# bayesGDS small test example 1

*Michael Braun*

*March 30, 2015*

This vignette is a test case of how to use the Braun and Damien (2015) algorithm to estimate a univariate posterior distribution. The other vignettes provide more detailed explanations about how to use the *bayesGDS* package.

First, let's load some packages we need. We will use *MCMCpack* to compare results with an MCMC simulation, and to evaluate an inverse gamma prior.

```r
require(reshape2)
require(plyr)
require(dplyr)
require(mvtnorm)
require(MCMCpack)
require(bayesGDS)
set.seed(1234)
```

The model is

$$x \sim N(\mu, \sigma^2)\mu \sim N(\mu_0, s_0)\sigma^{-2} \sim InvGamma(a, b)$$

The simulated dataset is 20 observations of $x$. The true mean of $x$ is $\mu = 100$, and the standard deviation is $\sigma = 5$.

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

The values for the prior parameters are:

```r
mu0 <- 0
s0 <- 1000
a <- .0001
b <- .0001
```

The log data likelihood, log prior, and log posterior are computed by the following functions. Since $\sigma$ must be positive, we estimate $\log \sigma$ instead. The factory function `makeLogPosterior` returns a closure `logPosterior`, which is a function that takes only the variables of interest as the lone argument. This approach simplifies life because we will not have to pass the prior parameters at subsequent function calls.

```r
logL <- function(theta){
    sum(dnorm(x, mean=theta[1], sd=exp(theta[2]), log=TRUE))
}

logPrior <- function(theta, mu0, s0, a, b){
    r1 <- dnorm(theta[1], mean=mu0, sd=s0, log=TRUE)
    r2 <- MCMCpack::dinvgamma(exp(2*theta[2]), a, b)
    res <- r1 + r2 + theta[2]
    return(res)
}

makeLogPosterior <- function(...) {
    function(theta){ logL(theta) + logPrior(theta, ...)}
}

logPosterior <- makeLogPosterior(mu0=mu0, s0=s0, a=a, b=b)
```

## Running the algorithm

The first phase of the algorithm is to find the posterior mode $\theta^*$. The Hessian at the mode is $H^*$. Since we are actually minimizing the negative log posterior, the Hessian will be positive definite at the optimum.

```
theta0 <- c( mean(x), log(sd(x)) )
fit0 <- optim(theta0,
              fn = function(th) {-logPosterior(th)},
              hessian=TRUE
              )

theta.star <- fit0$par ## posterior mode
H.star <- fit0$hessian
H.star.inverse <- solve(fit0$hessian)
log.c1 <- logPosterior(theta.star)
```

Next, we sample $M$ values from a proposal distribution $g(\theta)$. We will use a multivariate normal distribution with mean $\theta^*$, and covariance $sH^{-1*}$. The scalar $s$ is a scaling function to ensure that the proposal distribution is valid.

We set $s = 2.9$, which is the smallest value that generates a valid proposal. We found this value by adaptive search (i.e., "trial and error"). Larger values have no substantial effect on the results, but the acceptance rate of the sampler is lower.

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

```
## proposal functions
logg <- function(theta, prop.params){
    dmvnorm(theta,
            mean=prop.params[[1]], sigma=prop.params[[2]],
            log=TRUE)
}
drawg <- function(N, prop.params) {
    rmvnorm(N, mean=prop.params[[1]],
            sigma = prop.params[[2]]
            )
}
M <- 50000
s <- 2.9
log.c2 <- logg(theta.star, prop.params=list(theta.star, s*H.star.inverse))
prop.params <- list(mean=theta.star,
                    sigma = s*H.star.inverse
                    )

thetaM <- drawg(M, prop.params)
log.post.m <- apply(thetaM, 1, logPosterior)
log.prop.m <- apply(thetaM, 1, logg, prop.params=prop.params)
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.

```
n.draws <- 1000
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 = drawg,
                    prop.params = prop.params,
                    announce=FALSE,
                    report.freq = 1000
                    )
dimnames(draws$draws) <- list(draw=1:n.draws, var=c("mu","log.sigma"))
fr1 <- data.frame(draws$draws) %>%
  dplyr::mutate(method="BD", sigma=exp(2*log.sigma)) %>%
  dplyr::select(-log.sigma)
```

## Checking results

The posterior distribution does not have an analytic form, but we can compare what we get to MCMC results. The quantiles are reasonably close.

```
fit.mcmc <- MCMCregress(x~(1), mcmc=10000,burnin=5000,
                        b0 = mu0, B0 = 1/(s0^2),
                        c0=2*a, d0=2*b)
dimnames(fit.mcmc) <- list(draw=1:NROW(fit.mcmc), var=c("mu","sigma"))
fr2 <- data.frame(fit.mcmc) %>% mutate(method="MCMC")
fr <- rbind(melt(fr1), melt(fr2))
```

```
## Using method as id variables
## Using method as id variables
```

```
quants <- ddply(fr,c("variable","method"),
                function(X) quantile(X$value, probs=c(.05,.25,.5,.75,.95)))
knitr::kable(quants)
```

| variable | method | 5% | 25% | 50% | 75% | 95% |
|---|---|---|---|---|---|---|
| mu | BD | 96.784 | 97.976 | 98.762 | 99.556 | 100.730 |

| variable | method | 5% | 25% | 50% | 75% | 95% |
|---|---|---|---|---|---|---|
| mu | MCMC | 96.737 | 97.971 | 98.755 | 99.515 | 100.683 |
| sigma | BD | 17.123 | 22.225 | 28.508 | 35.870 | 52.579 |
| sigma | MCMC | 16.256 | 21.356 | 26.696 | 33.646 | 48.841 |