Differentially Private Counts with Additive Constraints

Ziang Wang

May-November 2020

Note: This is a draft of the research supervised by Prof. Jerome Reiter, and it is also a draft of a part of my MS thesis.

1 Introduction

People are always concerned if they will be affected by allowing their information and data to be used in the study or research. Simply releasing them would have potential privacy risk. Differential privacy, is not only a terminology and a mathematical definition, but also a promise to all participants who are likely to contribute their information, so that researchers would not be able to learn much about an individual but are able to achieve their goals about that whole sample and the population [1].

Among all research and surveys conducted by organizations including the U.S. Census Bureau, medical agencies and establishments, and private internet companies, the Census is the most popular and influential one that requires all citizens to participate. Some common statistics, for example, Resident Population, Overseas Population and Total Employment are counts that are calculated based on the state level. For these statistics that are integers (counts), we need the counts at individual level to add up to counts at an upper level. For example, to research on how many students we have in a class, we take the number of boys and the number of girls, and these two numbers need to add up to the total number of students. We need not only true counts, but also counts with differential privacy to have this feature. However, applications of differentially private algorithms to each count are not guaranteed to produce counts that satisfy the additivity constraint.

One way to ensure the additivity constraint is to do so after perturbing the true counts. This is known as post-processing. In this research, I would present a post-processing procedure with differential privacy, which implements geometric mechanism and properties of multinomial distribution to add privacy to true counts. The basic idea is to posit a multinomial model for the released counts, using the noisy counts to determine the trial size and probabilities for each category of the multinomial distribution. To begin, I add noise to each count and the total with the geometric mechanism. In some of the post-processing algorithms, I take a sample from the multinomial distribution with trial size the same as the noisy total and probabilities determined from the noisy components. This guarantees that the samples add up to the total. Then the sample outputs would be released private counts. Another approach is to release values of a mode of the multinomial distribution instead of taking a sample.

Another part of the research is how a user gets inferences about the true counts given counts with differential privacy. After the agency releases the counts with differential privacy and post-processing to ensure additivity by using such methods, there should be a corresponding procedure to help users make such inferences.

The remainder of this paper is organized as follows. Section 2 introduces the definition, theorems and some derivations which are used in the post-processing procedure. Section 3 introduces the post-processing procedures. Section 4 provides three examples with outputs. Section 5 comments on outputs and results and has a discussion of future research.

2 Background

2.1 Differential Privacy and Geometric Mechanism

The formal definition of differential privacy [1][2] is given below:

Definition 1 *(Differential Privacy):*
A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{\mathcal{X}}$ is α - differentially private if for all $\mathcal{S} \subset$ Range $(\mathcal{M})$ and for all $x, y \in \mathbb{N}^{\mathcal{X}}$ such that $||x-y|| \leq 1$:

$$Pr[\mathcal{M}(x) \in \mathcal{S}] \leq exp(\alpha) \cdot Pr[\mathcal{M}(y) \in \mathcal{S}].$$

where the probability space is over the mechanism $\mathcal{M}$.

Given this definition, the geometric mechanism is applied in this project. Thus, the probability space is over a two-sided geometric distribution. In this case, if Δ is a random variable, then for every integer δ,

$$Pr[\Delta = \delta] = \frac{1-\alpha}{1+\alpha}\alpha^{|\delta|}.$$

One could think of this algorithm as a discrete version that adds random noise δ from a double exponential distribution(where $\alpha = \frac{1}{exp(\epsilon)}$). $\epsilon = 1$ is used in this paper.

With the geometric mechanism, we could then sample δ. Once δ is sampled, add δ to the true count (individual count / total) would lead to a privacy count. In this research, we call such counts generated from geometric mechanism with center as the true count as "Noisy Counts". Since we apply this way independently on each individual count and the total, the noisy total count does not equal the sum of all noisy individual counts. It would not be reasonable to simply use these noisy counts as our final privacy counts. This is the main reason why we need additional steps.

Since smaller δ has a higher probability to be sampled from, it's very unlikely to get a big δ. In order to implement the sampling process, I make the simplification that the output from the double geometric distribution ranges from -50 to 50. The noise can only be sampled in this range.

2.2 Posterior Distribution of the True Count

With sampled noisy counts, we apply Bayesian inference to find the posterior counts.

Suppose N_s is the true count for each component s. N is the true count for the total, i.e. $N = \sum_s N_s$.

Suppose $n_s = N_s + \delta_s$ is the noisy count for each individual s. $n = n + \delta$ is the noisy count for the total.

Based on section 2.1, we would have:

$$p(n \mid N) = p(\delta = n - N) = \alpha^{|n-N|} \cdot \frac{1-\alpha}{1+\alpha} \tag{1}$$

$$p(n_s \mid N_s) = p(\delta_s = n_s - N_s) = \alpha^{|n_s - N_s|} \cdot \frac{1-\alpha}{1+\alpha} \tag{2}$$

The posterior distribution (Bayesian inference) of the true count N given the noisy count sampled from the geometric mechanism is needed. By Bayes' theorem, we could find the posterior distribution of N_s:

$$p(N_s \mid n_s) = \frac{p(n_s, N_s)}{p(n_s)} = \frac{p(n_s \mid N_s)p(N_s)}{p(n_s)} = \frac{p(n_s \mid N_s)p(N_s)}{\int_{N_s} p(n_s \mid N_s)p(N_s)dN_s} \tag{3}$$

In reality, we need to make sure that the posterior N is the sum of posterior N_s for all s. Then $p(N \mid n)$ would be a complex density, since it relates with all sampled posterior N_s. So, I would use some approximations to the posterior distribution of N, as described in section 2.2.1 and 2.2.2.

2.2.1 Approximation based on Independence Assumptions

We assume the total count and component counts are all independent, even though in reality they are not..

Assume that N and N_s are uniformly distributed (Example: Unif(0, 1000). $p(N)$ and $p(N_s)$ is a constant). Then the posterior distribution would be:

$$p(N \mid n) \propto p(N) \cdot p(n \mid N) \propto p(n \mid N) = \alpha^{|N-n|} \cdot \frac{1-\alpha}{1+\alpha} \tag{4}$$

$$p(N_s \mid n_s) \propto p(N_s) \cdot p(n_s \mid N_s) \propto p(n_s \mid N_s) = \alpha^{|N_s-n_s|} \cdot \frac{1-\alpha}{1+\alpha} \tag{5}$$

For independent case, we apply this way independently on each of the individual noisy counts and their total noisy counts. Note that the posterior total count doesn't necessarily equal the sum of posterior counts of all components. We would lose the connection between the posterior total count and the posterior component counts. However, it is much easier to implement this way to find the posterior distributions, and we could have computational simplicity.

2.2.2 Approximation based on Summed Noisy Totals

In addition to the noisy total, the noisy components provide information about the true total. Thus, we may be able to improve inferences about the total by using both sets of noisy counts. I now describe a posterior inference procedure that takes advantage of this information.

Let $n^* = \sum_s n_s$, the sum of all noisy counts. Then define η such that:

$$n^* = \sum_s n_s = N + \eta$$

Note that n^* and n are independent. We sample n based on equation (1), and sample n_s for each component s based on equation (2). Since n^* only depends on n_s, and equation (1) and equation (2) are independent, N^* and n are also independent.

Assume that N follows a uniform distribution (Example: Unif(0, 1000)). Then the posterior distribution of the total count would be:

$$p(N \mid n, n^*) \propto p(n, n^* \mid N)p(N) \propto p(n \mid N) \cdot p(n^* \mid N) \cdot p(N) \tag{6}$$

$p(N)$ is a constant, when assuming a uniform distribution.

$p(n \mid N)$ is the probability of sampling out $n - N$ based on equation (1).

$p(n^* \mid N)$ is the probability of sampling out $\eta = n^* - N$ from the sum of three independent double geometric distribution. Given all n_s, we could calculate one corresponding η. The probability would be proportional to the multiple of all probabilities sampling out n_s. We iterate to generate a list of values of η with corresponding probabilities. Then we could simulate the distribution by normalizing these multiple of probabilities to estimate $p(n^* \mid N)$.

Apply (5) to find the posterior distribution of N_s for each component.

2.3 Mode of the multinomial distribution

Randomly sampling from a distribution always has uncertainty. Instead of sampling, we may consider straightly use the mode of the distribution. We now introduces the Finucan's algorithm [3] to find the mode of a multinomial distribution.

Let $\lfloor . \rfloor$ to denote the floor function that finds the integer part.

Theorem 1 *(Finucan's algorithm):*

Given the sample size m and the S-length vector of probabilities $\mathbf{p}$,
consider initial values $k_i = \lfloor (m + r/2)p_i \rfloor$ and $m_0 = \sum_{i=1,..r} k_i$
Set $f_i = (m + r/2)p_i - k_i$

1. *if $m_0 < m$: Define $q_i = \frac{1-f_i}{k_i+1}$ for $i = 1, ..., S$*

$$\begin{aligned} &\textit{While } \; m_0 < m : \\ &\quad q_a = \min_{i=1,\dots r} q_i \\ &\quad k_a = k_a + 1 \\ &\quad f_a = f_a - 1 \\ &\quad q_a = \frac{1 - f_a}{k_a + 1} \\ &\quad m_0 = m_0 + 1 \end{aligned}$$

Continue updating all parameters until $m_0 = m$

2. *if $m_0 > m$: Define $q_i = \frac{f_i}{k_i}$ for $i = 1, ..., S$*

$$\begin{aligned} &\textit{While } \; m_0 > m : \\ &\quad q_a = \min_{i=1,\dots r} q_i \\ &\quad k_a = k_a - 1 \\ &\quad f_a = f_a + 1 \\ &\quad q_a = \frac{f_a}{k_a} \\ &\quad m_0 = m_0 - 1 \end{aligned}$$

Continue updating all parameters until $m_0 = m$

3. *$(k_1, ...k_s)$ is a mode for $MD(m, \mathbf{p})$*

3 Description of the procedure

3.1 Generate the Privacy Count

N_s is the confidential true count of an individual s, and N is the confidential true total count. Assume δ ranges from -50 to 50. Below I describe three algorithms to generate the released counts. The first two algorithms release samples from multinomial distributions. The third algorithm uses Finucan's method to release a mode of the multinomial distribution.

3.1.1 Algorithm 1 and Algorithm 2

1. Add independent geometric mechanism noise to the total N. Call the noisy count n.
2. Add independent geometric mechanism noise to all N_s. Call each of the noisy count n_s.

3.a. Under the approximation based on independence, take a draw from the posterior distribution of N given n based on (4). This draw must be positive. If $n - 50 < 0$, indicating a negative draw, re-parametrize with $p(N \mid n) = \frac{p(n|N)p(N)}{\sum_{n-\delta>0} p(n|N)p(N)}$ for each possible positive N. Take a draw with these new probabilities. Call it N^i.

3.b. Under the approximation based on the summed noisy totals, take a draw from the posterior distribution of N given n, n^* based on (6). This draw must be positive. If $n - 50 < 0$, indicating a negative draw, re-parametrize with $p(N \mid n, n^*) = \frac{p(n|N) \cdot p(n^*|N) \cdot p(N)}{\sum_{n-\delta>0} p(n|N) \cdot p(n^*|N) \cdot p(N)}$ for each possible positive N. Take a draw with these new probabilities. Call it N^c.

4. For each component s, take a draw from the posterior distribution of N_s based on (5). This draw must be positive. If $n_s - \delta_s < 0$, then re-parametrize with $p(N_s \mid n_s) = \frac{p(n_s|N_s)p(N)}{\sum_{n_s-\delta>0} p(n_s|N_s)p(N_s)}$ for each possible positive N_s. Take a draw with these new probabilities. Call it N_s^*.
5. Use the fact that the distribution of Poisson random variables given their sum is a multinomial distribution. Design two algorithms to generate final results:

   - Algorithm 1:

     Generate $\tilde{\mathbf{N}}^{1i}$ (Approximation based on Independence): Compute the probabilities $\theta_s^1 = \frac{N_s^*}{\sum_1^S N_s^*}$ for each component s. Then sample new values from the multinomial distribution with sample size N^i and S-length vector of probability parameters $(\theta_1^1, \theta_2^1, ..., \theta_S^1)$. Call the generated S-length vector $\tilde{\mathbf{N}}^{1i} = (\tilde{N}_1^{1i}, \tilde{N}_2^{1i}, ..., \tilde{N}_S^{1i})$. This is the output for privacy counts.

     Generate $\tilde{\mathbf{N}}^{1c}$ (Approximation based on Summed Noisy Counts): Using θ_s^1 for each component s, sample new values from the multinomial distribution with sample size N^c and S-length vector of probability parameters $(\theta_1^1, \theta_2^1, ..., \theta_S^1)$. Call the generated S-length vector $\tilde{\mathbf{N}}^{1c} = (\tilde{N}_1^{1c}, \tilde{N}_2^{1c}, ..., \tilde{N}_S^{1c})$. This is the output for privacy counts.

- Algorithm 2:

  Generate $\tilde{\mathbf{N}}^2$: If n_s is negative for component s, replace it with the posterior mode of $p(N_s \mid N_s)$ for s, and call the posterior mode n_s. Then compute the probabilities $\theta_s^2 = \frac{n_s}{\sum_1^S n_s}$ for each s. Then sample new values from the multinomial distribution with sample size n and s-length vector of probability parameters $(\theta_1^2, \theta_2^2, ..., \theta_S^2)$. Call the generated S-length vector $\tilde{\mathbf{N}}^2 = (\tilde{N}_1^2, \tilde{N}_2^2, ..., \tilde{N}_S^2)$. This is the output for privacy counts.

3.1.2 Algorithm 3

1. Repeat step 1, step 2, step 3.a., and step 4 in section 3.1.1.
2. Design Algorithm 3 to generate final results $\tilde{\mathbf{N}}^3 = (\tilde{N}_1^3, \tilde{N}_2^3, ..., \tilde{N}_S^3)$. i represents the case under the approximation based on independence.

   - Algorithm 3: Mode of the multinomial distribution

     Generate $\tilde{\mathbf{N}}^3$: Use θ_s^2 for each s. Then apply Finucan's algorithm as described in section 2.3. With the context of Theorem 1, $m = n$, and $\mathbf{p} = (\theta_1^2, \theta_2^2, ...\theta_S^2)$. Call the result generated from Finucan's algorithm $\tilde{\mathbf{N}}^3$. This is the output for privacy counts.

3.2 Posterior Inferences for the Confidential Counts

After privacy counts are generated by the above procedure and released, users need to have a method to estimate what confidential true counts could be given these privacy counts.

Let $\tilde{N}$ represent the privacy output for the total, and $\tilde{\mathbf{N}} = (\tilde{N}_1, \tilde{N}_2, ..., \tilde{N}_S)$ represent the privacy outputs for all components.

To estimate the possible true counts, we need to find the distribution of true counts given released privacy counts. Assume each individual true count and their total are uniformly distributed. Assume each N_s and N follow independent uniform distributions. Then $p(N_1, ..., N_s, N)$ would be a multiple of constants. This posterior distribution $p(N_1, ..., N_S, N \mid \tilde{N}_1, ..., \tilde{N}_S, \tilde{N})$ could be written as:

$$\begin{aligned} p(N_1, ..., N_S, N \mid \tilde{N}_1, ..., \tilde{N}_S, \tilde{N}) &\propto f(\tilde{N}_1, ..., \tilde{N}_S, \tilde{N} \mid N_1, ...N_S, N) p(N_1..., N_S, N) \\ &\propto \sum_{n_1} ... \sum_{n_S} \sum_{n} f(\tilde{N}_1, ..., \tilde{N}_S, \tilde{N}, n_1, ..., n_S, n \mid N_1, ..., N_S, N) \\ &\propto \sum_{n_1} ... \sum_{n_S} \sum_{n} f(\tilde{N}_1, ...\tilde{N}_S, \tilde{N} \mid n_1, ..., n_S, n, N_1, ..., N_S, N) p(n_1, ..., n_S, n \mid N_1, N_2, N) \end{aligned}$$

$p(n_1, ..., n_s, n \mid N_1, N_2, N)$ is the probability that sampling out $n_1, ...n_s$ and n given all true counts and their total, which is proportional to the multiple of all those corresponding densities of geometric distribution based on equation (1) and equation (2).

We need to study the distribution of $\tilde{N}_1, ...\tilde{N}_S, \tilde{N} \mid n_1, ..., n_S, n, N_1, ..N_S, N$, which is having all privacy counts given all true counts and noisy counts.

To start it, I implement the brute force way to study the simplest case when $S = 2$. For simplicity, I use the privacy counts generated by Algorithm 2 and Algorithm 3, i.e. $\mathbf{N}^2$ and $\mathbf{N}^3$ as an example.

$$p(N_1, N_2, N \mid \tilde{N}_1, \tilde{N}_2, \tilde{N}) \propto \sum_{n_1} \sum_{n_2} \sum_{n} f(\tilde{N}_1, \tilde{N}_2, \tilde{N} \mid n_1, n_2, n, N_1, N_2, N) p(n_1, n_2, n \mid N_1, N_2, N)$$

The procedure to simulate the distribution $\tilde{N_1}, \tilde{N_2}, \tilde{N} \mid n_1, n_2, n, N_1, N_2, N$ is the following:

- Initialize some possible combinations of N_1, N_2 by taking values from $(-30 + \tilde{N_1}, 30 + \tilde{N_1})$, $(-30 + \tilde{N_2}, 30 + \tilde{N_2})$. Truncate the intervals and only consider non-negative values when the interval includes negative values. Then $N = N_1 + N_2$.

- For each set of possible combination of true counts N_1, N_2 from step 1, initialize some possible combinations of n_1, n_2 by taking values from $(-30 + N_1, 30 + N_1)$, $(-30 + N_2, 30 + N_2)$. Note that $\tilde{N} = n$ in Algorithm 2 and Algorithm 3. Thus, $n = \tilde{N}$ must be sampled given N, which gives a fixed probability.

- For each set of possible combination of noisy counts n_1, n_2, find the probability that generate given privacy counts $\tilde{N_1}, \tilde{N_2}$.

To implement this method, I create a list that makes all possible combinations of N_1, N_2, N as the elements. For each element, we create a grid with rows representing n_1 and columns representing n_2. Then the output would be a list of $61 \cdot 61$ elements with each element a 61-by-61 grid. We sum up all probabilities for each list element and then normalize them to get estimation of probabilities to have each pair of N_1, N_2 as true counts.

4 Applications / Results

4.1 Results of the Procedure to Generate Privacy Counts

To examine different algorithms, I iterate the procedure described in section 3.1 10000 times for both Example 1 in section 4.1.1 and Example 2 in 4.1.2. Then we could have 10000 sets of outputs for each algorithm in each example. We would examine the distributions of all outcomes from these three algorithms.

4.1.1 Example 1: Simulated Data

We simulated counts of 50 components including some small counts. In particular, we generate three values of 0; one value each of 1,2 and 3; five values between 15 and 60; and the remainder of values scattered uniformly between 60 and 1000. The counts are displayed in Figure 1.

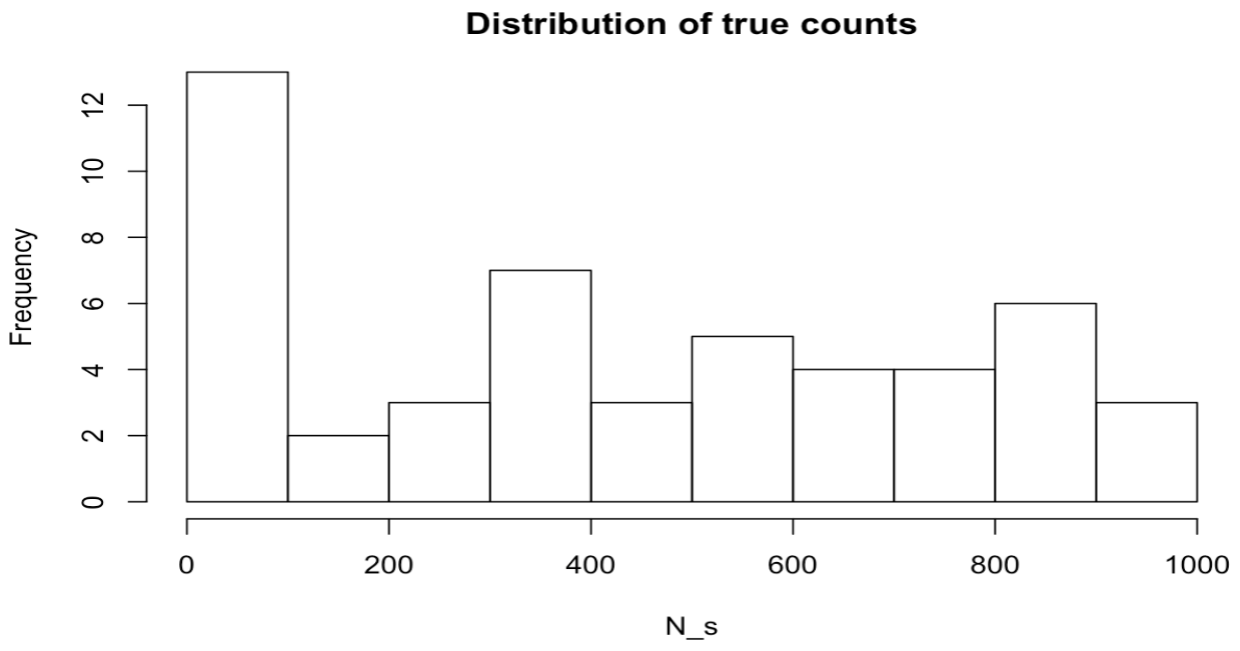

Figure (1) Distribution of true counts N_s

To show the performance, we picked 5 representative counts: N_{min}: the minimum, N_{q1}: the first quantile, N_{med}: the median, N_{q3}: the third quantile, and N_{max}: the maximum of the confidential counts. We calculate the mean and variance for each case of each algorithm.

The total N is 21249. Five representative true counts are $N_{min} = 0$, $N_{q1} = 60$, $N_{med} = 382$, $N_{q3} = 736$, $N_{max} = 977$.

By iterating the procedure in section 3.1 10000 times, we got the mean and variance of those 5 representative counts from three algorithms shown in Table (1). Variances are in the parentheses. (ind) refers to the case under the approximation based on independence, and (summed) refers to the case under the approximation based on the summed noisy totals.

Count	N_{min}	N_{q1}	N_{med}	N_{q3}	N_{max}
Algorithm 1 (ind)	0.9(2.4)	60(63)	382(375)	736(695)	977(940)
Algorithm 1 (summed)	0.9(2.5)	60(63)	382(380)	736(714)	977(936)
Algorithm 2	0.4(1.2)	60(62)	382(380)	736(708)	977(932)
Algorithm 3	0.4(0.7)	60(2)	382(2)	736(2)	977(2)

Table (1) Mean values and variances of 5 representative counts of Example 1

From Table (1), it's found that mean values for all algorithms are similar and very close to the true counts. For small count $N_{min} = 0$, since only positive number could be sampled out, mean values tend to be close to 1. For the variances, it's found that there is no big difference among all algorithms for $N_{min} = 0$, but variances are much different for counts that are big. Algorithm 2 performs slightly better than two cases of Algorithm 1 overall, but it's not obvious to conclude which case of which algorithm has more centered distribution. Variances are sensitive and are close to the true value for Algorithm 1 and Algorithm 2. Algorithm 3 has much smaller variances, and variances are similar even across different counts.

4.1.2 Example 2: Apportionment Population And Number of Representatives (2010 Census)

The data is about the Apportionment Population and number of representative that were collected in the U.S. 2010 Census by the U.S. Census Bureau. We chose the first 50 states on the list, and most counts are small counts ranging from 0 to 50.

N_s are shown in Table (2):

Count	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16	18	27	36	53	435
Instances	7	5	3	6	3	2	2	4	4	1	1	1	1	2	1	2	2	1	1	1

Table (2) Mean values of 5 representative counts of Example 2

To show the performance, we picked 5 representative counts: N_{min}: the minimum, N_{q1}: the first quantile, N_{med}: the median, N_{q3}: the third quantile, and N_{max}: the maximum of the confidential counts. We calculate the mean and variance for each case of each algorithm.

The total N is 863. Five representative true counts are $N_{min} = 1$, $N_{q1} = 2$, $N_{med} = 6$, $N_{q3} = 11$ and $N_{max} = 435$.

By iterating the procedure in section 3.1 10000 times, we got the mean and variance of those 5 representative counts from three algorithms shown in Table (3). Variances are in the parentheses. (ind) refers to the case under the approximation based on independence, and (summed) refers to the case under the approximation based on the summed noisy totals.

Count	N_{min}	N_{q1}	N_{med}	N_{q3}	N_{max}
Algorithm 1 (ind)	1.5(3.5)	2.2(4.7)	6.0(9.3)	11.0(14.0)	432.5(256.8)
Algorithm 1 (summed)	1.5(3.5)	2.2(4.9)	6.0(9.2)	11.0(14.5)	432.5(259)
Algorithm 2	1.1(2.2)	2.1(3.6)	6.0(7.9)	11.0(12.6)	434.2(235.1)
Algorithm 3	1.0(1.1)	2.0(1.5)	5.9(1.8)	11.0(1.8)	437.6(19.2)

Table (3) Mean values and variances of 5 representative counts of Example 2

Basically, we have similar patterns as in Example 1. Mean values for all algorithms are similar and very close to the true counts. Variances for Algorithm 1, and 2 are similar. Algorithm 2 has slightly better performance overall. Algorithm 3 has the smallest variances among all algorithms.

From Example 1 and Example 2, Algorithm 3 has much better performance in terms of the spread of results. Samples are much more concentrated around the mean value. If we take only one draw from Algorithm 4, our privacy data would be very similar to the true confidential data.

In addition, there is no big difference between the case of approximation based on independence and the case of approximation based on summed noisy totals. Results may vary a little when taking the sample, but it turns out to be almost no difference. We don't find the approximation based on summed noisy totals have better results than the case based on independence.

4.2 Results of posterior inferences for the Confidential Counts

As described in section 3.2, we use a simple example with $\tilde{N}_1 = 250$, $\tilde{N}_2 = 357$, and $\tilde{N} = 607$. Table (5) and Table (6) provide possible combinations N_1 and N_2 which have top 10 highest posterior probabilities with privacy counts generated by Algorithm 2 and Algorithm 3, correspondingly.

N_1	N_2	N	probability
250	357	607	0.03
251	356	607	0.03
249	358	607	0.03
252	355	607	0.03
248	359	607	0.03
253	354	607	0.02
247	360	607	0.02
254	353	607	0.02
246	361	607	0.02
255	352	607	0.02

Table (4) Distributions of true counts with privacy counts generated from Algorithm 2

N_1	N_2	N	probability
250	357	607	0.21
251	356	607	0.13
251	357	608	0.08
250	356	606	0.08
249	358	607	0.07
250	358	608	0.05
249	357	606	0.05
251	358	609	0.03
249	356	605	0.03
252	355	607	0.03

Table (5) Distributions of true counts with privacy counts generated from Algorithm 3

Both Algorithms provide similar posterior distributions for true counts. The estimations with the highest probability are the same for both Algorithms, and they are exactly the same as given privacy counts. It's found that estimations with high probabilities are around the given privacy accounts.

The main difference is that the highest probability for Algorithm 2 is about 0.03, which doesn't stand out from other combinations. These top 10 combinations have very close probabilities and their estimated sum is the same $N = 607$. For Algorithm 3, the highest probability is about 0.21, which is quite bigger than other probabilities of other combinations.

5 Discussion / Conclusion

5.1 The Procedure to Generate the Privacy Counts

The means of the distributions of each representative counts are almost the same as true confidential counts, and Algorithm 3 using the mode of the multinomial distribution could generate results that are close to the true confidential counts most of the time. This procedure also works for counts that are zero. After doing the adjustment when the sample space could include negative counts, it is most likely to give out values that are 0 or 1. The results of Example 1: simulated data and Example 2: real data indicate the procedure has stable performance overall. Algorithm 2 performs slightly better than Algorithm 1. Algorithm 3 has much more centered outputs compared with other two algorithms. There is no significant difference between case of approximation based on independence and the case of approximation based on summed noisy totals.

There are also some potential issues with this procedure. If a set of true counts has large quantities of counts that are 0 and only a few counts that are big, results of these large counts would be affected by the large quantities of 0. In the multinomial distribution, the sampling probability of each s depends on how large the noisy count n_s or posterior count N_s^*. When most true confidential counts are 0, a lot of their corresponding n_s or N_s^* may not be zero. These counts whose n_s or N_s^* are not 0 will take a portion from the total sample size. Thus, the final results of some counts whose true value are 0 may not be 0. The influence would be the final results of those big counts must be smaller. It's less likely that the final output could be the true value or closely around the true value.

We notice that sampling from the multinomial distribution would have more variability. This would be the reason why variances of Algorithm 3 are much smaller and stable compared with other two algorithms.

5.2 Posterior Inferences for the Confidential Counts

Based on section 4.2, we find for the case which only has two counts, the posterior distributions of true counts when implementing Algorithm 2 is more spread out, while the posterior distributions when implementing Algorithm 3 is more centered. In terms of the highest probability, a reasonable final estimations for true counts would be the same as the given privacy counts for both Algorithms.

If users could have estimations that are very close to true confidential counts, it would require the privacy counts to be as close as the true confidential counts. Though the distribution of privacy counts from Algorithm 1 and 2 have estimated mean values that are very close to the true confidential counts, their large variances may make privacy counts very different from true counts by implementing the procedure once. After users implement this method, their final estimations would be very close to the privacy counts, which are quite different from true confidential counts. Thus, Algorithm 3 is the best algorithm that generates privacy counts close to the true counts without large variability. Then it's likely that users would have a good estimation of true confidential counts.

5.3 Future Study

I may also consider doing posterior inferences on privacy counts generated by both cases of Algorithm 1, though I expect results would be similar to results of Algorithm 2.

The next step of the research is mainly how users estimate the posterior distributions of true confidential counts with $S > 2$. Besides the complicated brute force way, we would think of techniques like MCMC to estimate this posterior distribution.

References

[1] Dwork, C. and Roth A. (2014), The Algorithmic Foundations of Differential Privacy, The essence of Knowledge, Foundations and Trends in Theoretical Computer Science, Vol.9, Nos.3-4, 211-407

[2] Ghosh, A. and Toughgarden, T. and Sundararajan, M (2012), Universally Utility-Maximizing Priacy Mechanisms, Soceity for Industrial and Applied Mathematics, Vol. 41, No.6, pp. 1673-1693

[3] Le Gall, F. (2003), Determination of the Modes of a Multinomial Distribution, Statistics Probablity Letters 62, 325-333