gdpR: An R Package for studying differentially private algorithms

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

Abstract This paper serves as a reference and introduction on using the *gdpR* 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

(cliche...) The ease and pervasiveness of modern data collection technologies has raised concerns about data privacy. (Dwork and Roth 2013) introduced the differential privacy framework as a means to rigorously define privacy.

2 Using gdpR

The package is structured around the two functions `gdp_sample` and `new_privacy`. The first function is used to draw samples from the posterior. The second function is used to create the privacy model. 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.

Sampling

The main function in **gdpR** is the `gdp_sample` function. The call signature of the function is:

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

The three required inputs into `gdp_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`) [MAKE SURE NOTATION IS INTRODUCED]. The **gdpR** 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.

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_smpl = NULL, lik_smpl = NULL, ll_priv_mech = NULL,
            st_calc = NULL, add = FALSE, npar = NULL)
```

The internal implementation of the DA algorithm in `gdp_sample` requires some care in how each component in constructed.

- `lik_smpl` is an R function that samples from the likelihood. Its call signature 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 `gdp_sample`. The sampler need not be vectorized and vectorizing the sampler will not add any speed benefits.

- `post_smpl` 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)
- `ll_priv_mech` 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_calc` is an R function which calculates the summary statistic. The optional argument `add` is a flag which represents whether T is additive or not.

3 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 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.

	Male	Female		Male	Female
Admitted	1198	557	Admitted	1135	473
Rejected	1493	1278	Rejected	1511	1438

Below we walk through the process of defining a privacy model.

1. `lik_smpl`: 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 `gdp_sample` we must convert the matrix to a vector.

```
lik_smpl <- function(theta) {
  c(rmultinom(1, 4526, theta))
}
```

2. `post_smpl`: 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_smpl <- function(dmat, theta) {
  x <- c(dmat)
  t1 <- rgamma(length(theta), x + 1, 1)
  t1/sum(t1)
}
```

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

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

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

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

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

Once privacy model has been defined we can run `gdp_sample`

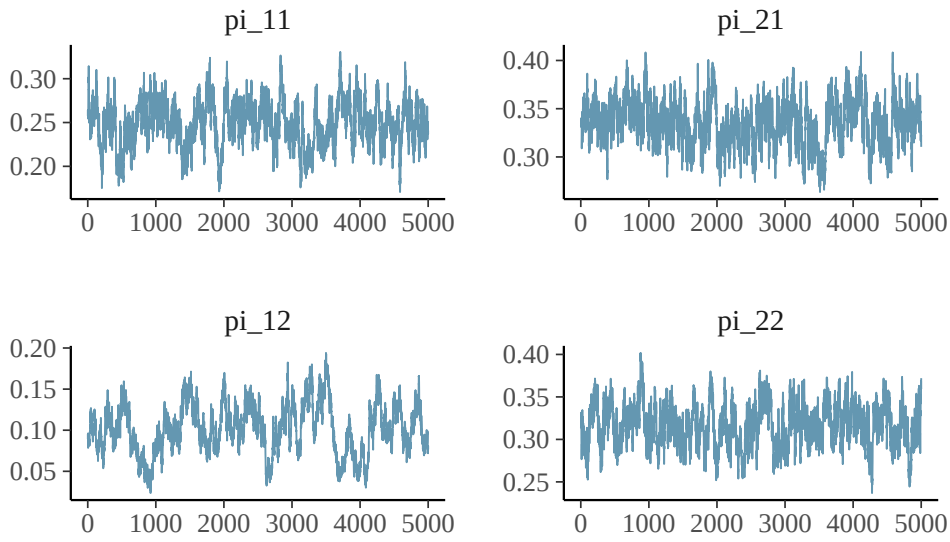
```
library(DPloglin)
dmod <- new_privacy(post_smpl = post_smpl,
                    lik_smpl = lik_smpl,
                    ll_priv_mech = ll_priv_mech,
                    st_calc = st_calc,
                    add = FALSE,
                    npar = 4)

gdp_out <- gdp_sample(dmod,
                  sdp = c(adm_prv),
                  nobs = 1,
                  niter = 6000,
                  warmup = 1000,
                  chains = 1,
                  init_par = rep(.25,4),
                  varnames = c("pi_11", "pi_21", "pi_12", "pi_22"))
```

results can be quickly summarized using the `summary` function

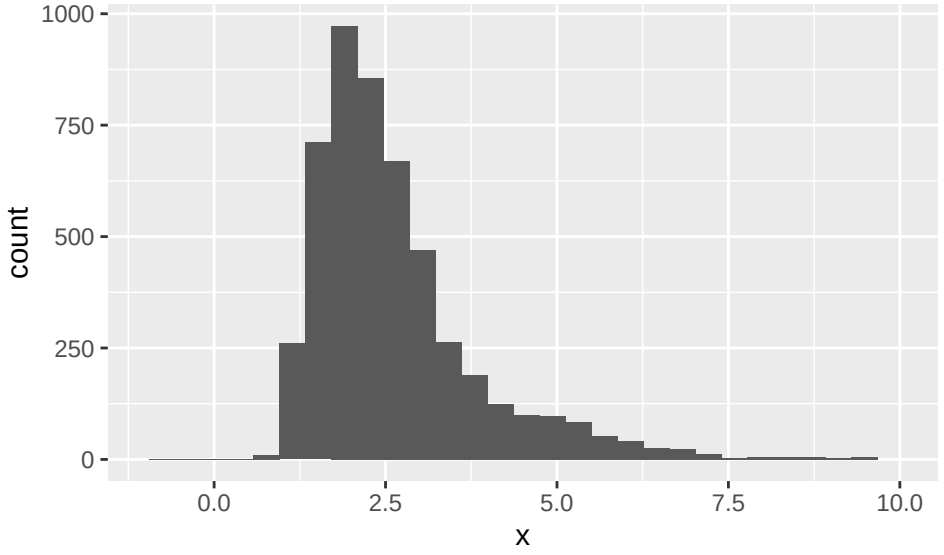
```
#> # A tibble: 4 x 10
#>   variable  mean median     sd    mad     q5   q95  rhat ess_bulk ess_tail
#>   <chr>    <num>  <num>  <num>  <num>  <num> <num> <num>    <num>    <num>
#> 1 pi_11    0.248  0.249 0.0258 0.0256 0.203  0.288  1.00     96.1      183.
#> 2 pi_21    0.335  0.335 0.0238 0.0239 0.293  0.373  1.01    107.       157.
#> 3 pi_12    0.102  0.101 0.0305 0.0321 0.0486 0.152  1.02     34.8      100.
#> 4 pi_22    0.316  0.316 0.0247 0.0256 0.275  0.356  1.00    119.       241.
```

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



log odds distribution

```
#>     2.5%      50%    97.5%
#> 1.215316 2.321789 6.062528
```



(Insert gdp Analysis Here)

For clean data, a estimate for the odds ratio and a confidence interval can be constructed using Woolf's method (i.e. A wald confidence interval). It uses the fact that the log of the odds ratio is approximately normal for large sample sizes.

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

```
or_confint <- function(x, alpha) {
  or <- log(x[1] * x[4]/ (x[2] * x[3]))
  se <- sqrt(sum(1/x))
  c(or - qnorm(alpha/2) * se, or + qnorm(alpha/2) * se)
}

#clean data
exp(or_confint(x, .95))

#> [1] 1.848471 1.833718

#privitized data
exp(or_confint(sdp, .95))

#> [1] 2.293115 2.274220
```

4 Summary

This package is cool. You should install it.

References

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.

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

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

Robin 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