEP = rep(0,numSteps)
mseEP1 = rep(0,numSteps)
mseEP2 =rep(0,numSteps)
mseEP3 = rep(0,numSteps)

for (j in 1:numSteps){
epsilon = startEpsilon + (j - 1)*stepEpsilon
ML = rep(NA, simSize)
for(i in 1:simSize) {
smoothrandom=rep(0,simSize)
data=rtnorm(dataSize,mean=mu0,sd=sigma,
lower=-Lambda,upper=Lambda)

data=sort(data)

data=matrix(data)
N=length(data)
delta=1/(N)
beta=(epsilon/2)*(log(1/delta))
alphaL=epsilon/2
alphaG=epsilon/(sqrt(log(1/delta)))

ML[i] = mean(data)
for (k in 1:N-1){
local[k]=(exp(-beta*k))*abs((sum(data[N-k+1:k])-k*ML[i])/(N-k))}
local
smoothrandom[i]=max(local)}

smooth[j]=sum((smoothrandom)^2)/simSize
mseML[j] = sum( (ML- mu0)^2 ) / simSize
```

```
tsigma=sigma^2*(1
+(-2*Lambda*dnorm(Lambda))/(2*pnorm(Lambda)-1))
varML=tsigma/N
mseEP[j]=  (varML
+(2*smooth[j])/(alphaL)^2 )/varML
#Laplace with smooth sensitivity
mseEP1[j]=(varML
+(2*Lambda^2)/(N^2*epsilon^2))/varML
 #Laplace with global sensitivity
mseEP2[j]=(varML
+((2*Lambda*(sqrt(2*log(1/delta)))/(N*epsilon)))^2)/varML
#Gaussian with global sensitivity
mseEP3[j]=(varML+(2*smooth[j])/(alphaG)^2)/varML
#Gaussian with smooth sensitivity

}
mseML=mseML/varML
mseML
mseEP
mseEP1
mseEP2
mseEP3

smooth

 postscript("NormalMsebyEpsilonLocal.ps")
 ltys = c("22", "44", "13", "1343", "73", "2262",
"12223242", "F282", "F4448444", "224282F2", "F1")
lineWD=2
textSize = 1
legendSize = 1
axisSize = 1
symbolSize = 1
range=range(mseML,mseEP,mseEP1,mseEP2,mseEP3)
plot(mseML, type="o", cex=symbolSize, pch=1,
```

```
lty=ltys[8], col="black", lwd=lineWD,ylim=range,
axes=FALSE, ann=FALSE)
lines(mseEP, type="o", cex=symbolSize, pch=1,
lty=ltys[2], col="blue", lwd=lineWD)
lines(mseEP1, type="o", cex=symbolSize, pch=1,
lty=ltys[1], col="red", lwd=lineWD)
lines(mseEP2, type="o", cex=symbolSize, pch=1,
lty=ltys[11], col="purple", lwd=lineWD)
lines(mseEP3, type="o", cex=symbolSize, pch=1,
lty=ltys[2], col="yellow", lwd=lineWD)

alphas = rep(0,numSteps)
for (i in 1:numSteps)
alphas[i] = startEpsilon + (i - 1)*stepEpsilon

axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,max(range),10), cex.axis=axisSize)
title(xlab=expression(epsilon),cex.lab=1)
title(ylab="Standardized MSE", cex.lab=textSize)
legend("top", c("MLE","Laplace DP - smooth",
"Laplace DP-global","Gaussian DP-global",
"Gaussian DP-smooth"), cex=legendSize,
fill=c("black","blue","red","purple","yellow"))

dev.off()

#######################################################
# simulations that compare the smooth sensitivity
# algorithms with the GD-algorithms
# as we varying sample size
#######################################################
library('normalp')
library(msm)

set.seed(1234);
```

```
simSize = 1000
epsilon=0.1
Lambda=1
mu0=0
sigma=1
startDataSize = 10
endDataSize = 100
stepDataSize = 10
numSteps = 1
+ ceiling( (endDataSize - startDataSize) / stepDataSize)
numSteps
smooth=matrix(0,numSteps)
varML=rep(0,numSteps)
mseML = rep(0,numSteps)
mseEP = rep(0,numSteps)
mseEP1 = rep(0,numSteps)
mseEP2 =rep(0,numSteps)
mseEP3 = rep(0,numSteps)

for (j in 1:numSteps){
dataSize = startDataSize + (j - 1)*stepDataSize
k = (dataSize^(alpha))
local=matrix(0,dataSize)
ML = rep(NA, simSize)
for(i in 1:simSize){
smoothrandom=rep(0,simSize)
data=rtnorm(dataSize,mean=mu0,
sd=sigma, lower=-Lambda,upper=Lambda)
data=sort(data)
data=matrix(data)
N=length(data)
delta=1/(N)
beta=(epsilon/2)*(log(1/delta))
alphaL=epsilon/2
alphaG=epsilon/(sqrt(log(1/delta)))
```

```
ML[i] = mean(data)
for (k in 1:N-1){
local[k]=(exp(-beta*k))*abs((sum(data[N-k+1:k])-k*ML[i])/(N-k))}

local

smoothrandom[i]=max(local)}

smooth[j]=sum((smoothrandom)^2)/simSize
mseML[j] = sum( (ML- mu0)^2 ) / simSize

tsigma=sigma^2*(1+(-2*Lambda*dnorm(Lambda))/(2*pnorm(Lambda)-1))
varML[j]=tsigma/k
mseEP[j]=  (varML[j]+(2*smooth[j])/(alphaL)^2 )/varML[j]
#Laplace with smooth sensitivity
mseEP1[j]=(varML[j]+(2*Lambda^2)/(N^2*epsilon^2))/varML[j]
#Laplace with global sensitivity
mseEP2[j]=(varML[j]
+((2*Lambda*(sqrt(2*log(1/delta)))/(N*epsilon)))^2)/varML[j]
#Gaussian with global sensitivity
mseEP3[j]=(varML[j]+(2*smooth[j])/(alphaG)^2)/varML[j]
#Gaussian with smooth sensitivity}
mseML=mseML/varML
mseML
mseEP
mseEP1
mseEP2
mseEP3

smooth
postscript("NormalMsebyDataSizeLocal.ps")

ltys = c("22", "44", "13", "1343", "73", "2262", "12223242",
"F282", "F4448444", "224282F2", "F1")
```

```
lineWD=2
textSize = 1
legendSize = 1
axisSize = 1
symbolSize = 1

range=range(mseML,mseEP,mseEP1,mseEP2,mseEP3)
plot(mseML, type="o", cex=symbolSize, pch=1,
lty=ltys[8], col="black", lwd=lineWD,ylim=range,
axes=FALSE, ann=FALSE)
lines(mseEP, type="o", cex=symbolSize, pch=1,
lty=ltys[2], col="blue", lwd=lineWD)
lines(mseEP1, type="o", cex=symbolSize, pch=1,
lty=ltys[1], col="red", lwd=lineWD)
lines(mseEP2, type="o", cex=symbolSize, pch=1,
lty=ltys[11], col="purple", lwd=lineWD)
lines(mseEP3, type="o", cex=symbolSize, pch=1,
lty=ltys[2], col="yellow", lwd=lineWD)

alphas = rep(0,numSteps)
for (i in 1:numSteps)
alphas[i] = startDataSize + (i - 1)*stepDataSize
axis(1, at=1:numSteps, lab=alphas, cex.axis=axisSize)
axis(2, las=1, at=seq(0,max(range),100), cex.axis=axisSize)
title(xlab=expression(N),cex.lab=1)
title(ylab="Standardized MSE", cex.lab=textSize)
legend("top", c("MLE","Laplace DP - smooth",
"Laplace DP-global","Gaussian DP-global","Gaussian DP-smooth"),
cex=legendSize,
fill=c("black","blue","red","purple","yellow"))

dev.off()

library('normalp')
library(msm)
```

```
set.seed(1234);

simSize = 100
mu0=0
sigma=1
epsilon=0.1
startLambda=1
endLambda=10
stepLambda=1
numStepsLambda=1+
ceiling((endLambda-startLambda)/stepLambda)
numStepsLambda
startDataSize=10
endDataSize=100
stepDataSize=10
numStepsDataSize=1
+ceiling((endDataSize-startDataSize)/stepDataSize)
numStepsDataSize
smooth=matrix(0,numStepsLambda,numStepsDataSize)
for (l in 1:numStepsLambda){

Lambda=startLambda+(l-1)*stepLambda

for (n in 1:numStepsDataSize){
DataSize=startDataSize+(n-1)*stepDataSize

local=matrix(0,DataSize)
for (j in 1:simSize){
smoothrandom=rep(0,simSize)
data=rtnorm(DataSize,mean=mu0,sd=sigma,
lower=-Lambda,upper=Lambda)
data=sort(data)
data=matrix(data)
N=length(data)
```

```
delta=1/(2*N)
beta=(epsilon/2)*(log(1/delta))
alphaL=epsilon/2

ML = mean(data)

for (k in 1:N-1){
local[k]=(exp(-beta*k))*abs((sum(data[N-k+1:k])-k*ML)/(N-k))}

local
smoothrandom[j]=max(local)}
smooth[l,n]=sum(smoothrandom^2)/simSize}}
smooth
ratio=matrix(0,numStepsLambda,numStepsDataSize)
for (l in 1:numStepsLambda){
Lambda=startLambda+(l-1)*stepLambda
for (n in 1:numStepsDataSize){
DataSize=startDataSize+(n-1)*stepDataSize
ratio[l,n]=Lambda/DataSize}
}
ratio<smooth
```

Bibliography

[1] Nabil R Adam and John C Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys (CSUR)*, 21(4):515–556, 1989.

[2] Xiaohong Cao, Karen B Maloney, and Vladimir Brusic. Data mining of cancer vaccine trials: a bird's-eye view. *Immunome Research*, 4(1):7, 2008.

[3] George Casella and Roger L Berger. *Statistical inference*, volume 70. Duxbury Press Belmont, CA, 1990.

[4] Ling Chen. Testing the mean of skewed distributions. *Journal of the American Statistical Association*, 90(430):767–772, 1995.

[5] Dorothy E Denning. Secure statistical databases with random sample queries. *ACM Transactions on Database Systems (TODS)*, 5(3):291–315, 1980.

[6] C. Dwork. Differential privacy. *Automata, languages and programming*, pages 1–12, 2006.

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *Theory of Cryptography*, pages 265–284, 2006.

[8] C. Dwork and A. Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2):2, 2010.

[9] Cynthia Dwork. An ad omnia approach to defining and achieving private data analysis. In *Privacy, Security, and Trust in KDD*, pages 1–13. Springer, 2008.

[10] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[11] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 371–380. ACM, 2009.

[12] S. Fienberg and A. Slavkovic. Data privacy and confidentiality.

[13] Anco Hundepool, Josep Domingo-Ferrer, Luisa Franconi, Sarah Giessing, Eric Schulte Nordholt, Keith Spicer, and Peter-Paul De Wolf. *Statistical disclosure control*. John Wiley & Sons, 2012.

[14] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, 2011.

[15] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 94–103. IEEE, 2007.

[16] John Neter, William Wasserman, Michael H Kutner, et al. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.

[17] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2007.

[18] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[19] W.J. Reed. The normal-laplace distribution and its relatives. *Advances in Distribution Theory, Order Statistics, and Inference*, pages 61–74, 2006.

[20] A. Smith. Efficient, differentially private point estimators. *Arxiv preprint ArXiv:0809.4794*, 2008.

[21] Abhradeep Guha Thakurta. Differentially private convex optimization for empirical risk minimization and high-dimensional regression, phd thesis. 2012.

[22] G. Van Belle. Sample size. *Statistical rules of thumb. New York (NY): John Wiley & Sons*, pages 29–51, 2002.

[23] D. Vu and A. Slavkovic. Differential privacy for clinical trial data: Preliminary evaluations. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*, pages 138–143. IEEE, 2009.

[24] L. Wasserman and S. Zhou. A statistical framework for differential privacy. *Journal of the American Statistical Association*, 105(489):375–389, 2010.

[25] Larry Wasserman. Minimaxity, statistical thinking and differential privacy. *Journal of Privacy and Confidentiality*, 4(1):3, 2012.

[26] Donald W Zimmerman. Comparative power of student t test and mann-whitney u test for unequal sample sizes and variances. *The Journal of Experimental Educational*, pages 171–174, 1987.