Differential Privacy and Synthetic Data

Tianen (Benjamin) Liu

5/8/2020

Acknowledgment

I would like to express my gratitude to Dr. Nicole Dalzell who helped, guided, and encouraged me through the research journey as my research advisor, and to the faculty of the Department of Mathematics and Statistics, especially Dr. Staci Hepler and Dr. Kenneth Berenhaut, for the great classes.

This research was made possible with the support from the URECA Center of Wake Forest University who provided the research opportunity during summer 2019 and the Wake Forest Research Fellowship.

I would also like to thank Dr. Denisha Champion and the Counseling Center at Wake Forest University for providing the $HealthyMinds$ data set and an opportunity for me to contribute to the Wake Forest community through analyzing this data set.

Abstract

A data set is subject to data breach when it is linked with another or more data sets, causing sensitive information released. Differential privacy is a privacy guarantee of a released data set that ensures whether one is in the data set or not is unidentifiable and no new information can be learned from the released data set. There are various methods to make a released result differentially private. Most of them do not create a differentially private data set, but a column or a count, meaning only a part of the data set is modified by differential privacy methods. Alternatively, methods for synthetic data can generate a synthetic data set with all elements in the data set modified, but does not guarantee differential privacy. The main objective of this research is to explore methods of differential privacy and synthetic data, and apply a subset of these methods on the *Healthy Minds* data set. The methods covered in this paper include Laplacian noise, randomized response, drawing from Bernoulli posterior predictive distribution, differentially private trees, and synthetic data creation. Laplacian noise, synthetic data, and differentially private trees are used to modify the *Healthy Minds* data set. A tree model is fit for all versions of the data set and we conclude that differentially private trees perform the best.

Contents

List of Figures

1 Introduction

In 2018, data of over 87 million users of a social media application was released to Cambridge Analytics for analysis on political views (Lapowsky 2018). Now in the big data-driven world, our personal data are essential for companies and government to make accurate decisions. However, as we provide useful data for analysis, sometimes they are attached with sensitive information from names or e-mail addresses to mental conditions or health records. Even when such sensitive information are often removed when releasing the data publically, we still frequently see data breach instances.

Years ago, the medical information of the former Massachusetts governor was leaked when hackers identified him by combining anonymous medical data set with anonymous voter list, which share zip code and date of birth information (Barth-Jones 2012). This example shows that even people's sensitive information is concealed, it still can be revealed by linkage with another seemingly secure data set.

Data providers want their sensitive information to be unidentifiable. However, companies and government cannot stop releasing or using data sets that contain as much information as possible to ensure accurate analysis. Thus we face a trade-off between a data set's accuracy and providers' privacy.

Differential privacy sets a privacy level for a public data set that benefits both sides of the trade-off. A differential privacy technique modifies a data set and ensures its usability for companies who analyze it without having to directly access the sensitive information in the original raw data. At the same time, data providers do not have to worry about data breach because, based on differential privacy, it is unknown if a piece of sensitive information is the actual raw data or not. Moreover, since privacy concerns are reduced, people will be encouraged to contribute to the data set on sensitive topics, such as a drug consumption survey, which requires a large population to ensure better analysis.

Synthetic data, which may contain one or more arributes, is a data set that has been completely modified but still preserves the relationship among its attributes. Differential privacy methods mostly cannot generate a synthetic data set. When necessary, synthetic data methods can be used to generate a different but similar data set to reduce privacy risks. Synthetic data does not guarantee differential privacy.

The main objective of this research is to explore methods of differential privacy and synthetic data, and apply a subset of these methods on the *Healthy Minds* data set. In Section 2, definition of differential privacy will be introduced. In Section 3, we present four differential privacy methods. The methods covered in this paper include Laplacian noise, randomized response, drawing from Bernoulli posterior predictive distribution, and differentially private trees. We also use *ncbirths* data set to show how certain methods work and their comparison. In Section 4, we introduce synthetic data creation. Laplacian noise, synthetic data, and differentially private trees are used to modify the *Healthy Minds* data set. We discuss our results and conclusions in Section 5.

2 Differential Privacy

2.1 Definition

Let I be the population whose data are collected, d_i be the information given by person i, $D_I = d_i | i \in I$ be the data set collected from all people in I, Q be the privatized query run on a data set, and $R = Q(D_I)$ be the resultant modified data set released to the public.

Ideally, differential privacy ensures that whether one person is in the data set does not impact the released data set. Based on this, we have $Q(D_{I-i}) = Q(D_I)$ (Task 2014). This should hold whenever, meaning the probability of $Q(D_{I-i})$ being equal to $Q(D_I)$ should be similar. Thus ϵ-differential privacy is defined as (Task 2014):

$$\frac{Prob(Q(D_I) = R)}{Prob(Q(D_{I\pm i}) = R)} \leq e^{\epsilon} \text{ for small } \epsilon \geq 0.$$

2.2 Clarifications

Even if you do not provide data for a particular data set, a hacker could still learn things about you from it, based on the analyses of the data in the survey. For example, a hacker may understand that as age increases, one has higher risks of a certain disease. A hacker can make predictions about anyone not in the data set based on this information.

Also, individual information is protected by differential privacy but group information is not necessarily. A hacker can guess with a high probability whether or not one is in the data set if one is in a known cohesive group. For example, a hacker can learn that college students are the main consumer group of fashion products. One's behavior would be easily predicted if one falls into the category of college students. What differential privacy can do is that it ensures the released data set R does not provide additional information of you. Mathematically, let $S(i)$ be the set of sensitive information of person i, $P(S(i)|R) = P(S(i))$.

3 Differential Privacy: Explorative

3.1 Laplacian Noise

Definition

Let Global Sensitivity of F, a function that modifies data sets, $= \Delta F = max_{(D_1,D_2)}|F(D_1) - F(D_2)|$, which means max difference in answers that adding or removing any individual from the data set can cause. If we take many noise samples from a Laplace distribution centered at 0 with scale $b = \frac{\Delta F}{\epsilon}$ to be added to true count, the released counts will have a Laplace distribution $Prob(R = x|D) = \frac{\epsilon}{2\Delta F} e^{-\frac{|x-F(D)|\epsilon}{\Delta F}}$ with ϵ-differential privacy (Task 2014).

Proof that Laplacian Noise Satisfies Epsilon Differential Privacy

Proof.

$$\frac{Prob(R|Q(D_I))}{Prob(R|Q(D_{I\pm 1}))} \leq e^{\epsilon}$$
$$\frac{\frac{\epsilon}{2\Delta F} e^{-\frac{|R-F(D)|\epsilon}{\Delta F}}}{\frac{\epsilon}{2\Delta F} e^{-\frac{|R-F(D_{I+i})|\epsilon}{\Delta F}}} \leq e^{\epsilon}$$
$$e^{\frac{\epsilon}{\Delta F}|F(D_I)-F(D_{I\pm i})|} \leq e^{\epsilon}$$
$$\frac{|F(D_I) - F(D_{I\pm i})|}{\Delta F} \leq 1$$

□

Adding Laplacian Noise on a Number

Laplacian noise is a useful method for releasing numerical values such as a count or a statistic. Adding Laplacian noise to a result ensures ϵ-differential privacy in the case of a counting query, i.e., when one is only interested in the count of a certain value, or a query for median. For example, if we were to release the number of smokers in a medical data set, we can add Laplacian noise to this true number of smokers before releasing the count.

We did a simulation of 10,000 counts of smokers using Laplacian noise on the *ncbirths* data set provided by John Holcomb at Cleveland State University. We use the attribute `habit`, a binary variable indicating whether a particular person is a smoker. The count of smokers, with true value being 126, is what we try to protect with $\epsilon = 0.1$ differential privacy. Figure 1 below shows that if we draw many noises from Laplace distribution with $\mu = 126$ and $b = \frac{1}{\epsilon}$, and add the noises to the true count (126), those values will exhibit a

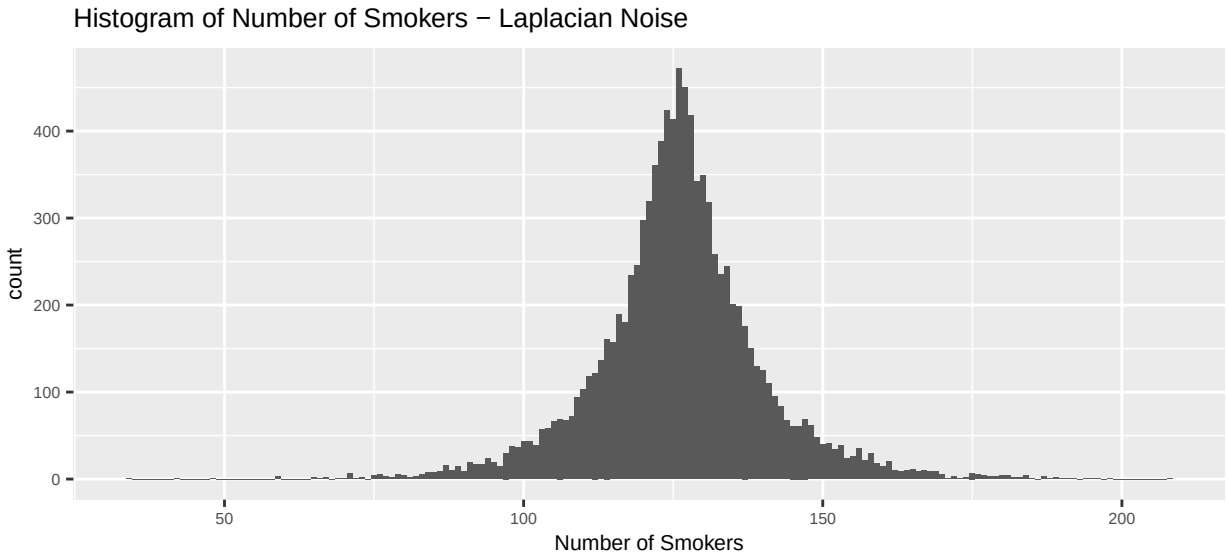

Figure 1: Shown is the histogram of the number of smokers in ncbirths data set with Laplacian noises added 10,000 times.

Laplace distribution $Lap \sim (\mu = 126, b = \frac{1}{0.1}$. Most of the values are centered around the true value 126 and most of the noises are very small.

The significance of this count when we release it publically is that it may expose the sensitive information when another person's data is added to the data set. When a person with smoking habit is added to the data set, the count of smokers should increase from 126 to 127, which exposes the persons smoking habit.

To ensure differential privacy of the released count, we add a Laplacian noise to it. Suppose we get a released count of 127, and suppose a hacker knows we use Laplacian noise in releasing the count. The hacker will know the distributions shown in Figure 2. Even if the hacker knows true count is 126 before adding the additional person, the hacker does not know if the additional person is a smoker or not because the probability of generating a released 127 using Laplacian noises is roughly the same no matter the true count after adding the additional person is 126 or 127. This is shown by the bottom part of Figure 2.

Adding Laplacian Noise on a Numerical Value

In addition to count or statistics, we can also add Laplacian noises to numerical variables to ensure differential privacy of numerical data while also preserving the relationship between the numerical variable and other variables. Shown in Figure 3 are histograms of the distribution of mother's age using original data and data with Laplacian noises added with epsilon equals 0.1, 0.5, and 1. When epsilon is small, the released data is more private, as shown in the histogram with $\epsilon = 0.1$, the distribution differs greatly from the original one. On the contrary, shown in the histogram with $\epsilon = 1$, the distribution is roughly the same as the original one.

3.2 Randomized Response

Similar to Laplacian noise, randomized response ensures ϵ-differential privacy for numerical data and a numerical value such as count or median of binary data. The methods to generate them will be introduced in the following sections.

Randomized Response for Numerical Values and Binary Data

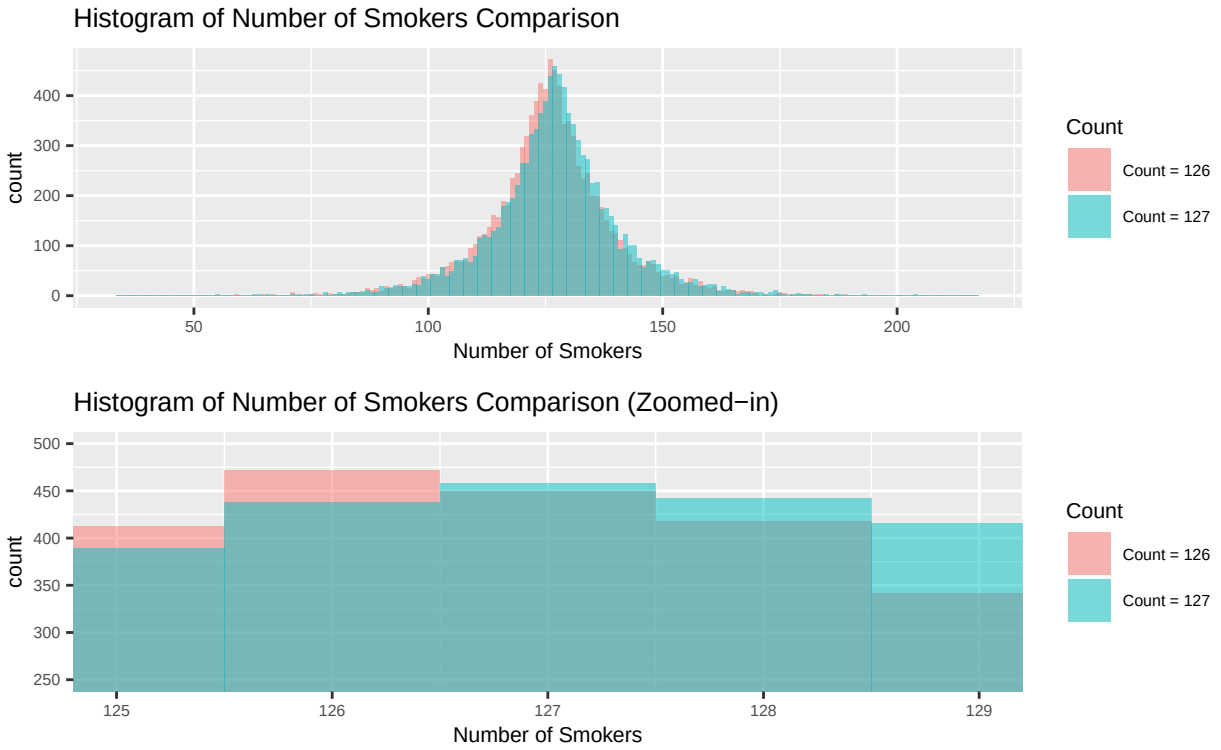

Figure 2: Shown are the distributions of the values 126 (true number of smokers) and 127 with Laplacian noises added 10,000 times.

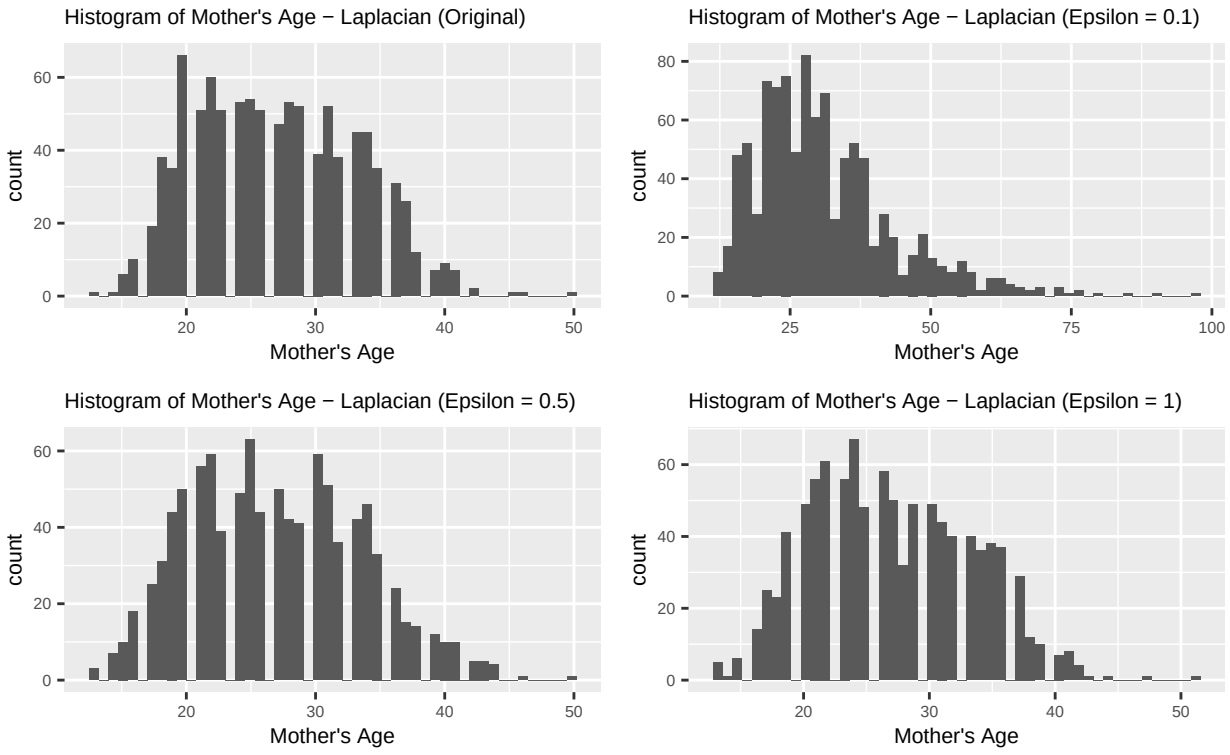

Figure 3: Shown are the comparison of distributions of mother's age in ncbirth data set using original data and data with Laplacian noises added with epsilon equals 0.1, 0.5, and 1.

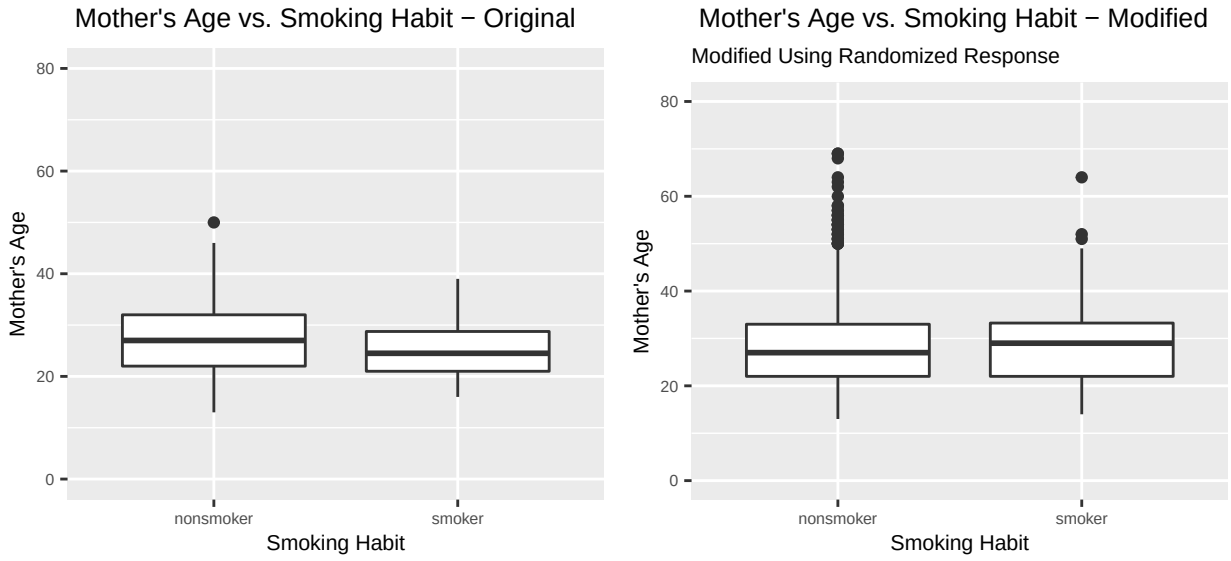

Figure 4: Shown is the comparison of box plots generated with original data and data modified using randomized response showing the relationship between mother's age vs. smoking habit in ncbirths data set.

One different of randomized response from Laplacian noise is that it does not modify the numerical value directly. Instead, it creates a new set of data by applying randomized response for each element. For each binary element $d \in D$ in the data set, the randomized response applied to it is described as follows.

We flip a biased coin with probability of heads α. If heads, then answer truthfully with d. If tails, flip a coin with probability of heads β and then answer with one response if heads and the other response if tails. Not every pair of α and β can achieve differential privacy. We need to tune the parameters and set a limit, exceeding which will not guarantee differential privacy.

Assume a binary variable contains two levels: *yes* and *no*. We control the parameters α and β that satisfy differential privacy using an extreme case of definition (Dwork, Roth, and others 2014), which is when no matter the original data is *yes* and *no*, we both get the released data $r = yes$. We write the definition of ϵ-differential privacy using this information:

$$\frac{P[Q(d_{yes}, \alpha, \beta) = yes]}{P[Q(d_{no}, \alpha, \beta) = yes]} \leq e^{\epsilon}$$

from which we can get

$$ln(\frac{\alpha + (1-\alpha)\beta}{1 - (\alpha + (1-\alpha)\beta)}) \leq \epsilon$$

This bound is how much we need to tune α and β within. The subsection below *Parameter Tuning: Choice of Alpha and Beta* shows how the parameters are selected.

We use randomized response to create a modified `habit` column (denoted as `Smoking Habit` in Figure 4) in *ncbirths* data set. We create box plots (Figure 4) showing the relationship between `Mother's Age` and `Smoking Habit` using original and modified `Smoking Habit` data. Modified data exhibits a higher median `Mother's Age` when one is a smoker, which should be lower. Randomized response fails to preserve the relationship between a binary variable and other variables.

Randomized Response for Numeric Data

Randomized response creates a differentially private version of numeric data by applying randomized response for each element. The method applied to each element is different from that applied to binary elements

described above. For each numeric element $d \in D$ in the data set, we flip a biased coin with probability of heads α. If heads, then answer truthfully with d. If tails, we apply Laplacian noise on it and then release it.

Parameter α, as in the case of binary data, needs to be tuned for the best utility, but the algorithm satisfy ϵ-differential privacy no matter what value α is.

Proof. If the released data r is the original value, which means we have a head in the coin toss, we have

$$\begin{aligned} \frac{Prob(r|Q(D_I))}{Prob(r|Q(D_{I\pm1}))} &\leq e^{\epsilon} \\ \frac{a}{a} &\leq e^{\epsilon} \end{aligned}$$

which holds. The probabilities are a because the result r being the original value is due to the coin toss, which is irrelevant to how the data look like.

If the released data r is not the original value, which means we have a tail in the coin toss, we multiply $1-\alpha$ to the probabilities of Laplace distribution. We then have

$$\begin{aligned} \frac{Prob(r|Q(D_I))}{Prob(r|Q(D_{I\pm1}))} &\leq e^{\epsilon} \\ \frac{(1-\alpha)\frac{\epsilon}{2\Delta F}e^{-\frac{|R-F(D)|\epsilon}{\Delta F}}}{(1-\alpha)\frac{\epsilon}{2\Delta F}e^{-\frac{|R-F(D_{I+i})|\epsilon}{\Delta F}}} &\leq e^{\epsilon} \\ e^{\frac{\epsilon}{\Delta F}|F(D_I)-F(D_{I\pm i})|} &\leq e^{\epsilon} \\ \frac{|F(D_I)-F(D_{I\pm i})|}{\Delta F} &\leq 1 \end{aligned}$$

which holds. □

Parameter Tuning: Choice of Alpha and Beta

We generated 40 α values and 40 β values between 0 and 1, with gap of 0.025, which gives us 1600 pairs of different α and β values. For each pair, if they satisfy the value bounds for 0.1-differential privacy, we generated 100 samples of `habit` column in *ncbirths* data set, which is a binary variable, and record the count of smokers. Recall the true value is 126. Of the 100 recorded count of smokers, we take the mean to make it the result for a particular pair of α and β values. Then we compute the difference between this count and the true value, shown in Figure 5. The pair of α and β values that generates the least difference will be used because it has the highest utility among all values that satisfy 0.1-differential privacy. We chose $\alpha = 0.3$ and $\beta = 0.125$. When $\beta = 0.125$, the difference is generally low because 0.125, among all β values, is the closest to the true percentage of smokers in the original data set ($\frac{126}{999} = 0.126$).

Comparison of Laplacian Noise and Randomized Response on Model Relationship Preservation

We compare the results released by Laplacian noise and randomized response under 0.1-differential privacy. As shown in Figure 6, we compare Laplacian noise and randomized response generating 10,000 samples of count of smokers in *ncbirths* data set. Laplacian noise generates a wider range of values, from about 50 to nearly 200. Both methods have most of their values between 100 and 150 and center at about true value 126. Laplacian noise has a higher probability in generating a value close to the true value, shown by the spike in the left portion of Figure 6.

Next we compare Laplacian noise and randomized response in their released numerical data. In Figure 7, we show a comparison of Laplacian noise and randomized response creating a simple linear regression of `Weight` versus `Mother's Age` in *ncbirths* data set. We use $\alpha = 0.5$ for randomized response. Although the

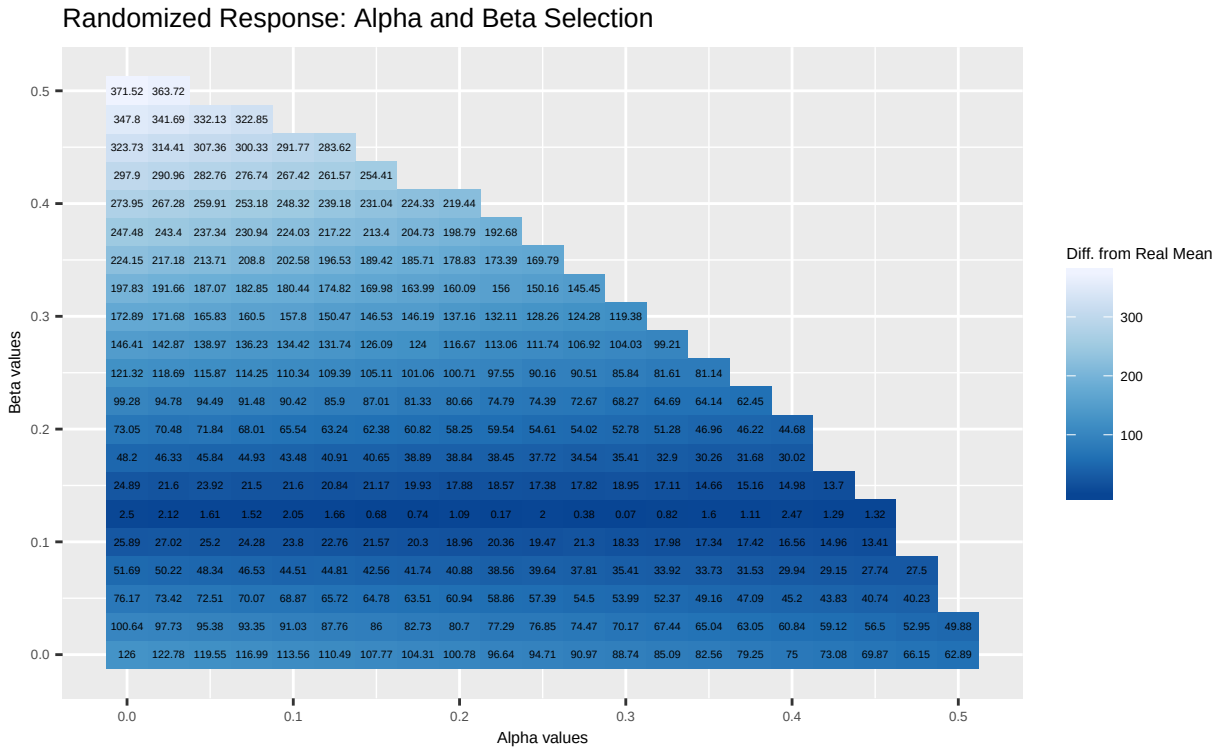

Figure 5: Shown is the heat map indicating the difference of the generated value between the true value using different alpha and beta parameters in randomized response.

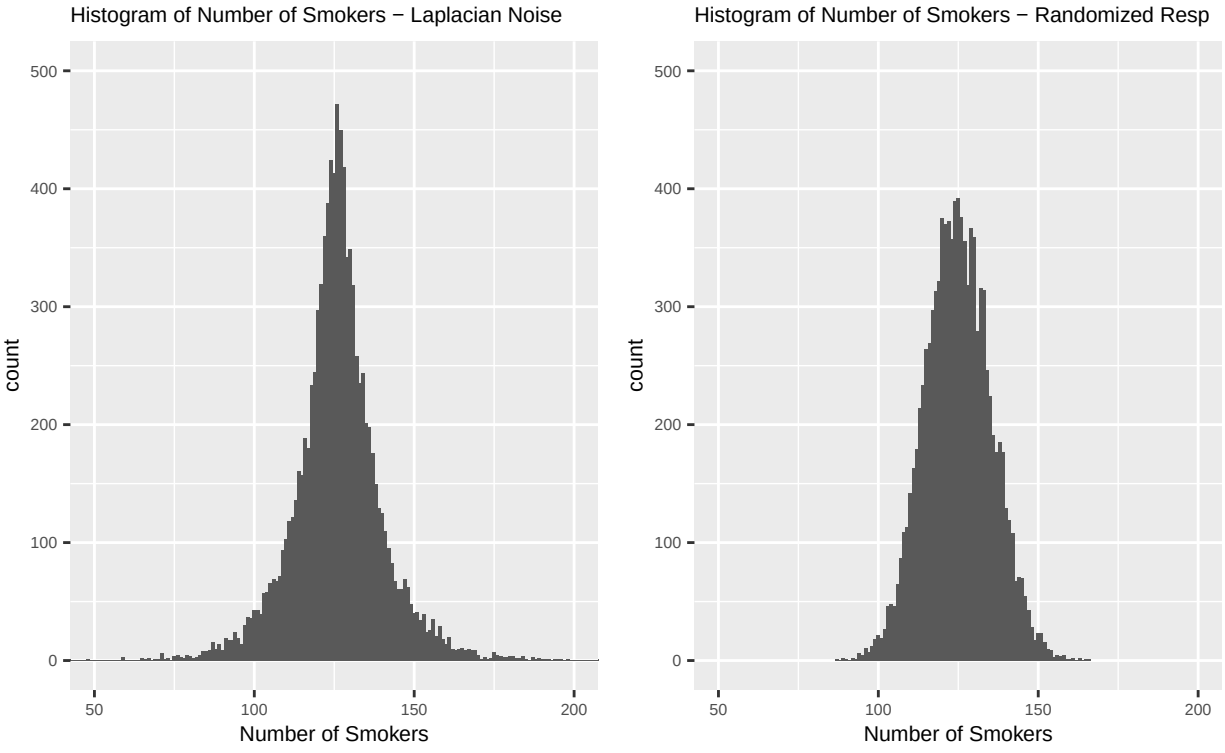

Figure 6: Shown is the comparison of Laplacian noise and randomized response generating 10,000 samples of count of smokers in ncbirths data set.

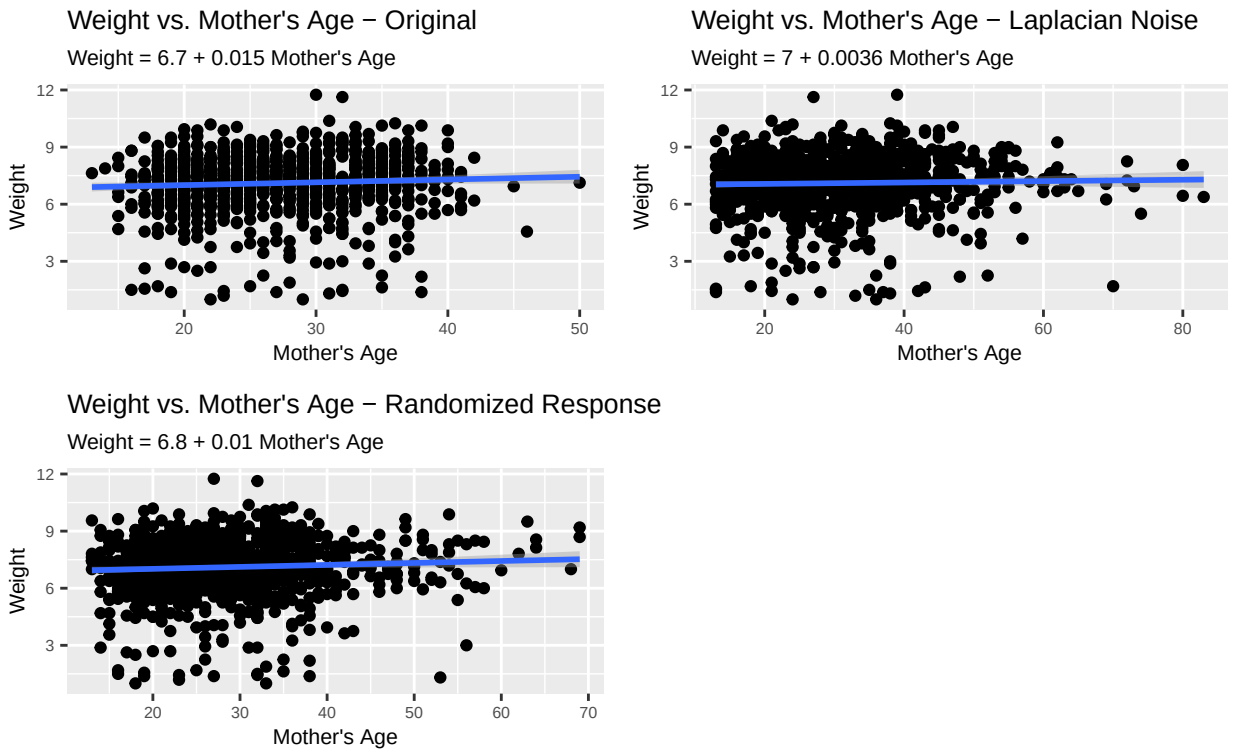

Figure 7: Shown is the comparison of Laplacian noise and randomized response creating a simple linear regression of weight vs. mother's age in ncbirths data set.

data distributions of both methods are greatly different from the original one, randomized response, which performs better than Laplacian noise, preserves the relationship between these two variables very well. This might be due to the fact that randomized response preserves more real values than Laplacian noise, in which every element is added a noise.

3.3 Bernoulli Posterior Predictive Distribution

As in randomized response, we are interested in generating a synthetic version of the binary column. Drawing from a Bernoulli distribution in generating synthetic binary column is feasible because a Bernoulli distribution gives the probability of *yes* in a binary variable, if we assume the two levels are *yes* and *no*. To make ensure utility, we need to make sure that we draw from a Bernoulli distribution with the right parameter that defines the probability of *yes*.

A Bayesian method involving Bernoulli posterior predictive distribution is suitable for this situation. What we need is a probability of *yes*. Let Y be a target attribute/column that we try to protect, and Y_i be a data point in Y, and Y_i^* be a synthetic version of Y_i. Mathematically, this probability is $p(Y_i^* = 1|D)$ because we shall generate Y_i^* based on the original data set D. $Y_i^* = 1|D$ should follow a Bernoulli distribution. The following subsection will show the derivation of this probability. Before deriving, we are going to introduce some essential terms.

We define p to be the true probability of *yes* in binary column Y. However, p is not a constant, but follows a distribution. We use $Beta(\alpha, \beta)$ distribution because it is bounded between 0 and 1. The distribution of p is our prior knowledge of the true probability before we observe any data. We write prior as $p|\alpha_\epsilon, \beta_\epsilon \sim Beta(\alpha_\epsilon, \beta_\epsilon)$, where α_ϵ and β_ϵ are their values for the corresponding ϵ. We have $p(p|\alpha_\epsilon, \beta_\epsilon) = \frac{p^{\alpha_\epsilon - 1}(1-p)^{\beta_\epsilon - 1}}{B(\alpha_\epsilon, \beta_\epsilon)}$.

The probability of the binary data D given p is $p(D|p, \alpha_\epsilon, \beta_\epsilon) = \Pi p^k (1-p)^{1-k}$. We call this likelihood.

According too Bayes Rule, using prior and likelihood, we can get the posterior, or the probability of p given D is proportional to likelihood times prior. $p(p|D, \alpha_\epsilon, \beta_\epsilon) \propto p(D|p, \alpha_\epsilon, \beta_\epsilon) * p(p|\alpha_\epsilon, \beta_\epsilon)$

To use the posterior for predicting new data, we set up the posterior predictive too be $p(Y_i^* = 1|D, \alpha_\epsilon, \beta_\epsilon) = \int p(Y_i^* = 1|p) * p(p|D, \alpha_\epsilon, \beta_\epsilon)dp$ and this is the probability we need for *yes* when creating synthetic binary data. To make use of it, we need to know the form of distribution of $Y_i^* = 1|D, \alpha_\epsilon, \beta_\epsilon$. Luckily, we know this follows a Bernoulli distribution because Beta distribution is the conjugate prior for Bernoulli distributions. We eventually get

$$Y_i^*|D, \alpha_\epsilon, \beta_\epsilon \sim \text{Bernoulli}(\frac{\alpha_\epsilon + \sum_{j=1}^n Y_j}{\alpha_\epsilon + \beta_\epsilon + n})$$

with $p(Y_i^* = 1|D, \alpha_\epsilon, \beta_\epsilon) = \frac{\alpha_\epsilon + \sum_{j=1}^n Y_j}{\alpha_\epsilon + \beta_\epsilon + n}$. The derivation process is shown below. We tune $\alpha_\epsilon = \beta_\epsilon = \frac{1}{e^{\epsilon/n_s} - 1}$ to ensure ϵ-differential privacy (McClure and Reiter 2012).

Derivation of Bernoulli Posterior Predictive Distribution

Prior:

$$p(p|\alpha_\epsilon, \beta_\epsilon) = \frac{p^{\alpha_\epsilon - 1}(1-p)^{\beta_\epsilon - 1}}{B(\alpha_\epsilon, \beta_\epsilon)} = \frac{p^{\alpha_\epsilon - 1}(1-p)^{\beta_\epsilon - 1}}{\frac{(\alpha-1)!(\beta-1)!}{(\alpha+\beta-1)!}}$$

Likelihood:

$$\begin{aligned} p(D|p, \alpha_\epsilon, \beta_\epsilon) &= \Pi p^k (1-p)^{1-k} \\ &= p^{\Sigma y_j}(1-p)^{n - \Sigma y_j} \end{aligned}$$

Posterior:

$$\begin{aligned} p(p|D, \alpha_\epsilon, \beta_\epsilon) &\propto p(D|p, \alpha_\epsilon, \beta_\epsilon) * p(p|\alpha_\epsilon, \beta_\epsilon) \\ &\propto p^{\Sigma y_j}(1-p)^{n - \Sigma y_j} * \frac{p^{\alpha_\epsilon - 1}(1-p)^{\beta_\epsilon - 1}}{B(\alpha_\epsilon, \beta_\epsilon)} \\ &\propto \frac{p^{\Sigma y_j + \alpha_\epsilon - 1}(1-p)^{n - \Sigma y_j + \beta_\epsilon - 1}}{B(\alpha_\epsilon, \beta_\epsilon)} \\ &\propto \frac{p^{\Sigma y_j + \alpha_\epsilon - 1}(1-p)^{n - \Sigma y_j + \beta_\epsilon - 1}}{B(\Sigma y_j + \alpha_\epsilon, n - \Sigma y_j + \beta_\epsilon)} \end{aligned}$$

Bernoulli posterior predictive:

$$p(Y_i^* = 1|D, \alpha_\epsilon, \beta_\epsilon) = \int p(Y_i^* = 1|p) * p(p|D, \alpha_\epsilon, \beta_\epsilon)dp$$

$$\begin{aligned} p(Y_j^* = 1|D, \alpha_\epsilon, \beta_\epsilon) &= \int p(Y_j^* = 1|p) * p(p|D, \alpha_\epsilon, \beta_\epsilon)dp \\ &= \int p * \frac{p^{\Sigma y_j + \alpha_\epsilon - 1}(1-p)^{n - \Sigma y_j + \beta_\epsilon - 1}}{B(\Sigma y_j + \alpha_\epsilon, n - \Sigma y_j + \beta_\epsilon)}dp \\ &= \int \frac{p^{\Sigma y_j + \alpha_\epsilon}(1-p)^{n - \Sigma y_j + \beta_\epsilon - 1}}{B(\Sigma y_j + \alpha_\epsilon, n - \Sigma y_j + \beta_\epsilon)}dp \\ &= \int \frac{p^{\Sigma y_j + \alpha_\epsilon}(1-p)^{n - \Sigma y_j + \beta_\epsilon - 1}}{\frac{(\Sigma y_j + \alpha_\epsilon - 1)!(n - \Sigma y_j + \beta_\epsilon - 1)!}{(\alpha+\beta+n-1)!}}dp \\ &= \int \frac{p^{\Sigma y_j + \alpha_\epsilon}(1-p)^{n - \Sigma y_j + \beta_\epsilon - 1}}{\frac{(\Sigma y_j + \alpha_\epsilon)!(n - \Sigma y_j + \beta_\epsilon - 1)!}{(\alpha_\epsilon + \beta_\epsilon + n)!} * \frac{\alpha_\epsilon + \beta_\epsilon + n}{\Sigma y_j + \alpha_\epsilon}}dp \end{aligned}$$

$$
\begin{aligned}
p(Y_j^* = 1|D, \alpha_\epsilon, \beta_\epsilon) &= \int \frac{p^{\Sigma y_j + \alpha_\epsilon}(1-p)^{n-\Sigma y_j+\beta_\epsilon-1}}{\frac{(\Sigma y_j+\alpha_\epsilon)!(n-\Sigma y_j+\beta_\epsilon-1)!}{(\alpha_\epsilon+\beta_\epsilon+n)!} * \frac{\alpha_\epsilon+\beta_\epsilon+n}{\Sigma y_j+\alpha_\epsilon}} dp \\
&= \int \frac{p^{\Sigma y_j + \alpha_\epsilon}(1-p)^{n-\Sigma y_j+\beta_\epsilon-1}}{B(\Sigma y_j + \alpha_\epsilon + 1, n - \Sigma y_j + \beta_\epsilon) * \frac{\alpha_\epsilon+\beta_\epsilon+n}{\Sigma y_j+\alpha_\epsilon}} dp \\
&= \frac{\Sigma y_j + \alpha_\epsilon}{\alpha_\epsilon + \beta_\epsilon + n} * \int \frac{p^{\Sigma y_j + \alpha_\epsilon}(1-p)^{n-\Sigma y_j+\beta_\epsilon-1}}{B(\Sigma y_j + \alpha_\epsilon + 1, n - \Sigma y_j + \beta_\epsilon)} dp \\
&= \frac{\Sigma y_j + \alpha_\epsilon}{\alpha_\epsilon + \beta_\epsilon + n} * \int Beta(\Sigma y_j + \alpha_\epsilon + 1, n - \Sigma y_j + \beta_\epsilon) dp
\end{aligned}
$$

which could be simplified to:

$$
p(Y_i^* = 1|D, \alpha_\epsilon, \beta_\epsilon) = \frac{\alpha_\epsilon + \sum Y_i}{\alpha_\epsilon + \beta_\epsilon + n}
$$

3.4 Differentially Private Trees

Decision trees are often used as a non-parametric method for prediction. However, privacy risks could be found in decision tree learning (Truex et al. 2017). By releasing the model and using the model for evaluation, sensitive information are exposed to inversion attacks.

An inversion attack refers to a situation when a hacker, given certain information about the training set, correctly predicts the full training data or the sensitive information in the training data. Mathematically, suppose our real training data set has X and their corresponding predicted value Y using model f. We have $f(X) = Y$. The entire form of X is unknown, but the Y under f is known. Suppose a hacker uses θ to infer information about the real training set. Let X' be the data set containing part or none of the information in the real training set. The hacker gets $f_\theta(X') = Y'$, with Y' correlated to Y (Fredrikson, Jha, and Ristenpart 2015). Based on this correlation, it is possible for the hacker to determine the entire form of X. Sensitive information in X is then compromised.

A decision tree, when released, is considered private when the original data used for training is not identifiable through inversion attack. Next, we will show the algorithm to generate a decision tree model to release that also achieves this goal.

Differentially Private Trees Algorithm

A decision tree for a categorical response variable works in a way as follows. The whole data set (root node) D is split based on attributes A in the explanatory variables. For any given attribute $a \in A$, D is split into disjoint subsets D_{a_v} for $v \in a$. We then compute for each D_{a_v} its Gini Index (Breiman et al. 1984), which measures how often a randomly chosen record to be incorrectly classified. D is split into D_{a_v} for $v \in a$ based on the largest Gini Index for a. This process is iterated for each D_{a_v} until a limit is reached, such as a maximum depth (Fletcher and Islam 2015). Then we have several leaf nodes. We classify them based on the occurances of each response variable level. We classify into the level that occurs the most frequent.

Here we only consider binary decision trees, which means the response variable is binary. To ensure binary decision trees differentially private, we use the following algorithm inspired by Bojarski et al. (2014). We take the tree fit from the original data. Then, we take out each leaf nodes, which are discrete subsets of the original data. Within each leaf node, we can calculate the probability of occurance of each level. The one with the larger probability becomes our classification result. We then add Laplacian noise on the probability of one of the levels. By subtracting this new probability from 1 we automatically get the probability of the other level. Now we use this set of new probabilities to sample a new set of response variable in this particular leaf node. This process is to be repeated for all leaf nodes. Then every element in the response variable is generated by sampling and their values may be modified from the original data. We use this new data to fit

the tree again. Note that to ensure utility, the new tree should have the same number of splits as the tree fit with original data.

When adding Laplacian noises to the probabilities, global sensitivity is no longer 1 which is used for generating counts or median. Global sensitivity for probability means the maximum change in probability caused by adding or removing one row of data. Here we suppose we had count of x of a certain level out of n data points. Our original probability is $\frac{x}{n}$ and can only change into the following four values: $\frac{x-1}{n-1}, \frac{x+1}{n+1}, \frac{x}{n-1}$, and $\frac{x}{n+1}$. Results show that $\frac{x-1}{n-1}$ has the largest difference among the four. Thus $\frac{x}{n} - \frac{x-1}{n-1} = \frac{n-x}{n(n-1)}$ is the global sensitivity, and this value will be used too draw noises from the Laplace distribution.

4 Synthetic Data

4.1 Creating Synthetic Data Set

Synthetic Data are created such that when finished, we have an entire synthetic data set rather than one variable. In previous sections we have introduced several methods synthetic column or a synthetic value such as count. In this section, we are going too introduce the way to create a whole synthetic data set.

A synthetic data set is created column by column. Typically the first column is created by random sample. Then, according to the sequence of synthesis, latter variables are synthesized using previous ones as predictors, which helps capture relationships. The sequence of synthesis is subject to changes to ensure a synthetic data set similar to the original data set. It is also possible to choose parametric or non-parametric methods for synthesis when predicting the latter columns, depending on the data type and correlation between variables. Whether null data are considered in synthesis is to be considered. We can create a synthetic data that captures the missing data characteristics in the original data. We can also set certain rules of synthesis, such as not generating negative numbers for age.

To help synthetic data set capture the trend better, it is necessary to consider these practices: 1) the sequence of synthesis is ranked from the variables with less missing values to those with more; 2) for the first several variables to synthesize, use random sample method. CART does not work well when there are few less correlated predictors; 3) for the variables we are interested in modeling, we synthesize them later so that they would have more predictors when being synthesized; 4) we do not use variables that are less correlated with or do not make sense in predicting the target variable as its predictors.

There are various ways of creating a synthetic data set. In this research project, we use a package in `R` called `synthpop` (Nowok, Raab, and Dibben 2016).

4.2 Synthetic Data Experiment on Healthy Minds Data Set

The Healthy Minds data set is the results collected from the Healthy Minds survey conducted at Wake Forest University as a part of the Healthy Minds Study (Healthy Minds Network, 2020). This study is an annual survey for the college and university populations focusing on mental healthy related issues.

First we are interested in releasing the number of international student responses among the 1631 responses using differential privacy methods. Here (Figure 8) Laplacian noise is added to the true count (128). Releasing a synthetic value with Laplacian noise will hide the information of the next person added to the data set.

We are also interested in knowing how Kristen-Eff Self-Compassion Scale short questions (Raes et al. 2010) influence depression diagnosis. To keep the data safe, we create a synthetic data set containing depression diagnosis (denoted `Depression`), a binary variable, and responses to the Compassion Scale questions (denoted `Comp_Scale`), 12 numerical variables for 12 questions. The order of synthesis is `Depression`, followed by each `Comp_Scale` questions, `Comp_Scale_Q1`... `Comp_Scale_Q12`. The method of synthesis is using CART (Breiman et al. 1984), and we take missing values into account. The tables below (Figure 9 and 10) show subsets of the original data and synthetic data.

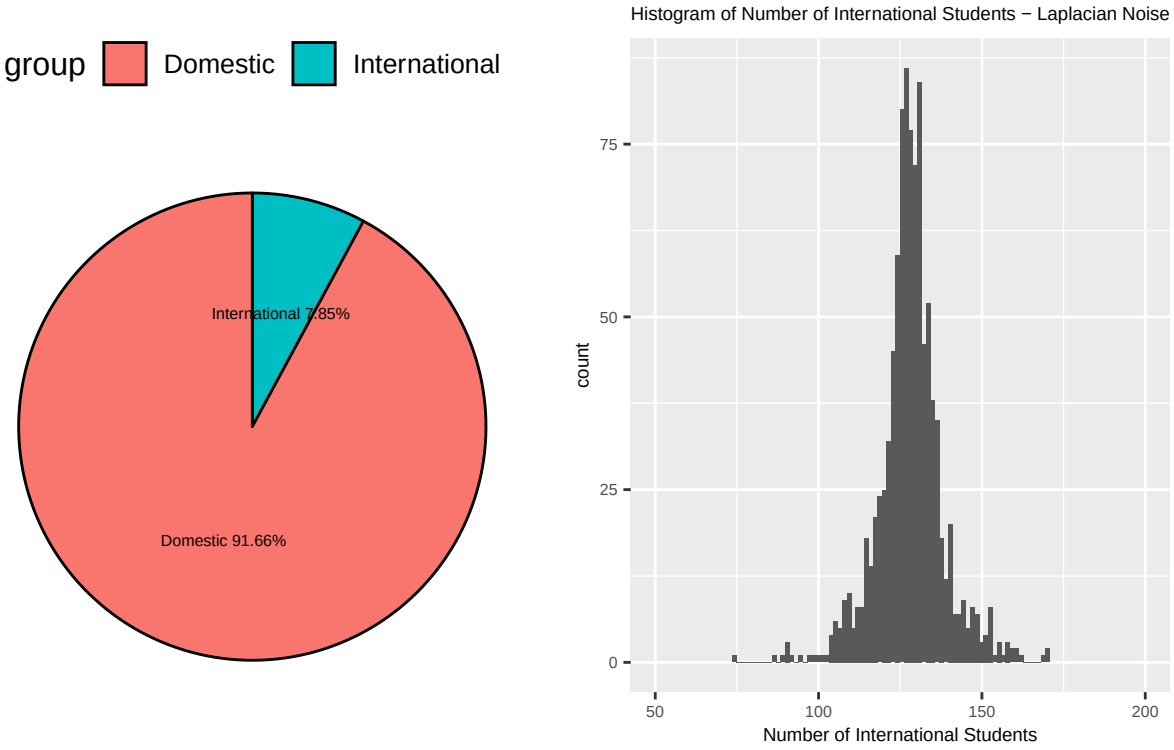

Figure 8: Shown are the pie chart of domestic and international student respoonses and a histogram of the distribution of count of international student reponses by adding 1,000 Laplacian noises.

	Depression	Comp_Scale_Q1	Comp_Scale_Q2	Comp_Scale_Q3
2	0	NA	NA	NA
3	0	3	2	3
4	0	4	1	3
5	0	3	4	4
6	1	1	4	4

Figure 9: Shown is the subset of the original Healthy Minds data set.

	Depression	Comp_Scale_Q1	Comp_Scale_Q2	Comp_Scale_Q3
2	0	NA	NA	NA
3	0	4	2	3
4	1	4	3	3
5	0	1	4	5
6	0	4	3	3

Figure 10: Shown is the subset of the synthetic Healthy Minds data set.

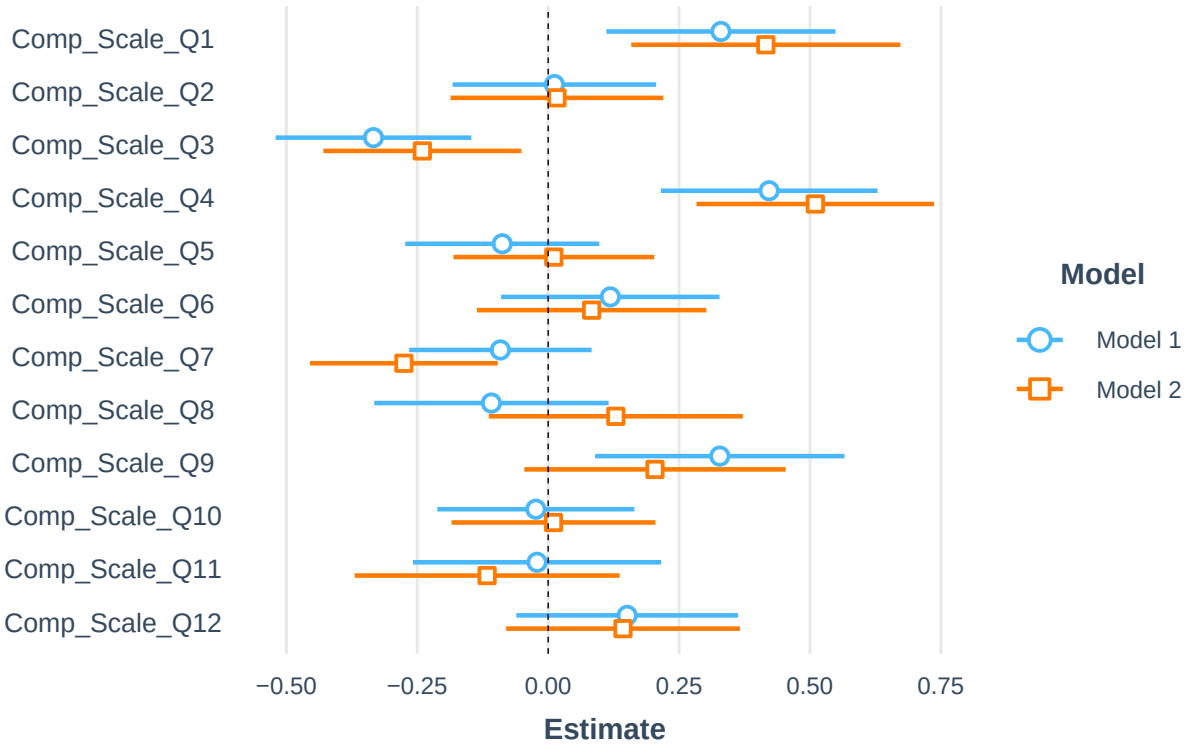

Figure 11: Shown is the forest plot showing confidence intervals of the logistic regression parameters of models fit with original and synthetic Healthy Minds data set

A logistic regression was fit to understand the relationship between `Depression` and `Comp_Scale_Q1`... `Comp_Scale_Q12`. With "Model 1" indicating the model fit using original data set and "Model 2" indicating the model fit using synthetic data set, the forest plot (Figure 11) shows that the confidence intervals of the logistic regression parameters overlap largely. This means the relationship is well preserved in the synthetic data set.

Since some confidence intervals include 0, we conclude that logistic regression is not a good method for understanding the relationship between `Depression` and `Comp_Scale_Q1`... `Comp_Scale_Q12`. A decision tree is a better moodel. Here we use differentially private trees as the method. Figure 12 shows a decision tree fit using original data. For explanation purposes, we use a tree with only 5 splits to show how a differentially private tree is created. In each leaf node, the probabilities indicate the probability of *diagnosed* or *not diagnosed*. We add Laplacian noise to the probability of *diagnosed* on each node, which enables us to sample a new set of *diagnosed* elements in the node. Fill the rest with *not diagnosed* to ensure the synthetic node is the same size as the original node. Since every node is discrete, creating synthetic nodes for all nodes essentially creates a synthetic column. Now use this synthetic column of `Depression` and the original `Comp_Scale_Q1`... `Comp_Scale_Q12` to fit the tree again.

We compare trees fit using different data sets. Fig 13 is a tree with 11 splits fit using original data. Fig 14 is a tree fit using synthetic data generated by `synthpop`. Fig 15 is a tree fit using data with Laplacian noises added to `Comp_Scale_Q1`... `Comp_Scale_Q12`. A differentially private tree is shown in Figure 16, which is in the same form as the original tree with 11 splits.

5 Discussion & Conclusion

According to the definition of differential privacy, even if the same value as the original is released by a differential privacy method, as long as the hacker knows that the result is generated using differential privacy

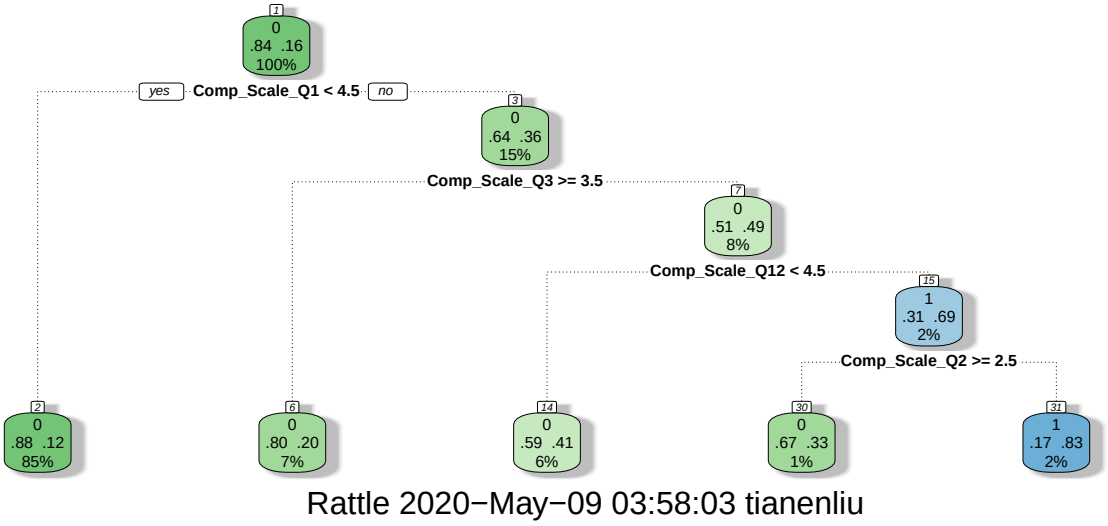

Figure 12: Shown is the decision tree with 5 splits fit using original data

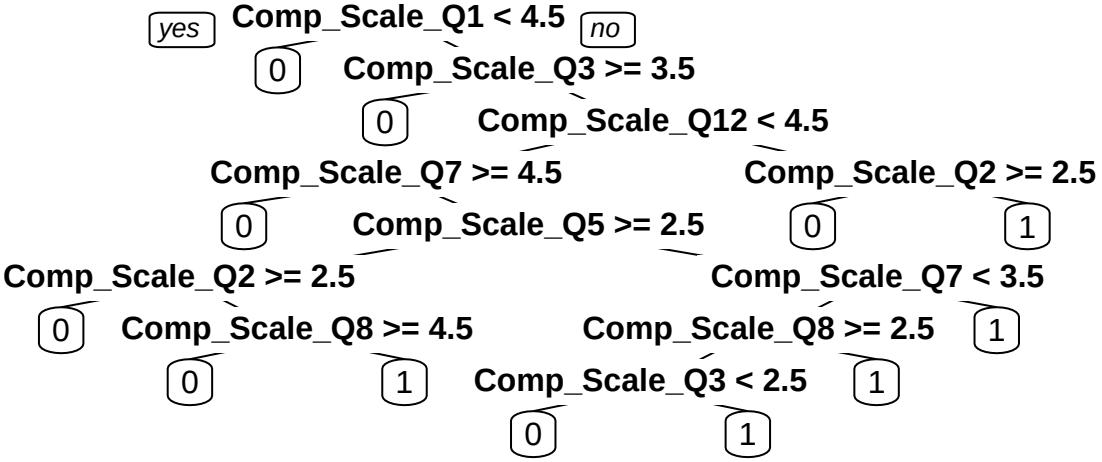

Figure 13: Shown is the decision tree with 11 splits fit using original data

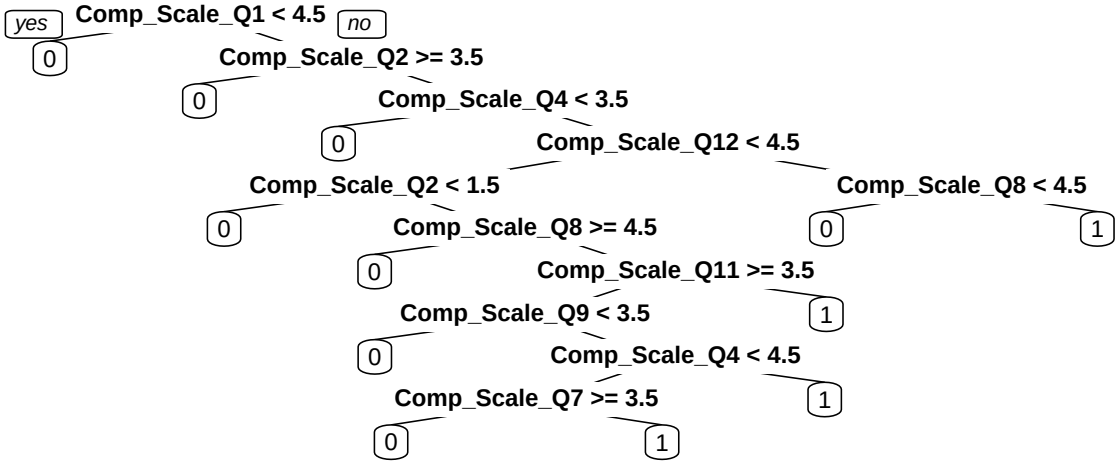

Figure 14: Shown is the decision tree fit using synthetic data generated by the synthpop package

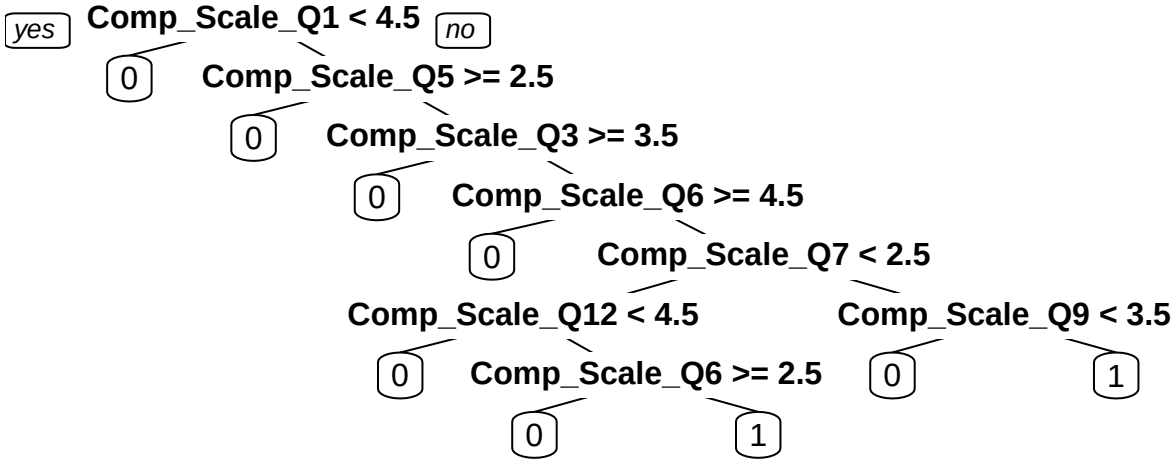

Figure 15: Shown is the decision tree fit using a data set with Laplacian noises added to the Compassion Scales responses

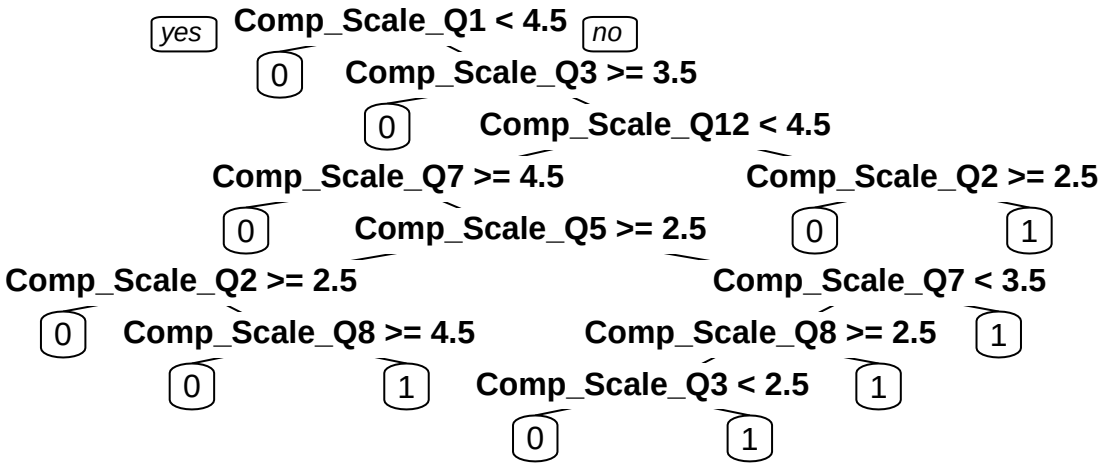

Figure 16: Shown is the differentially private decision tree

methods, he/she will not know the original data, and will not know the identity of an additional person added to data.

Of the four differential privacy methods, to keep a number private, such as count or median, Laplacian noise is the most convenient method to do so. To keep a binary data private, all four methods are able to create a synthetic version of the binary column. To keep a numerical data private, Laplacian noise and randomized response can modify numeric data.

Synthetic data does not guarantee differential privacy but the risks for data breach is little because most of the elements in the original data are changed. For $HealthyMinds$ data set, synthetic data preserves a great amount of utility when fitting a logistic regression. When fitting a tree model, synthetic data and Laplacian noise fail to capture the relationships of a tree. While differentially private trees can preserve the utility, we have to make sure that the model use original data and the model using data generated by differentially private trees equal number of splits.

References

Barth-Jones, Daniel C. 2012. "The 'Re-Identification' of Governor William Weld's Medical Information: A Critical Re-Examination of Health Data Identification Risks and Privacy Protections, Then and Now." In.

Bojarski, Mariusz, Anna Choromanska, Krzysztof Choromanski, and Yann LeCun. 2014. "Differentially- and Non-Differentially-Private Random Decision Trees."

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Monterey, CA: Wadsworth; Brooks.

Dwork, Cynthia, Aaron Roth, and others. 2014. "The Algorithmic Foundations of Differential Privacy." *Foundations and Trends in Theoretical Computer Science* 9 (3–4). Now Publishers, Inc.: 211–407.

Fletcher, Sam, and Md Zahidul Islam. 2015. "A Differentially Private Decision Forest." In *AusDM*.

Fredrikson, Matt, Somesh Jha, and Thomas Ristenpart. 2015. "Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures." In *Proceedings of the 22nd Acm Sigsac Conference on Computer and Communications Security*, 1322–33. CCS '15. New York, NY, USA: Association for Computing Machinery. doi:10.1145/2810103.2813677.

Lapowsky, Issie. 2018. "Facebook Exposed 87 Million Users to Cambridge Analytica." *Wired*. Conde Nast. https://www.wired.com/story/facebook-exposed-87-million-users-to-cambridge-analytica/.

McClure, David, and Jerome P. Reiter. 2012. "Differential Privacy and Statistical Disclosure Risk Measures: An Investigation with Binary Synthetic Data." *Trans. Data Privacy* 5 (3). Catalonia, Spain: IIIA-CSIC: 535–52. http://dl.acm.org/citation.cfm?id=2423656.2423658.

Nowok, Beata, Gillian M. Raab, and Chris Dibben. 2016. "synthpop: Bespoke Creation of Synthetic Data in R." *Journal of Statistical Software* 74 (11): 1–26. doi:10.18637/jss.v074.i11.

Raes, Filip, Elizabeth Pommier, Kristin D. Neff, and Dinska Van Gucht. 2010. "Construction and Factorial Validation of a Short Form of the Self-Compassion Scale." *Clinical Psychology & Psychotherapy* 18 (3): 250–55. doi:10.1002/cpp.702.

Task, Christine. 2014. "Privacy-Preserving Datamining: Differential Privacy and Applications."

Truex, S., L. Liu, M. E. Gursoy, and L. Yu. 2017. "Privacy-Preserving Inductive Learning with Decision Trees." In *2017 Ieee International Congress on Big Data (Bigdata Congress)*, 57–64.