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. The goal of this package is to provide some tools 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. 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 (), Google (), Microsoft (), and the U.S. Census Bureau ().

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 (Gong 2022). Therefore, fitting models without accounting for the added privacy noise can lead to incorrect inference.

Unfortunately, making the necessary adjustments poses formidable mathematical challenges, 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.

In the previously mentioned case where privacy noise is directly added to each cell of a data table, there exist readily available methods from measurement error literature for making 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 2015).

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.

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 $d(x, x')$), 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.

Data Augmentation

The idea behind data augmentation is to run a Gibbs sampler on a coupling of two random variables where one of the marginals is the target distribution. Suppose we wish to sample from a density f_A which is difficult. The data augmentation method instead considers sampling from a joint distribution $f(a, b)$. Since we are ultimately interested in samples from f_A, the joint distribution should be chosen so that (i) the marginal distribution with respects to a is f_A and (ii) $f(a \mid b)$ and $f(b \mid a)$ are easy to sample from. The choice f is not unique and can require some foresight.

3 Methodology

Given 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, 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 left up to user. The next two sections describe in detail the inputs into these functions and highlight some considerations that should be taken into account in order 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 are often multiple ways of doing this. 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 2×2 matrix, or a 100×2 matrix. Both are mathematically equivalent, but the 2×2 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 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 DA algorithm in `dapper_sample` requires some care in how each component is constructed.

- `latent_f` is an R function that samples from the likelihood. Its syntax should be `lik_smpl(theta)` where `theta` is a vector representing the likelihood 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 the vector of likelihood parameters.
- `post_f` is a function which represents the posterior sampler. It should have the call signature `post_smpl(dmat, theta)`. Where `dmat` is the complete 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 such as **mcmc** whose implementation clashes with `gdp_sample`. In the case of **mcmc**, the Metropolis-Hastings loop is implemented in C which incurs a very large overhead in `gdp_sample` since it is reinitialized every iteration. In general, repeatedly calling an R function that hooks into C code is slow. (NOT QUITE ACCURATE FIX LATER)
- `priv_f` is an R function that represents the log-likelihood of $\eta(s_{sdp} \mid x)$. The function can output the log likelihood up to an additive constant.
- `st_f` is an R function which calculates the summary statistic. The optional argument `add` is a flag which represents whether T is additive 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`), 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.

The `data_model` input is a `privacy` object that can be constructed using the `new_privacy` constructor. The process of constructing a `privacy` object will be discussed in the next section.

5 Examples

2x2 Contingency Table

A common procedure when analyzing contingency tables is to estimate the odds ratio. Something something about safetab to connect back to DP (dont forget citation!). 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	473
Rejected	1511	1438

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. In order to conform with `dapper_sample` we must convert the matrix to a vector.

```
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(length(theta), 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.

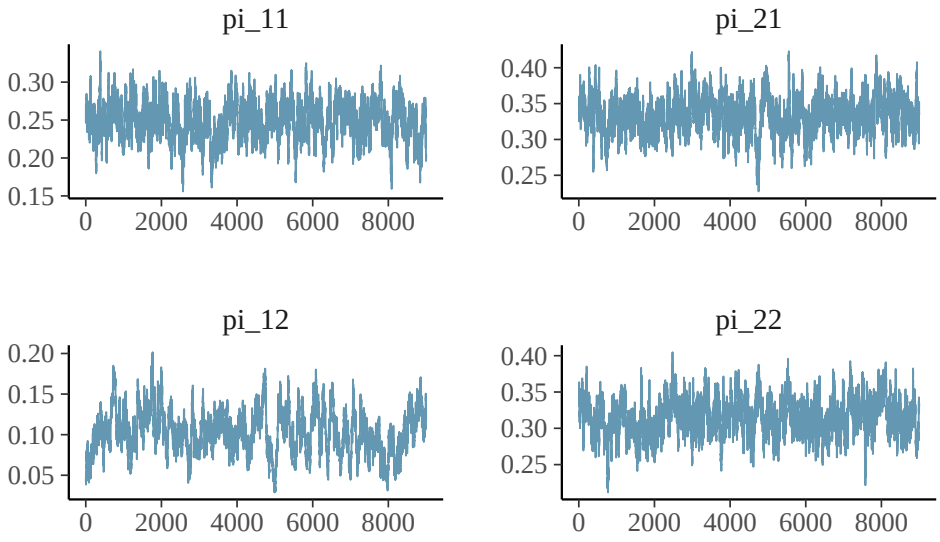
```
library(dapper)
dmod <- new_privacy(post_f   = post_f,
                    latent_f = latent_f,
                    priv_f   = priv_f,
                    st_f     = st_f,
                    add      = FALSE,
                    npar     = 4,
                    varnames = c("pi_11", "pi_21", "pi_12",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

dp_out <- dapper_sample(dmod,
                  sdp = c(adm_prv),
                  niter = 10000,
                  warmup = 1000,
                  chains = 1,
                  init_par = rep(.25,4))
```

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.

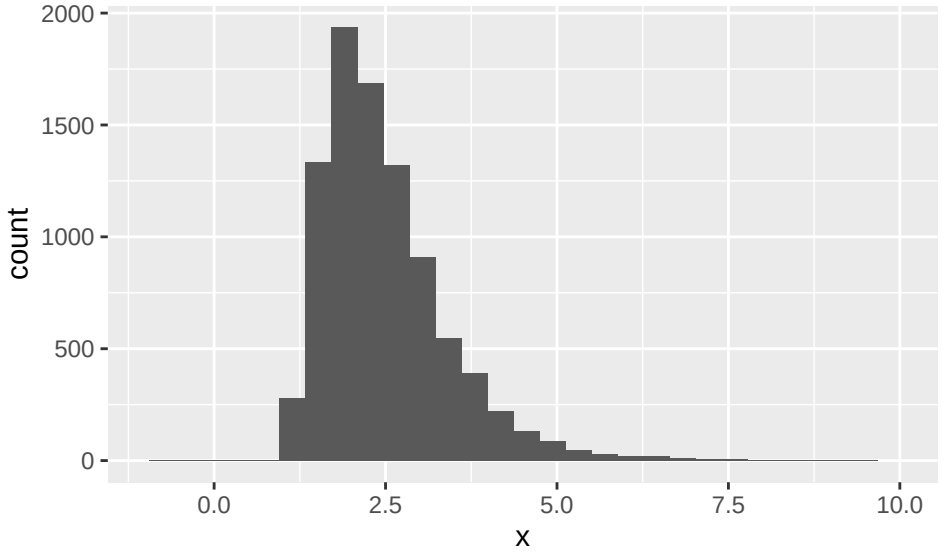
```
#> # A tibble: 4 x 10
#>   variable  mean median     sd    mad     q5   q95  rhat ess_bulk ess_tail
#>   <chr>    <num>  <num>  <num>  <num>  <num> <num> <num>    <num>    <num>
#> 1 pi_11    0.248  0.249 0.0254 0.0257 0.206  0.288  1.00    191.      388.
#> 2 pi_21    0.334  0.333 0.0261 0.0254 0.291  0.377  1.00    210.      329.
#> 3 pi_12    0.103  0.103 0.0272 0.0271 0.0586 0.149  1.02     76.0     159.
#> 4 pi_22    0.315  0.315 0.0256 0.0261 0.275  0.358  1.00    164.      460.
```

Diagnostic checks can be done using the **Bayesplot** package.



log odds distribution

```
#>     2.5%      50%    97.5%
#> 1.297400 2.289552 4.785425
```



For comparison, we run a standard statistical analysis on the noise infused table ignoring the privacy mechanism. The results from dapper will allow us to evaluate the impact of privacy on the analysis. A popular estimator for the log odds ratio in a 2×2 table is

$$\hat{\theta} := \log\left(\frac{n_{11} \cdot n_{22}}{n_{12} \cdot n_{21}}\right)$$

which corresponds to the maximum likelihood estimator under the assumption the rows or columns (but not both) are drawn from two independent binomial distributions. This is a reasonable assumption for the admissions data since sex can be viewed as two independent populations. The approximate variance of this estimator is

$$Var(\log(\hat{\theta})) \approx \frac{1}{n_{11}} + \frac{1}{n_{22}} + \frac{1}{n_{12}} + \frac{1}{n_{21}}$$

and the associated Wald based $(1-\alpha)$-level confidence interval is

$$\log\left(\frac{n_{11} \cdot n_{22}}{n_{12} \cdot n_{21}}\right) \pm z_{\alpha/2}\sqrt{\frac{1}{n_{11}} + \frac{1}{n_{22}} + \frac{1}{n_{12}} + \frac{1}{n_{21}}}.$$

```
#> [1] 1.848471 1.833718

#> [1] 2.293115 2.274220

or_confint <- function(x, alpha) {
  or <- log(x[1] * x[4]/ (x[2] * x[3]))
  se <- sqrt(sum(1/x))
  list(est = or, ci = qnorm(alpha/2) * se)
}
tibble(estimate = or_confint(x,.05))

#> # A tibble: 2 x 1
#>   estimate
#>   <named list>
#> 1 <dbl [1]>
#> 2 <dbl [1]>
```

Plugging in the numbers,

Linear Regression

(Ju et al. 2022) consider applying a Laplace privacy mechanism to a sufficient summary statistic for a linear regression model.

data generating process:

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

Suppose data are generated by the process

$$y = \beta_0 + \beta_1 x_2 + \beta_2 x_2 + \epsilon/$$

ϵ-DP is achieved by clamping the data. 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.

Use the statistic

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

1. `lik_f`: Conditional on the table total, the table counts follow 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. In order to conform with `dapper_sample` we must convert the matrix to a vector.

   ```
   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 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 the gamma distribution via the following relation (INSERT)

   ```
   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: The complete data can be represented in two ways. Micro vs cell totals. (what section to introduce?) This function must return a vector.

```
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))
}
```

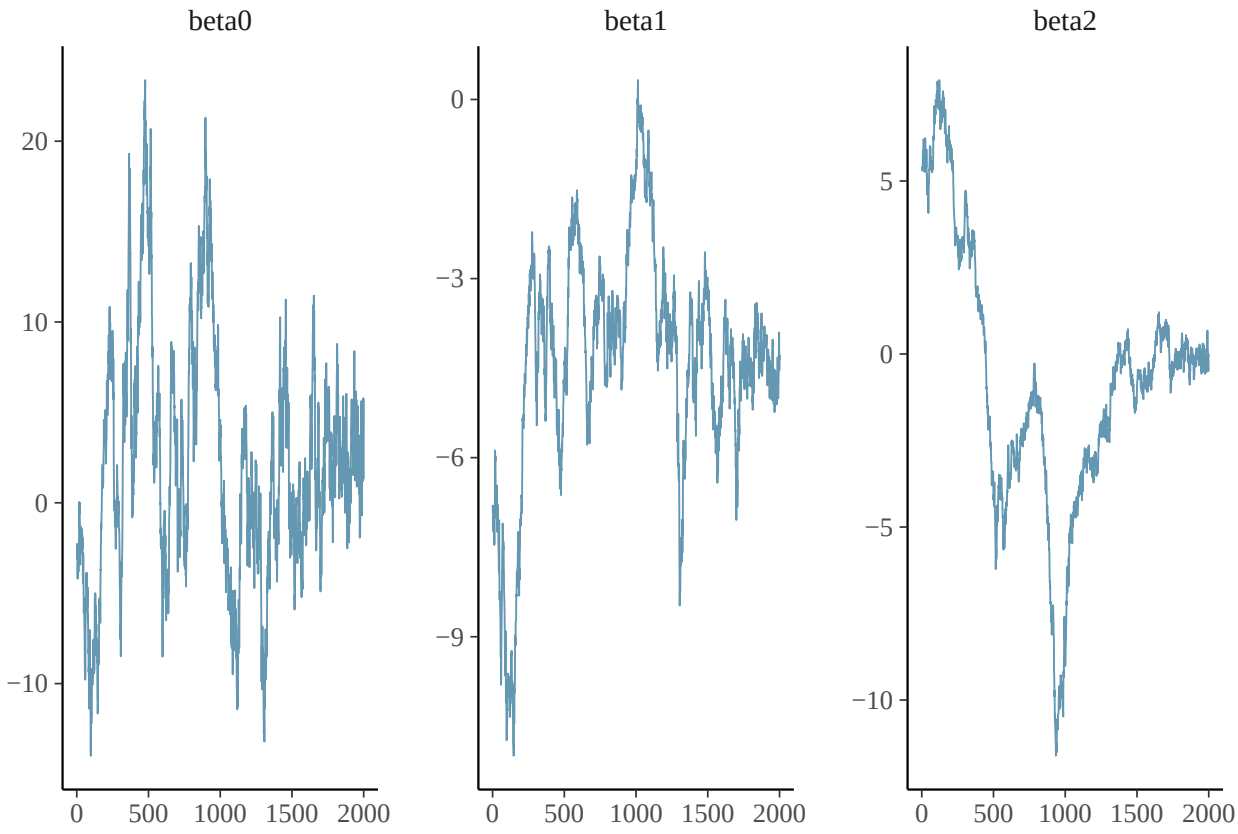
RUN

```
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: 3 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 beta0     2.23   1.69   6.62  5.95 -8.25 15.0   1.16    14.3      42.5
#> 2 beta1    -4.34  -4.17   1.94  1.22 -8.32 -1.41  1.06    16.9      13.2
#> 3 beta2    -0.813 -0.690  3.76  2.86 -7.58  6.15  1.42     7.33     12.4
```



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

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.

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.

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

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.

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.

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