dapper: Data Augmentation for Private Posterior Estimation in R

by Kevin Eng, Jordan A. Awan, Ruobin Gong, Nianqiao Phyllis Ju, and Vinayak A. Rao

Abstract This paper serves as a reference and introduction on using the dapper R package. It is an MCMC sampling framework which targets the exact posterior distribution given privitized data. The goal of this package is to provide researchers a tool for exploring the impact of different privacy regimes on a Bayesian analysis. A strength of this framework is the ability to target the exact posterior in settings where the likelihood is too complex to analytically express.

1 Introduction

Differential privacy provides a rigorous framework for protecting confidential information (Dwork et al. 2006). In this framework, privacy is obtained through the injection of random noise in the data analysis workflow. It has served as the theoretical foundation for recent advances in privacy technology. Several high profile examples include Apple (Tang et al. 2017), Google (Erlingsson, Pihur, and Korolova 2014), Microsoft (Ding, Kulkarni, and Yekhanin 2017), and the U.S. Census Bureau (Abowd 2018).

There are several approaches for constructing or modifying a data analysis workflow to provide privacy guarantees, which loosely fall into three categories: direct, query, and dissemination. The approaches mainly differ in where in the workflow privacy noise is injected. In the dissemination setting, data needs to be released to the public, and privacy is maintained by first perturbing the data with random noise before being released. There is strong interest in further developing this approach because data curators are often interested in releasing data to the public without knowledge of how their data will be used or analyzed. In this scenario, it can be impossible to apply the direct or query method because they require knowing before hand how the data will be analyzed. For a recent example of a high profile application of the dissemination approach see the the U.S. Census Bureau's `TopDown` algorithm (Bureau 2023). (Ji, Lipton, and Elkan 2014) provide a nice survey of the direct and query based methods applied to common machine learning algorithms.

Correctly carrying out statistical inference in the dissemination setting requires adjusting statistical workflows to account for privacy noise. As an example, one instance of implementing the dissemination approach for tabular data involves directly adding independent, random error to each cell. For regression models, this corresponds to having data with measurement errors in the covariates. This, unfortunately, violates the assumptions of most statistical models. In the presence of such errors, standard estimators can exhibit significant bias and incorrect uncertainty quantification (Gong 2022) (Karwa, Kifer, and Slavković 2015a) (Wang et al. 2018). These issues are a serious concern for researchers (Santos-Lozada, Howard, and Verdery 2020) (Kenny et al. 2021) (Winkler et al. 2021). Therefore, developing privacy aware statistical workflows is necessary in order for science and privacy to coexists.

Unfortunately, making the necessary adjustments poses formidable mathematical challenges (Williams and Mcsherry 2010), even for seemingly simple models like linear regression. The difficulty can be seen from considering the marginal likelihood that results from correctly accounting for the injected privacy noise. This function is often analytically intractable and as a result, it is difficult or impossible to apply common maximum likelihood methods to derive estimators. In particular, the marginal likelihood can involve a complex integral where it is not possible to even evaluate the likelihood at a point. This makes gradient based methods, like the Newton-Raphson method, impractical. And approximating the likelihood can be computationally unfeasible since the integral is also of high dimension. Few tools are available to researchers to address these issues, and their absence is a serious barrier to the wider adoption of dissemination methods.

The `dapper` package provides a tool for conducting valid statistical inference in the presence of privacy noise. It implements the Bayesian framework proposed in (Ju et al. 2022). This framework describes how to modify an existing Bayesian model to account for privacy noise. `dapper` serves as user-friendly interface for implementing the framework in R. It allows the user to specify a sampler from an existing Bayesian model and automatically constructing a valid posterior sampler that accounts for the added privacy noise. Compared to other Bayesian approaches, the `dapper` framework requires no specialized knowledge for tuning samplers since there is no tuning parameter, and when the record additivity is satisfied (see section 6), the algorithmic run time complexity is similar to the non-private analysis. The goal of `dapper` is to provide a fast, flexible, and user-friendly way to analyze the impact of privacy mechanisms.

The rest of this article is organized as follows. Section 2 covers the necessary background to understand the mathematical notation and ideas used throughout the paper. Section 3 goes over the main algorithm without going into mathematical detail, for specifics see (Ju et al. 2022). Section 4 provides an overview of the dapper package and discusses important implementation details. Section 5 contains two example of how one might use the package to analyze the impact of adding noise for privacy. The first example goes over a typical odds ratio analysis for a 2×2 table and the second example covers a linear regression model.

2 Background

Let $x = (x_1, \ldots, x_n) \in \mathcal{X}^n$ represent a confidential database containing n records. Usually, the goal of collecting data is to learn some characteristic about the underlying population. To accomplish this task, a common approach is to assume the population is represented by some statistical model $f(\cdot \mid \theta)$. It is often the case that some function of θ has relevant meaning to the scientific question at hand. In this setting, learning characteristics of a population reduces to learning about θ.

In the Bayesian statistical framework, learning about θ is accomplished by drawing samples from the posterior $p(\theta \mid x) \propto f(x \mid \theta)p(\theta)$. For large data sets, it is common to work with a summary statistic $s = s(x)$ that has much smaller dimension than the original data. Doing so can greatly simplify calculations. In general, there can be information loss with using summary statistics, but for models with a sufficient statistic, there is no loss. Curators of large databases often use summary statistics to publicize data since it allows them to efficiently communicate information contained in large data sets. For this reason, summary statistics are a natural target for dissemination based privacy approaches.

Differential Privacy

While a summary statistic can already partially anonymize data, it is still possible to deduce information about an individual entry depending on the distribution of x. Differential privacy solves this problem by taking a summary statistic s, and adding noise to it to produce a noisy summary statistic s_{dp}. While this method is not new [insert citation on swapping and randomized responses], differential privacy provides a rigorous framework for specify where and how much noise to add.

We now describe the ϵ-DP privacy framework in more detail. For the noisy summary statistic, we write $s_{dp} \sim \eta(\cdot \mid x)$. Here, η is a known noise infusion process designed to meet a certain property: The privacy mechanism η is said to be ϵ-differentially private (Dwork and Roth 2013) if for all values of s_{dp}, and all "neighboring" databases $(x, x') \in \mathcal{X}^n \times \mathcal{X}^n$ differing by one record (denoted by $d(x, x') \leq 1$), the probability ratio is bounded:

$$\frac{\eta(s_{dp} \mid x)}{\eta(s_{dp} \mid x')} \leq \exp(\epsilon), \quad \epsilon > 0.$$

The parameter ϵ is called the privacy loss budget, and controls how strong the privacy guarantee is. Larger values of ϵ correspond to weaker privacy guarantees which in turn means less noise being added.

The differential privacy framework is used to create and verify privacy mechanisms. One such mechanism is the *Laplace mechanism*. It works by taking a deterministic statistic $s : X \mapsto \mathbb{R}^m$ and constructing the privatized statistic $s_{dp} := s(x) + u$ where u is a m-dimensional vector of i.i.d. Laplace noise. Using the ratio bound, if we draw each $u_i \sim Lap(\Delta(s)/\epsilon)$, we can show $s_d p$ is ϵ-differentially private. Here $\Delta(s) := \max_{(x,x') \in \mathcal{X}^n \times \mathcal{X}^n; d(x,x') \leq 1} \|s(x) - s(x')\|$ is the ℓ_1 (global) sensitivity of s. Roughly speaking global sensitivity can be thought of as quantifying how easy it is to identify a particular record. The idea being the easier it is to identify a record (high global sensitivity), the more noise that needs to be added to achieve a given privacy guarantee (Ju et al. 2022). Example 2, will cover an application of the Laplace mechanism to linear regression.

3 Methodology

Given data privatized data, s_{dp}, the goal of Bayesian inference is to sample from the posterior distribution $p(\theta \mid s_{dp})$. Since the observed likelihood, $p(s_{dp} \mid \theta)$, often has no simple closed form expression (Williams and Mcsherry 2010), most standard sampling schemes do not apply. To conduct privacy-aware Bayesian inference, the dapper package implements the data augmentation algorithm which allows us to sample from $p(\theta \mid s_{dp})$ without needing to specify $p(s_{dp} \mid \theta)$.

The algorithm considers the joint distribution $p(\theta, x \mid s_{dp})$ and alternates sampling from the two distributions

- $p(\theta \mid x, s_{dp})$
- $p(x \mid \theta, s_{dp})$

Since s_{dp} is derived from x, we have $p(\theta \mid x, s_{dp}) = p(\theta \mid x)$ which is just the usual posterior distribution given the confidential data x. The dapper package assumes the user has access to a sampler for $p(\theta \mid x)$. This can come from any R package such as fmcmc. For the second distribution, $p(x \mid \theta, s_{dp})$, may only be known up to a constant. The dapper package samples from this distribution by running a Gibbs-like sampler. Each of the n components of x is individually updated. However unlike the standard Gibbs sampler, each component is updated using a Metropolis-Hasting algorithm. This method is sometimes called the Metropolis within Gibbs sampler (Robert and Casella 2004).

In some cases, sampling from $p(x \mid \theta, s_{dp})$ can be made more efficient when the privacy mechanism can be written as a function of s_{dp} and a sum consisting of contribution from each individual record. More precisely, we say the privacy mechanism satisfies the *record additivity* property if

$$\eta(s_{dp} \mid x) = g(s_{dp}, \sum_{i=1}^{n} t_i(x_i, s_{dp}))$$

for some known and tractable functions $g, t_1, \ldots, t_n$. The sample mean is a example of a summary statistic satisfying record additivity where $t_i(x_i, s_{dp}) = x_i$.

The algorithm is in the following pseudo code:

1. Sample θ^{t+1} from $p(\cdot \mid x^{(t)})$.
2. Sample from $p(x \mid \theta, s_{dp})$ using a three step process
    - Propose $x_i^* \sim f(\cdot \mid \theta)$.
    - If s satisfies the record additive property then update $s(x^*, s_{dp}) = t(x, s_{dp}) - t_i(x_i, s_{dp}) + t_i(x_i^*, s_{dp})$.
    - Accept the proposed state with probability $\alpha(x_i^* \mid x_i, x_{-i}, \theta)$ given by:

$$\alpha(x_i^* \mid x_i, x_{-i}, \theta) = \min\left\{1, \frac{\eta(s_{dp} \mid x_i^*, x_{-i})}{\eta(s_{dp} \mid x_i, x_{-i})}\right\}.$$

4 The Structure of dapper

The package is structured around the two functions `dapper_sample` and `new_privacy`. The first function is used to draw samples from the posterior. The second function is used to create a privacy data model object which the `dapper_sample` function requires as input. The purpose of the data model object is to collect all the components specific to the data augmentation algorithm into one bundle. This way, the other arguments into `dapper_sample` pertain only to sampling parameters such as the number of iterations.

Since the input to these functions are R functions, there is a great deal of freedom in implementation. The next two sections describe in detail the inputs into these functions and highlight some considerations that should be taken into account in order to avoid slow or unexpected behavior.

Before delving into the specifics of each component, it is necessary to clearly define how the confidential data is represented. Internally, the confidential database is encoded as a 2D matrix. There can be multiple ways of doing this and is often the case when dealing with a likelihood function that has a sufficient statistic. For example, if our data consist of 100 responses from a two question, yes/no, survey. Then we can either encode the data as a 1×4 matrix of total counts, or a 100×2 matrix of binary responses. Both are mathematically equivalent, but the 1×4 matrix will be much more memory efficient. In general, the representation that uses the least amount of memory should be used. Correctly specifying the privacy model will require a consistent representation among all of its components.

Privacy Model

Creating a privacy model is done using the `new_privacy` constructor. The main arguments consist of the four components as outlined in the methodology section.

```
new_privacy(post_f = NULL, latent_f = NULL, priv_f = NULL,
            st_f = NULL, add = FALSE, npar = NULL)
```

The internal implementation of the data augmentation algorithm in `dapper_sample` requires some care in how each component is constructed.

- `latent_f` is an R function that samples from the latent process. The latent process being the probability model which dictates how to generate a new confidential record x, given θ. Its syntax should be `latent_f(theta)` where `theta` is a vector representing the model parameters being estimated. This function must work with the supplied initial parameter provide in the `init_par` argument of `dapper_sample` function. The output should be a $n \times k$ matrix where k is the dimension of θ and n is the number of observations. It is critical to remember the correct value for n is dependent on the choice of how the latent data is represented as discussed in the beginning of section 4.
- `post_f` is a function which makes draws from the posterior sampler. It should have the syntax `post_f(dmat, theta)`. Here `dmat` is the hypothetical data set representing the confidential data. This sampler can be generated by wrapping mcmc samplers generated from other R packages (e.g. **rstan**, **fmcmc**, **adaptMCMC**). If using this approach, it is recommended to avoid using packages with a large initialization overhead such as **mcmc** since the sampler is reinitialized every loop iteration. In the case of **mcmc**, the Metropolis-Hastings loop is implemented in C so there is a significant initialization cost when calling from an R function. The purpose of the `theta` argument is to serve as the initialization point if samples from `post_f` are draws from say a Metropolis-Hastings sampler.
- `priv_f` is an R function that represents the log of the privacy mechanism density, $\eta(s_{sdp} \mid x)$.
- `st_f` is an R function which calculates a summary statistic. The optional argument `add` is a flag which represents whether st_f should be interpreted as satisfying record additivity or not.

Sampling

The main function in **dapper** is the `dapper_sample` function. The syntax of the function is:

```
dapper_sample(data_model, sdp, nobs, init_par, niter = 2000, warmup = floor(niter / 2),
              chains = 1, varnames = NULL)
```

The three required inputs into `dapper_sample` function are the privacy model (`data_model`) whose construction is described in 4.1, the value of the observed privatized statistic (`sdp`), and the total number of observations in the complete data (`nobs`). The dapper package is best suited for problems where the complete data can be represented in tabular form. This is because internally, it is represented as a matrix.

The optional arguments are the number of mcmc draws (`niter`), the burn in period (`warmup`), number of chains (`chains`) and character vector that names the parameters. Running multiple chains can be done in parallel using the **furrr** package. Additionally, progress can be monitored using the **progressr** package. Adhering to the design philosophy of the two packages, we leave the setup to the user so that they may choose the most appropriate configuration for their system. The contingency table demonstration given in section 5 walks through a typical setup of **furrr** and **progressr**.

The `dapper_sample` function returns a `draw_matrix` object as described in the **posterior** package. One of the advantages with working with a `draw_matrix` object is that is compatible with many of the packages in the **rstan** ecosystem. For example, any `draw_matrix` object can be plugged directly into the popular **bayesplot** package. Additionaly, `dapper`'s basic summary function provides the same posterior summary statistics as those found when using **rstan**. Overall, this should make working with `dapper` easier for anyone already familiar with the **rstan** ecosystem.

5 Examples

2x2 Contingency Table

As a demonstration, we analyze the UC Berkeley admissions data, which is often used as an illustrative example of Simpson's paradox. The question posed is whether the data suggest there is bias against females during the college admissions process. Below is a table of the aggregate admissions result from six departments based on sex for a total of $N = 4526$ applicants. The table on the left represents

the true admissions data and the table on the right is the result of adding independent, $N(0, 100^2)$ error to each cell. Throughout the example we assume we only have access to the original total count, N, and the noise infused table.

	Male	Female
Admitted	1198	557
Rejected	1493	1278

	Male	Female
Admitted	1135.35	473.44
Rejected	1511.36	1437.53

As mentioned in section 4, it is critical to choose a consistent matrix representation for the confidential data. In this example we represent the confidential admissions data as a 1×4 matrix. Below we walk through the process of defining a privacy model.

1. `latent_f`: Since we can condition on the table total N, we can model the original, unobserved table counts as a multinomial distribution. We can easily draw from this distribution using the `rmultinom` function in the base stats package. Note, in this example, the return value of one sample from `rmultinom` is a 4×1 matrix, so in order to conform with our confidential data representation we take the transpose.

   ```
   latent_f <- function(theta) {
     t(rmultinom(1, 4526, theta))
   }
   ```

2. `post_f`: Given confidential data, we can derive the posterior analytically using a Dirichlet prior. In this example, we use a flat prior which corresponds to Dirch(1) distribution. A sample from the Dirichlet distribution can be generated using random draws from the gamma distribution.

   ```
   post_f <- function(dmat, theta) {
     x <- c(dmat)
     t1 <- rgamma(4, x + 1, 1)
     t1/sum(t1)
   }
   ```

3. `st_f`: Since the latent model only returns one observation from a multinomial distribution, we can just use the identity function as the summary statistic.

   ```
   st_f <- function(dmat) {
     c(dmat)
   }
   ```

4. `priv_f`: The privacy mechanism is Guassian white noise drawn from independent $N(0, 100^2)$ distributions. Hence given confidential table cells $(n_{11}, n_{22}, n_{12}, n_{21})$

   $$\eta(s_{dp} \mid x) = \prod \phi(s_{sd}; n_{ij}, 100^2).$$

   Here $\phi(\cdot; \mu, \sigma^2)$ is the density of the normal distribution with mean and variance μ, σ^2.

   ```
   priv_f <- function(sdp, x) {
     dnorm(sdp - x, mean = 0, sd = 100, log = TRUE)
   }
   ```

Once we have defined all components of the model we can create a new privacy model object using the `new_privacy` function and feed this into the `dapper_sample` function. Below we simulate 10,000 posterior draws with a burn-in of 1000.

```
library(dapper)
dmod <- new_privacy(post_f   = post_f,
                    latent_f = latent_f,
                    priv_f   = priv_f,
                    st_f     = st_f,
                    add      = FALSE,
                    npar     = 4,
                    varnames = c("pi_11", "pi_21", "pi_12", "pi_22"))

dp_out <- dapper_sample(dmod,
                  sdp = c(adm_prv),
```

```
                  niter = 10000,
                  warmup = 1000,
                  chains = 1,
                  init_par = rep(.25,4))
```

To run the `dapper_sample` function with parallel chains we can import the `furrr` package and use the `plan` function to determine the number of works used. The example code chunk below would produce four chains using two CPU's.

```
library(furrr)
plan(multisession, workers = 2)

dp_out <- dapper_sample(dmod,
                  sdp = c(adm_prv),
                  niter = 10000,
                  warmup = 1000,
                  chains = 4,
                  init_par = rep(.25,4))
```

If the run time of `dapper_sample` is exceptionally long, one can use the `progressr` package to monitor progress. The `progressor` framework allows for a unified handling of progress bars in both the sequential and parallel computing case.

```
library(progressr)

with_progress({
  dp_out <- dapper_sample(dmod,
                  sdp = c(adm_prv),
                  niter = 10000,
                  warmup = 1000,
                  chains = 4,
                  init_par = rep(.25,4))
})
```

results can be quickly summarized using the `summary` function which is displayed below. The `rhat` values in the table are close to 1, which indicates the chain has run long enough to achieve adequate mixing.

```
#> # A tibble: 4 x 10
#>   variable  mean median     sd    mad     q5   q95  rhat ess_bulk ess_tail
#>   <chr>    <num>  <num>  <num>  <num>  <num> <num> <num>    <num>    <num>
#> 1 pi_11    0.247  0.247 0.0248 0.0247 0.207  0.287  1.00    173.     434.
#> 2 pi_21    0.332  0.333 0.0248 0.0255 0.291  0.371  1.00    187.     448.
#> 3 pi_12    0.107  0.107 0.0253 0.0272 0.0659 0.147  1.01     62.8     98.7
#> 4 pi_22    0.314  0.313 0.0248 0.0252 0.274  0.355  1.00    213.     508.
```

Diagnostic checks using trace plots can be done using the **Bayesplot** package as shown in figure 1. It is especially important to check for good mixing with `dapper` since sticky chains are likely to be produced when the amount of injected noise is high. See the discussion section for a more detailed explanation.

To see if there is evidence of gender bias we can look at the odds ratio. Specifically, we look at the odds of a male being admitted to that of female. A higher odds ratio would indicate a bias favoring males. Figure 2 shows the posterior draws from the dapper model. The large odds ratio values would seem to indicate there is bias favoring the males. See [XXX] for an explanation of the "paradox" of this result.

For comparison, we run a standard Bayesian analysis on the noise infused table ignoring the privacy mechanism. This will correspond exactly to the model defined in the `post_f` component. Figure 3 shows a density estimate for the odds ratio under the confidential and noisy data. The posterior distribution for the odds ratio under the noisy data is shifted significantly, indicating a large degree of bias. Looking at left hand plot in 3 shows the MAP estimate from `dapper` is similar to that in the case of the confidential data. The width of the posterior is also much larger since it properly accounts for the uncertainty due to the privacy mechanism. This illustrates the dangers of ignoring the privacy mechanism. The naive analysis not only has bias, but also severely underestimate the uncertainty associated with the odds ratio estimate.

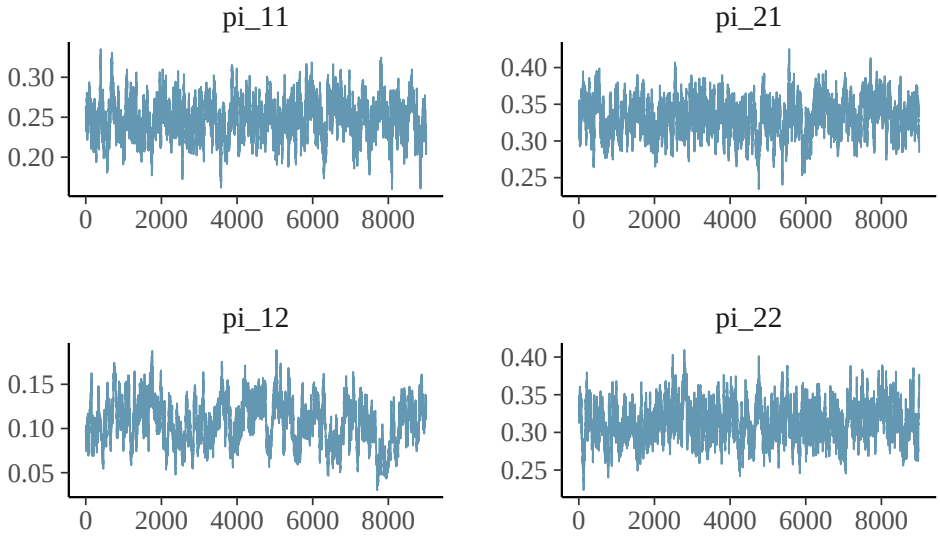


Figure 1: trace plots.

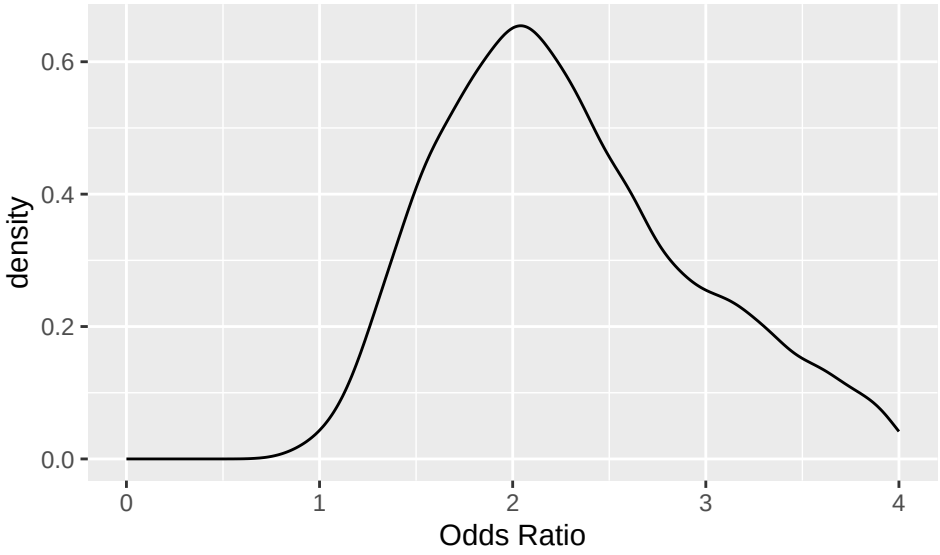


Figure 2: posterior density estimate for the odds ratio using 9000 MCMC draws.

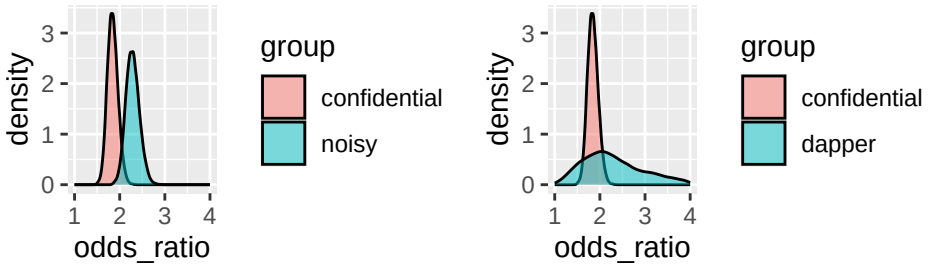


Figure 3: comparison.

Linear Regression

Below we review an application of `dapper` to an example presented in (Ju et al. 2022) where they apply a Laplace privacy mechanism to a sufficient summary statistic for a linear regression model. Let $\{(x_i, y_i)\}_{i=1}^n$ be the original, confidential data with $x_i \in \mathbb{R}^2$. They assume the true data generating process follows the model

$$\begin{aligned} y &= -1.79 - 2.89x_2 - 0.66x_2 + \epsilon \\ \epsilon &\sim N(0, 2^2) \\ X &\sim N_2(\mu, I_2) \\ \mu &= \begin{pmatrix} 0.9 \\ -1.17 \end{pmatrix} \end{aligned}$$

Note, in most settings involving linear regression, the covariates are assumed to be fixed, known constants. Thus the formulation above is a departure from the norm since we are assuming a random design matrix. More details on why this framing is necessary will be provided later when constructing the `dapper` likelihood component.

The paper considers the scenario where one desires to publicly release the sufficient summary statistics

$$t(x_i, y_i) = ((\tilde{x}^i)^T \tilde{y}_i, \tilde{y}_i^2, (\tilde{x}^i)^T \tilde{x}_i)$$

with ϵ-DP privacy guarantees. This was accomplished by clamping the data and adding Laplace random error. More precisely, we define the clamp function $[z] := \min\{\max\{z, -10\}, 10\}$ which truncates a value z so that it falls into the interval $[-10, 10]$. Furthermore, we let $\tilde{z} := [z]/10$ denote the normalized clamped value of z. Thus the clamped statistic is

$$t(x_i, y_i) = ((\tilde{x}^i)^T \tilde{y}_i, \tilde{y}_i^2, (\tilde{x}^i)^T \tilde{x}_i)$$

and the ϵ-DP private statistic is obtained by adding Laplace error to the unique elements of ...

data generating process:

$$\begin{aligned} X &\sim N_2(\mu, I_2) \\ \mu &= \begin{pmatrix} 0.9 \\ -1.17 \end{pmatrix} \end{aligned}$$

1. `latent_f`: Since the privacy mechanism involves injecting noise into the design matrix, it is not possible to use the standard approach where one assumes the design matrix is a fixed, known constant. Hence to draw a sample from the latent data generating process we use the relation $f(x, y) = f(x)f(y \mid x)$.

```
latent_f <- function(theta) {
  xmat <- MASS::mvrnorm(100 , mu = c(.9,-1.17), Sigma = diag(2))
  y <- cbind(1,xmat) %*% theta + rnorm(1, sd = sqrt(2))
  cbind(y,xmat)
}
```

2. `post_f`: Given confidential data X we can derive the posterior analytically using a normal prior on β.

$$\begin{aligned} \beta &\sim N_{p+1}(0, \tau^2 I_{p+1}) \\ \beta \mid x, y &\sim N(\mu_n, \Sigma_n) \\ \Sigma_n &= (x^T x/\sigma^2 + I_{p+1}/\tau^2)^{-1} \\ \mu_n &= \Sigma_n (x^T y)/\sigma^2 \end{aligned}$$

In the example, we use $\sigma^2 = 2$ and $\tau^2 = 4$.

```
post_f <- function(dmat, theta) {
  x <- cbind(1,dmat[,-1])
  y <- dmat[,1]

  ps_s2 <- solve((1/2) * t(x) %*% x + (1/4) * diag(3))
  ps_m <- ps_s2 %*% (t(x) %*% y) * (1/2)
```

```
  MASS::mvrnorm(1, mu = ps_m, Sigma = ps_s2)
}
```

3. st_f: Although the summary statistic contains $x^T x$, we do not need to include all entries since the matrix is symmetric.

```
clamp_data <- function(dmat) {
  pmin(pmax(dmat,-10),10) / 10
}

st_f <- function(dmat) {
  sdp_mat <- clamp_data(dmat)
  ydp <- sdp_mat[,1, drop = FALSE]
  xdp <- cbind(1,sdp_mat[,-1, drop = FALSE])

  s1 <- t(xdp) %*% ydp
  s2 <- t(ydp) %*% ydp
  s3 <- t(xdp) %*% xdp

  ur_s1 <- c(s1)
  ur_s2 <- c(s2)
  ur_s3 <- s3[upper.tri(s3,diag = TRUE)][-1]
  c(ur_s1,ur_s2,ur_s3)
}
```

4. priv_f: Privacy Mechanism Guassian white noise is added to each cell total. Hence given confidential data $(n_{11}, n_{22}, n_{12}, n_{21})$

$$\eta(s_{dp} \mid x) = \prod \phi(s_{sd}; n_{ij}, 100^2)$$

```
#deltaa <- 13
#epsilon <- 10
priv_f <- function(sdp, zt) {
  sum(VGAM::dlaplace(sdp - zt, 0, 13/10, log = TRUE))
}
```

RUN

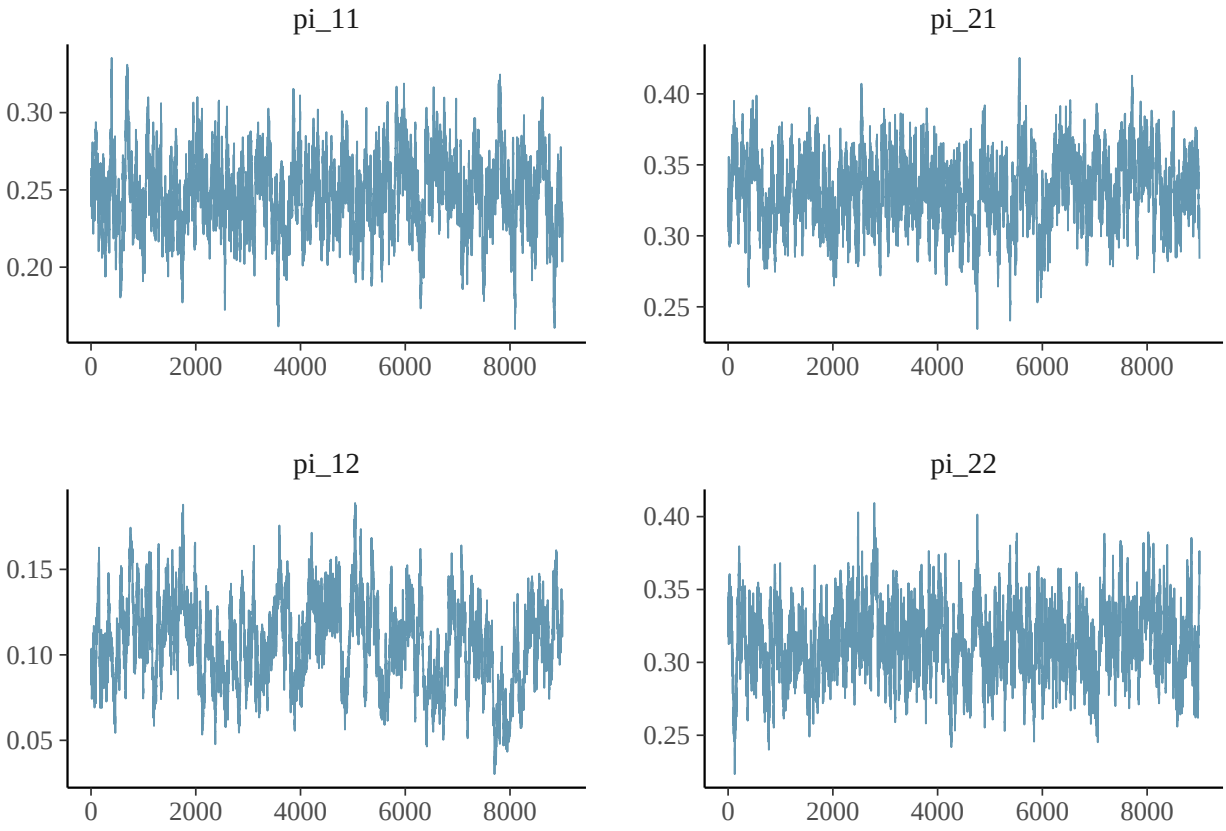
```
deltaa <- 13
epsilon <- 10
n <- 100
xmat <- MASS::mvrnorm(n, mu = c(.9,-1.17), Sigma = diag(2))
beta <- c(-1.79, -2.89, -0.66)
y <- cbind(1,xmat) %*% beta + rnorm(n, sd = sqrt(2))
z <- st_f(cbind(y,xmat))
z <- z + VGAM::rlaplace(length(z), location = 0, scale = deltaa/epsilon)

dmod <- new_privacy(post_f   = post_f,
                    latent_f = latent_f,
                    priv_f   = priv_f,
                    st_f     = st_f,
                    npar     = 3,
                    varnames = c("beta0", "beta1", "beta2"))

dp_out <- dapper_sample(dmod,
                        sdp = z,
                        niter = 3000,
                        warmup = 1000,
                        chains = 1,
                        init_par = rep(0,3))
```

```
#> # A tibble: 4 x 10
#>   variable  mean median     sd    mad     q5   q95  rhat ess_bulk ess_tail
#>   <chr>    <num>  <num>  <num>  <num>  <num> <num> <num>    <num>    <num>
#> 1 pi_11    0.247  0.247 0.0248 0.0247 0.207  0.287  1.00    173.     434.
#> 2 pi_21    0.332  0.333 0.0248 0.0255 0.291  0.371  1.00    187.     448.
```

```
#> 3 pi_12    0.107  0.107 0.0253 0.0272 0.0659 0.147  1.01     62.8     98.7
#> 4 pi_22    0.314  0.313 0.0248 0.0252 0.274  0.355  1.00    213.     508.
```



6 Discussion

Poor Mixing

Mixing can be poor when the posterior under a given privacy mechanism is much wider than the posterior that would arise using the confidential data. In other words, when the privacy budget is small, poor mixing can be expected. The rest of this section explores a toy example that will provide insight into this phenomenon.

Suppose the confidential data consist of a single observation $x \in \mathbb{R}$, and consider the scenario where a user makes a request to view x and in return receives $s := x + \eta$, which is a noise infused version of x. For simplicity, we do not worry about constructing an ϵ-DP privacy mechanism, and take $\eta \sim N(0, \epsilon^{-2})$ for some $\epsilon > 0$. However, it will still be useful to think of ϵ as the privacy budget since smaller values of ϵ correspond to a larger amounts of noise. Using a flat prior and a normally distributed likelihood results in a normally distributed posterior described below.

$$
\begin{aligned}
f(\theta) &\propto 1 \\
s \mid x &\sim N(x, \epsilon^{-2}) \\
x \mid \theta &\sim N(\theta, \sigma^2)
\end{aligned}
$$

With the above model, the data augmentation process consist of the two steps

- Step 1: Sample from $x \mid \theta, s \sim N(\mu, \tau^2)$.

$$\mu = \frac{\frac{1}{\epsilon^{-2}}s + \frac{1}{\sigma^2}\theta}{\frac{1}{\epsilon^{-2}} + \frac{1}{\sigma^2}}$$
$$\tau^2 = \frac{1}{\frac{1}{\epsilon^{-2}} + \frac{1}{\sigma^2}}$$

- Step 2: Sample from $\theta \mid x, s \sim N(x, \sigma^2)$.

It turns out the lag-1 auto-correlation is closely related to the convergence rate of the chain in this example. [Jun] showed the lag-1 autocorrelation is related to the Bayesian fraction of missing information and the latter, in fact, gives the exact convergence rate. The Bayesian fraction of missing information, γ is defined as

$$\gamma := 1 - \frac{E[Var(\theta \mid s, x) \mid s]}{Var(\theta \mid s)} = 1 - \frac{E[Var(\theta \mid x)]}{Var(\theta \mid s)}.$$

plugging in the appropriate quantities gives us

$$\gamma = 1 - \frac{\sigma^2}{\sigma^2 + \epsilon^{-2}} = 1 - \frac{1}{1 + \frac{\epsilon^{-2}}{\sigma^2}}.$$

The chain converges faster as $\gamma \to 0$ an slower as $\gamma \to 1$. From the right hand term in the above panel, we can see γ depends only on ϵ^{-2}/σ^2 and as the privacy budget decreases (i.e. more noise is being added to x), $\gamma \to 1$.

Thus when poor mixing is observed, we recommend seeing if increasing the privacy budget helps. Unfortunately, if one is not at liberty to alter the privacy budget, there is no quick fix. In the case where a small privacy budget results in a posterior much more diffuse than under the confidential data set, one can draw samples using the pseudo-likelihood scheme as proposed in []. This scheme fits in the same data augmentation framework as `dapper` but is not implemented.

Related Work

Adjusting statistical workflows to account for added noise in covariates is extensively studied in the context of measurement error (e.g. noisy sensor readings) and there exists readily available tools for making the proper adjustments. For textbook length treatments on the topic see (Yi 2017; Carroll et al. 2006). Work in this area mostly focuses on methods which do not require fully specifying the measurement error model, since this is often assumed unknown. However, in differential privacy, the measurement error model is exactly known. This difference, makes feasible some ideas which the measurement error community has not previously considered (Smith 2011; Karwa, Kifer, and Slavković 2015b).

7 Summary

Currently, there is a dearth of software tools privacy researchers can use to evaluate the impact of privacy mechanisms on statistical analyses. While there have been tremendous gains in the theoretical aspects of privacy, the lack of software resources to deploy and work with new privacy techniques has hampered their adoption. This gap in capability has been noted by several large industry entities who have begun building software ecosystems for working with differential privacy. SmartNoise by Microsoft [insert citation], for example, is a tool for generating synthetic data that has differentially private guarantees. However, the majority of these software tools only address privacy and not the ensuing analysis, or if it does address the analysis, only for specific models. Privacy researchers currently lack good tools for evaluating the impact of privacy mechanisms on a statistical analysis.

Thus DAPPER helps fill an urgent need by providing researchers a way to evaluate how a particular privacy mechanism might effect a statistical analysis. A notable feature is its flexibility which allows the users to specify a custom privacy mechanism. The benefit being that DAPPER can evaluate already established privacy mechanisms and those that have yet to be discovered.

While DAPPER can be a nice addition to a privacy researcher's tool kit, there is still considerably more work that can be done. From a methodological standpoint, DAPPER suffers from slow convergence when the privacy budget is small which is problematic because this is the scenario where a privacy mechanism can have the greatest impact on a statistical analysis. On the computational end, without a statistic that satisfies record additivity, computation time can be unpalatable.

References

Abowd, John M. 2018. "The u.s. Census Bureau Adopts Differential Privacy." In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &Amp; Data Mining*. KDD '18. ACM. https://doi.org/10.1145/3219819.3226070.

Bureau, U. S. Census. 2023. "Disclosure Avoidance and the 2020 Census: How the TopDown Algorithm Works." 2023. https://www.census.gov/library/publications/2023/decennial/c2020br-04.html.

Carroll, Raymond J., David Ruppert, Leonard A. Stefanski, and Ciprian M. Crainiceanu. 2006. *Measurement Error in Nonlinear Models*. Chapman; Hall/CRC. https://doi.org/10.1201/9781420010138.

Ding, Bolin, Janardhan Kulkarni, and Sergey Yekhanin. 2017. "Collecting Telemetry Data Privately." https://arxiv.org/abs/1712.01524.

Dwork, Cynthia, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. "Calibrating Noise to Sensitivity in Private Data Analysis." In *Lecture Notes in Computer Science*, 265–84. Springer Berlin Heidelberg. https://doi.org/10.1007/11681878_14.

Dwork, Cynthia, and Aaron Roth. 2013. "The Algorithmic Foundations of Differential Privacy." *Foundations and Trends® in Theoretical Computer Science* 9 (3-4): 211–407. https://doi.org/10.1561/0400000042.

Erlingsson, Úlfar, Vasyl Pihur, and Aleksandra Korolova. 2014. "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response." In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. CCS'14. ACM. https://doi.org/10.1145/2660267.2660348.

Gong, Ruobin. 2022. "Transparent Privacy Is Principled Privacy." *Harvard Data Science Review*, no. Special Issue 2 (June). https://doi.org/10.1162/99608f92.b5d3faaa.

Ji, Zhanglong, Zachary C. Lipton, and Charles Elkan. 2014. "Differential Privacy and Machine Learning: A Survey and Review." https://arxiv.org/abs/1412.7584.

Ju, Nianqiao, Jordan Awan, Ruobin Gong, and Vinayak Rao. 2022. "Data Augmentation MCMC for Bayesian Inference from Privatized Data." In *Advances in Neural Information Processing Systems*, edited by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. https://openreview.net/forum?id=tTWCQrgjuM.

Karwa, Vishesh, Dan Kifer, and Aleksandra B. Slavković. 2015b. "Private Posterior Distributions from Variational Approximations." https://arxiv.org/abs/1511.07896.

———. 2015a. "Private Posterior Distributions from Variational Approximations." https://arxiv.org/abs/1511.07896.

Kenny, Christopher T., Shiro Kuriwaki, Cory McCartan, Evan T. R. Rosenman, Tyler Simko, and Kosuke Imai. 2021. "The Use of Differential Privacy for Census Data and Its Impact on Redistricting: The Case of the 2020 u.s. Census." *Science Advances* 7 (41). https://doi.org/10.1126/sciadv.abk3283.

Robert, Christian P., and George Casella. 2004. *Monte Carlo Statistical Methods*. *Springer Texts in Statistics*. Springer New York. https://doi.org/10.1007/978-1-4757-4145-2.

Santos-Lozada, Alexis R., Jeffrey T. Howard, and Ashton M. Verdery. 2020. "How Differential Privacy Will Affect Our Understanding of Health Disparities in the United States." *Proceedings of the National Academy of Sciences* 117 (24): 13405–12. https://doi.org/10.1073/pnas.2003714117.

Smith, Adam. 2011. "Privacy-Preserving Statistical Estimation with Optimal Convergence Rates." In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*. STOC'11. ACM. https://doi.org/10.1145/1993636.1993743.

Tang, Jun, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. 2017. "Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12." https://arxiv.org/abs/1709.02753.

Wang, Yue, Daniel Kifer, Jaewoo Lee, and Vishesh Karwa. 2018. "Statistical Approximating Distributions Under Differential Privacy." *Journal of Privacy and Confidentiality* 8 (1). https://doi.org/10.29012/jpc.666.

Williams, Oliver, and Frank Mcsherry. 2010. "Probabilistic Inference and Differential Privacy." In *Advances in Neural Information Processing Systems*, edited by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2010/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf.

Winkler, Richelle L., Jaclyn L. Butler, Katherine J. Curtis, and David Egan-Robertson. 2021. "Differential Privacy and the Accuracy of County-Level Net Migration Estimates." *Population Research and Policy Review* 41 (2): 417–35. https://doi.org/10.1007/s11113-021-09664-5.

Yi, Grace Y. 2017. *Statistical Analysis with Measurement Error or Misclassification*. *Springer Series in Statistics*. Springer New York. https://doi.org/10.1007/978-1-4939-6640-0.

Kevin Eng
Rutgers University
Department of Statistics
Piscataway, NJ 08854
https://www.britannica.com/animal/quokka
ke157@stat.rutgers.edu

Jordan A. Awan
Purdue University
Department of Statistics
West Lafayette, IN 47907
https://www.britannica.com/animal/quokka
jawan@purdue.edu

Ruobin Gong
Rutgers University
Department of Statistics
Piscataway, NJ 08854
https://www.britannica.com/animal/quokka
ruobin.gong@rutgers.edu

Nianqiao Phyllis Ju
Purdue University
Department of Statistics
West Lafayette, IN 47907
https://www.britannica.com/animal/quokka
nianqiao@purdue.edu

Vinayak A. Rao
Purdue University
Department of Statistics
West Lafayette, IN 47907
https://www.britannica.com/animal/quokka
varao@purdue.edu