Package: dapper (via r-universe)

September 8, 2024

Title Data Augmentation for Private Posterior Estimation

Version 1.0.0

Description A data augmentation based sampler for conducting privacy-aware Bayesian inference. The dapper_sample() function takes an existing sampler as input and automatically constructs a privacy-aware sampler. The process of constructing a sampler is simplified through the specification of four independent modules, allowing for easy comparison between different privacy mechanisms by only swapping out the relevant modules. Probability mass functions for the discrete Gaussian and discrete Laplacian are provided to facilitate analyses dealing with privatized count data. The output of dapper_sample() can be analyzed using many of the same tools from the `rstan` ecosystem. For methodological details on the sampler see Ju et al. (2022) <doi:10.48550/arXiv.2206.00710>, and for details on the discrete Gaussian and discrete Laplacian distributions see Canonne et al. (2020) <doi:10.48550/arXiv.2004.00010>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

URL https://github.com/mango-empire/dapper

BugReports https://github.com/mango-empire/dapper/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports bayesplot, checkmate, furrr, memoise, posterior, progressr, stats

Repository https://mango-empire.r-universe.dev

RemoteUrl https://github.com/mango-empire/dapper

RemoteRef HEAD

RemoteSha bdebe71456dd8a358ee0f0ba35e0a83dc6ab0568

Contents

`dapper_sample` *Private Posterior Sampler*

Description

Generates samples from the private posterior using a data augmentation framework.

Usage

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

Arguments

`data_model`	a data model represented by a `privacy` class object.
`sdp`	the observed privatized data. Must be a vector or matrix.
`init_par`	initial starting point of the chain.
`seed`	set random seed.
`niter`	number of draws.
`warmup`	number of iterations to discard as warmup. Default is half of niter.
`chains`	number of MCMC chains to run. Can be done in parallel or sequentially.

Details

Generates samples from the private posterior implied by `data_model`. The `data_model` input must by an object of class `privacy` which is created using the new_privacy() constructor. MCMC chains can be run in parallel using furrr::future_map(). See the **furrr** package documentation for specifics. Long computations can be monitored with the **progressr** package.

Value

A dpout object which contains: *chain: a draw_matrix object containing niter - warmpup draws from the private posterior. *accept_prob: a (niter - warmup) row matrix containing acceptance probabilities. Each column corresponds to a parameter.

References

Ju, N., Awan, J. A., Gong, R., & Rao, V. A. (2022). Data Augmentation MCMC for Bayesian Inference from Privatized Data. *arXiv*. doi:10.48550/ARXIV.2206.00710

See Also

new_privacy()

Examples

```
#simulate confidential data
#privacy mechanism adds gaussian noise to each observation.
set.seed(1)
n <- 100
eps <- 3
y <- rnorm(n, mean = -2, sd = 1)
sdp <- mean(y) + rnorm(1, 0, 1/eps)

post_f <- function(dmat, theta) {
    x <- c(dmat)
    xbar <- mean(x)
    n <- length(x)
    pr_m <- 0
    pr_s2 <- 4
    ps_s2 <- 1/(1/pr_s2 + n)
    ps_m <- ps_s2 * ((1/pr_s2)*pr_m + n * xbar)
    rnorm(1, mean = ps_m, sd = sqrt(ps_s2))
}
latent_f <- function(theta) {
    matrix(rnorm(100, mean = theta, sd = 1), ncol = 1)
}
st_f <- function(xi, sdp, i) {
    mean(xi)
}
priv_f <- function(sdp, sx) {
  sum(dnorm(sdp - sx, 0, 1/eps, TRUE))
}
dmod <- new_privacy(post_f = post_f,
  latent_f = latent_f,
  priv_f = priv_f,
  st_f = st_f,
  npar = 1)

out <- dapper_sample(dmod,
                     sdp = sdp,
                     init_par = -2,
```

```
                    niter = 500)
summary(out)

# for parallel computing we 'plan' a session
# the code below uses 2 CPU cores for parallel computing
library(furrr)
plan(multisession, workers = 2)
out <- dapper_sample(dmod,
                    sdp = sdp,
                    init_par = -2,
                    niter = 500,
                    chains = 2)

# to go back to sequential computing we use
plan(sequential)
```

---

ddlaplace *Discrete Laplace Distribution*

Description

The probability mass function and random number generator for the discrete Laplacian distribution.

Usage

```
ddlaplace(x, scale = 1, log = FALSE)

rdlaplace(n, scale = 1)
```

Arguments

x	a vector of quantiles.
scale	the scale parameter.
log	logical; if TRUE, probabilities are given as log(p).
n	number of random deviates.

Details

Probability mass function

$$P[X = x] = \frac{e^{1/t} - 1}{e^{1/t} + 1} e^{-|x|/t}.$$

Value

- ddlaplace() returns a numeric vector representing the probability mass function of the discrete Laplace distribution.
- rdlaplace() returns a numeric vector of random samples from the discrete Laplace distribution.

References

Canonne, C. L., Kamath, G., & Steinke, T. (2020). The Discrete Gaussian for Differential Privacy. *arXiv*. doi:10.48550/ARXIV.2004.00010

Examples

```
# mass function
ddlaplace(0)

# mass function is vectorized
ddlaplace(0:10, scale = 5)

# generate random samples
rdlaplace(10)
```

`ddnorm` *The Discrete Gaussian Distribution*

Description

The probability mass function and random number generator for the discrete Gaussian distribution with mean `mu` and scale parameter `sigma`.

Usage

```
ddnorm(x, mu = 0, sigma = 1, log = FALSE)

rdnorm(n, mu = 0, sigma = 1)
```

Arguments

`x`	vector of quantiles.
`mu`	location parameter.
`sigma`	scale parameter.
`log`	logical; if `TRUE`, probabilities are given as log(p).
`n`	number of random deviates.

Details

Probability mass function

$$P[X = x] = \frac{e^{-(x-\mu)^2/2\sigma^2}}{\sum_{y \in \mathbb{Z}} e^{-(x-\mu)^2/2\sigma^2}}.$$

Value

- ddnorm() returns a numeric vector representing the probability mass function of the discrete Gaussian distribution.
- rdnorm() returns a numeric vector of random samples from the discrete Gaussian distribution.

References

Canonne, C. L., Kamath, G., & Steinke, T. (2020). The Discrete Gaussian for Differential Privacy. *arXiv*. doi:10.48550/ARXIV.2004.00010

Examples

```
# mass function
ddnorm(0)

# mass function is also vectorized
ddnorm(0:10, mu = 0, sigma = 5)

# generate random samples
rdnorm(10)
```

`new_privacy` `privacy` *Object Constructor.*

Description

Creates a `privacy` object to be used as input into dapper_sample().

Usage

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

Arguments

`post_f`	a function that draws posterior samples given the confidential data.
`latent_f`	a function that represents the latent data sampling model.
`priv_f`	a function that represents the log likelihood of the privacy mechanism.
`st_f`	a function that calculates the statistic to be released.

npar	dimension of the parameter being estimated.
varnames	an optional character vector of parameter names. Used to label summary outputs.

Details

- post_f() is a function which makes draws from the posterior sampler. It has the syntax post_f(dmat, theta). Here dmat is a numeric matrix representing the confidential database and theta is a numeric vector which serves as the initialization point for a one sample draw. The easiest, bug-free way to construct post_f() is to use a conjugate prior. However, this function can also be constructed by wrapping a MCMC sampler generated from other R packages (e.g. **rstan**, **fmcmc**, **adaptMCMC**).
- priv_f() is a function that represents the log of the privacy mechanism density. This function has the form priv_f(sdp, sx) where sdp and sx are both either a numeric vector or matrix. The arguments must appear in the exact stated order with the same variables names as mentioned. Finally, the return value of priv_f() must be a numeric vector of length one.
- st_f() is a function which calculates a summary statistic. It has the syntax st_f(i, xi, sdp) where the three arguments must appear in the stated order. The role of this function is to represent terms in the definition of record additivity. Here i is an integer, while xi is an numeric vector and sdp is a numeric vector or matrix.
- npar is an integer equal to the dimension of theta.

Value

A S3 object of class privacy.

---

plot.dpout *Plot dpout object.*

Description

Plot dpout object.

Usage

```
## S3 method for class 'dpout'
plot(x, ...)
```

Arguments

x	dp_out object.
...	optional arguments to mcmc_trace().

Value

trace plots.

`summary.dpout` *Summarise dpout object.*

Description

Summarise dpout object.

Usage

```
## S3 method for class 'dpout'
summary(object, ...)
```

Arguments

`object`	dp_out object
`...`	optional arguments to `summarise_draws()`.

Value

a summary table of MCMC statistics.

Index