bayesGDS small test example 2

Michael Braun

March 30, 2015

```
require(reshape2)
require(plyr)
require(dplyr)
require(ggplot2)
require(bayesGDS)
set.seed(123)
theme_set(theme_bw())
```

This vignette is a test case of how to use the Braun and Damien (2015) algorithm to estimate a univariate posterior distribution.

The model is

$$
\begin{aligned}
x &\sim N(\mu, 1/\tau) \\
\mu &= 100 \\
\tau &\sim gamma(.001, .001)
\end{aligned}
$$

τ is a precision parameter. We are estimating $\theta = \log \tau$, so $\mathrm{var}(x) = \exp(-\theta)$.

The simulated dataset is 20 observations of x. The true standard deviation of x is 5.

```
x <- rnorm(20, mean=100, sd=5)
nX <- length(x)
```

The log data likelihood, log prior, and log posterior are computed by the following functions. We assume that the mean μ is known.

```
## log-likelihood function
logL <- function(theta){
    sum( dnorm(x, mean=100, sd=exp(-.5*theta), log=TRUE) )
}

logPrior <- function(theta){
    dgamma(exp(theta), shape=0.001, scale=1000, log=TRUE) + theta
}

## Unnormalized log-posterior distribution function.
logPosterior <- function(theta){
    logL(theta) + logPrior(theta)
}
```

This problem has an analytical solution. We can sample from the posterior distribution of τ with the following function. We will use this function to compare our estimates with "truth."

```
n.draws <- 10000
tauPost.true <- rgamma(n.draws,
                       shape=nX/2+0.001,
                       scale=1000/(500*sum((x-100)^2)+1)
                       )
```

Running the algorithm

The first phase of the algorithm is to find the posterior mode θ^*. The Hessian at the mode is H^*.

```
theta0 <- log(1/var(x))
fit0 <- optim( theta0,
              function(th){ -logPosterior(th) },
              method="BFGS",
              hessian=TRUE )

theta.star <- fit0$par       ## Posterior mode
H.star <- as.vector(fit0$hessian)
H.star.inverse <- 1/H.star   ## Inverse hessian of the posterior at the mode

  ## Value of log-posterior at theta.star
log.c1 <- logPosterior(theta.star)
```

Next, we sample M values from a proposal distribution $g(\theta)$. We will use a normal distribution as the proposal. The proposal mean is θ^*, and the proposal variance is $-sH^{*-1}$, where $s = 1.8$. This choice of s is the smallest value that generates a valid proposal. We set M to be large to reduce approximation error.

The `sample.GDS` function requires the proposal functions to take distribution parameters as a single list, so we need to write some wrapper functions.

```
logg <- function(theta, params){
    dnorm(theta, mean=params[[1]], sd=params[[2]], log=TRUE)
}
## Draw samples from the proposal distribution.
draw.norm <- function(N, params){
    rnorm(N, mean=params[[1]], sd=params[[2]])
}

M <- 20000
sSq <- 1.8
prop.params <- list(mean=theta.star,
                    sigma = sqrt(sSq*H.star.inverse)
                    )

## Value of log(g(theta.star))
log.c2 <- logg(theta.star, params=prop.params)
thetaM <- draw.norm(M, prop.params)
log.post.m <- sapply(thetaM, logPosterior)
log.prop.m <- sapply(thetaM, logg, params=list(theta.star, sqrt(sSq*H.star.inverse)))
log.phi <- log.post.m - log.prop.m + log.c2 - log.c1
valid.scale <- all(log.phi <= 0)
stopifnot(valid.scale)
```

Now, we can sample from the posterior.

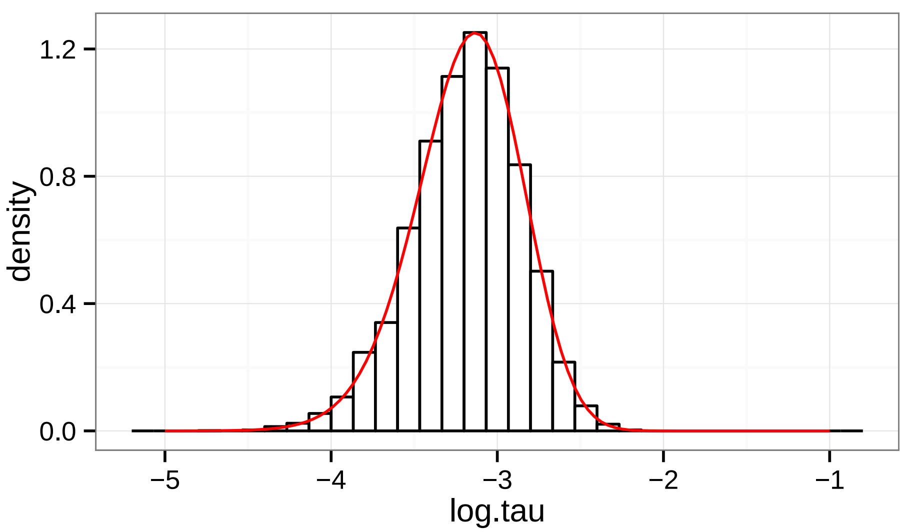
```
draws <- sample.GDS(n.draws = n.draws,
                    log.phi = log.phi,
                    post.mode = theta.star,
                    fn.dens.post = logPosterior,
```

```
                     fn.dens.prop = logg,
                     fn.draw.prop = draw.norm,
                     prop.params = prop.params,
                     announce = FALSE,
                     report.freq = 100
                     )
```

Checking results

The following plot compares the estimated posterior distribution of $\theta = \log \tau$ with samples from the analytically-derived posterior.

```
D <- data_frame(log.tau = draws$draws[,1])
G <- data_frame(log.tau = seq(-5, -1, length=100),
                y= dgamma(exp(log.tau),
                    shape=nX/2+0.001,
                    scale=1000/(500*sum((x-100)^2)+1))*exp(log.tau)
           )
P <- ggplot(D, aes(x=log.tau, y=..density..)) %>%
    + geom_histogram(fill="white", color="black")  %>%
        + geom_line(aes(x=log.tau, y=y), G, color="red")
P
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



We can also compare the point estimates and 95% intervals for the standard deviation. The true standard deviation is $1/\sqrt{\tau} = 5$. `freq` is the frequentist confidence interval, `BD` is this method, and `true` is the analytical solution.

```
## Point estimate and 95% credible interval for the standard deviation using GDS method

W <- data_frame(
    BD = exp(-.5*draws$draws[,1]),
    true = 1/sqrt(tauPost.true)
)
```

```r
freq <- sd(x)*c(sqrt( (nX-1)/qchisq(c(.975,.5, .025), nX-1)))
quants <- melt(cbind(apply(W,2, quantile, p=c(.025, .5, .975)), freq))
colnames(quants) <- c("quantile","method","value")
tab <- dcast(quants, method~quantile)
knitr::kable(tab, digits=rep(3,4))
```

method	2.5%	50%	97.5%
BD	3.680	4.877	6.961
true	3.687	4.886	6.917
freq	3.699	4.950	7.103

A simple table illustrates the efficiency of the method.

```r
tmp <- table(draws$counts)
tab <- data.frame(count=c(1:9,"10+"), draws=c(tmp[1:9],sum(tmp[10:length(tmp)])))
knitr::kable(tab, row.names=FALSE)
```

count	draws
1	8177
2	1183
3	351
4	131
5	67
6	32
7	12
8	11
9	8
10+	28

```r
acc.rate <- 1/mean(draws$counts)
acc.rate
## [1] 0.74722
```

We can use the method to estimate the log marginal likelihood of the data under the model.

```r
LML <- get.LML(counts=draws$counts,
               log.phi = log.phi,
               post.mode = theta.star,
               fn.dens.post = logPosterior,
               fn.dens.prop = logg,
               prop.params = prop.params
               )
LML
## [1] -66.771
```