Implementing the Hamiltonian Monte Carlo Sampling algorithm in PROC MCMC

Elsa Vazquez Arreola, Jeffrey R Wilson, Arizona State University

ABSTRACT

Bayesian statistical methods are widely being used in a variety of fields, such as biomedical research, public health, social sciences and banking, among others. Researchers may fit Bayesian models in SAS using the PROC MCMC procedure. However, default options offered to researchers for sampling algorithms for Markov Chains in this procedure are based on the Metropolis and Metropolis-Hastings sampling methods and can sometimes be inefficient and take a high number of iterations to converge. The Hamiltonian Monte Carlo algorithm is an alternative sampling method that usually converges faster than Metropolis approaches. We show how to use this algorithm in PROC MCMC using the ALG statement and compare its performance with the default option using clustered data to fit a hierarchical logistic regression model.

INTRODUCTION

Bayesian statistical methods are being applied in several fields, such as public health, psychology, education, social sciences, banking and management, among others. These methods can be used when sample sizes are small or when usual methods do not converge. They can also be utilized when investigators have prior knowledge about how the system they are modeling works and want to include it in their inference or predictive models. Some advantages of Bayesian statistical methods are that their results are easy to interpret and understand and that they can be used to fit a wide variety of models.

In Bayesian statistics, inferences for parameters of interest $(\boldsymbol{\theta})$ are summarized through random draws from the posterior distribution of such parameters, $p(\boldsymbol{\theta}|\boldsymbol{y})$. This posterior distribution depends on the likelihood function of the data $p(\boldsymbol{y}|\boldsymbol{\theta})$ and the prior distribution of the parameters, $p(\boldsymbol{\theta})$, such that

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})$$

The goal of Bayesian computation is to obtain sets of independent draws $\boldsymbol{\theta^t}$ with $t = 1, \ldots, T$, from the posterior distribution, with enough draws T so that any function of the parameters, $g(\theta)$, can be estimated with reasonable accuracy (Gelman et.al, 2014). Bayesian computations are usually done using Markov Chain Monte Carlo (MCMC) methods.

There is a wide range of Markov Chain Monte Carlo methods available in PROC MCMC, by default, this procedure uses Metropolis based samplers. However, it has the capability of using more advanced samplers such as Hamiltonian Monte Carlo, if instructed to do so. In the next section, we describe the Metropolis, Metropolis-Hastings and Hamiltonian Monte Carlo sampling algorithms. We then fit a Bayesian hierarchical logistic regression model using the default sampler and the Hamiltonian Monte Carlo sampling algorithm in PROC MCMC and compare their performance.

MARKOV CHAIN MONTE CARLO SAMPLING ALGORITHMS

Markov Chain Monte Carlo methods can be used to generate draws from a distribution that approximates the posterior, $p(\boldsymbol{\theta}|\boldsymbol{y})$, when such distribution can be evaluated but not easily sampled from (Givens and Hoeting, 2013). All these methods are based on Markov Chains, which are sequences of vectors composed of random variables $\{\boldsymbol{\theta}^{(t)}\}$, $t = 0,1,2, \ldots$, where the next observed values $\boldsymbol{\theta}^{(t+1)}$ are only dependent on the present values $\boldsymbol{\theta}^{(t)}$. When using MCMC methods, we build Markov Chains that start at some point, $\boldsymbol{\theta}^{(0)}$, and then for each iteration t ($t = 1,2,\dots,T$), we draw values $\boldsymbol{\theta}^{(t)}$ from a transition or proposal distribution $J\left(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t-1)}\right)$ which depends on the previous values $\boldsymbol{\theta}^{(t-1)}$. The proposal distribution, $J\left(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t-1)}\right)$, must be constructed to guarantee that the Markov Chains converge to a unique stationary distribution that is the posterior, $p(\boldsymbol{\theta}|\boldsymbol{y})$. According to Gelman, et.al. (2014), the key aspect in MCMC methodology is to create Markov processes whose stationary distribution is $p(\boldsymbol{\theta}|\boldsymbol{y})$. Thus, for a sufficiently large number of simulations, the chain corresponding to such Markov process will have a marginal distribution that approximates the posterior (Givens & Hoeting, 2013).

When conducting statistical inferences using MCMC methods, one must check the convergence of the Markov Chains. It is usually the case that initial values of the Markov Chains are not close to the values of the target distribution of the parameters, and as such must be discarded to avoid their influence on estimated parameters, this process is known as burn-in. It is recommended that burn-in stops once convergence to a stationary distribution so that all random draws used for inference come from the target distribution. There are several methods to determine whether a Markov Chain has converged available in PROC MCMC. Two of these are visual tests and effective sample sizes. Visual tests consist of looking at the trace plots of the Markov Chains and determining whether these plots suggest convergence to a stationary distribution. Effective sample sizes are used to determine the number of independent samples drawn through the Markov Chains, having effective sample sizes of at least 100 is recommended to make valid inferences about the parameters of interest, higher effective sample sizes indicate higher precision in inferences (Givens & Hoeting, 2013). We now describe the steps followed in Metropolis, Metropolis-Hastings and Hamiltonian Monte Carlo sampling algorithms. We then compare their performance using the visual tests and effective sample sizes to asses convergence of the chains.

METROPOLIS ALGORITHM

The metropolis sampling algorithm is mostly used when there are no conjugate priors for parameters of interest. For only one parameter to be estimated, it proceeds by sampling a new proposed value θ^* that is close to the previous value $\theta^{(t-1)}$ using a symmetric proposal distribution. A proposal distribution $J\,(.\,|.)$ is symmetric if $J(\theta_b|\theta_a) = J(\theta_a|\theta_b)$ meaning that the probability of going from θ_a to θ_b is the same as the probability of going from θ_b to θ_a. The most common proposal distributions used in the Metropolis algorithm are

$$J\left(\theta^{(t)}|\theta^{(t-1)}\right) = Uniform(\theta^{(t-1)} - \delta, \theta^{(t-1)} + \delta)$$

$$J\left(\theta^{(t)}|\theta^{(t-1)}\right) = Normal(\theta^{(t-1)}, \delta^2)$$

The Metropolis algorithm with T random draws starts by sampling a starting random value, $\theta^{(0)}$, from the proposal distribution such that $p\left(\theta^{(0)}|y\right) > 0$. Then, according to Hoff (2009) at each iteration t ($t = 1,\dots,T$) the following process is conducted:

1. Sample $\theta^* \sim J\left(\theta^*\middle|\theta^{(t-1)}\right)$, where θ^* is a new proposed value that the parameter might take at iteration t
2. Compute acceptance ratio

$$r = \frac{p(\theta^*|\boldsymbol{y})}{p(\theta^{(t-1)}|\boldsymbol{y})} \qquad (1)$$

Where $p(\theta^*|\boldsymbol{y})$ represents the posterior probability of the proposed value θ^* and $p\left(\theta^{(t-1)}|\boldsymbol{y}\right)$ is the posterior probability of the parameter value at the previous iteration, $\theta^{(t-1)}$.

3. Sample $u \sim Uniform(0,1)$
   If $u < r$ then $\theta^{(t)} = \theta^*$
   If $u > r$ then $\theta^{(t)} = \theta^{(t-1)}$

For step 3, if the posterior probability of the proposed value θ^* is higher than that of the previous value $\theta^{(t-1)}$, then $r > 1$ and the proposed value will certainly be part of our sample of values. However, if $r < 1$ there will be times when we will take the proposed value θ^* as our new value $\theta^{(t)}$ and others when we will keep the previous value such that $\theta^{(t)} = \theta^{(t-1)}$. Thus, there is the possibility that we will have the same value for the parameter for several consecutive iterations.

METROPOLIS-HASTINGS ALGORITHM

The Metropolis-Hastings algorithm is also mostly used when there are no conjugate prior distributions and follows the same procedure for building Markov Chains as the Metropolis algorithm. However, this method is different from the Metropolis algorithm in that the proposal distributions are usually not symmetric. Each iteration t $(t = 1, \ldots, T)$ consists of the following steps:

1. Sample $\theta^* \sim J\left(\theta\middle|\theta^{(t-1)}\right)$ where θ^* is the new value proposed for the parameter of interest and $J\left(\theta\middle|\theta^{(t-1)}\right)$ is not necessarily symmetric
2. Compute acceptance ratio
$$r = \frac{p(\theta^*|\boldsymbol{y})}{p(\theta^{(t-1)}|\boldsymbol{y})} \times \frac{J\left(\theta^{(t-1)}|\theta^*\right)}{J(\theta^*|\theta^{(t-1)})} \qquad (2)$$
   Where $p(\theta^*|\boldsymbol{y})$ is the posterior probability of the proposed value θ^* given the data, $p\left(\theta^{(t-1)}|\boldsymbol{y}\right)$ is the posterior probability of the parameter's previous value given the data, $J\left(\theta^*|\theta^{(t-1)}\right)$ is the probability of going from $\theta^{(t-1)}$ to θ^* under the proposal distribution and $J\left(\theta^{(t-1)}|\theta^*\right)$ represents the probability of going from θ^* to $\theta^{(t-1)}$ also under the proposal.
3. Sample $u \sim Uniform(0,1)$
   If $u < r$ then $\theta^{(t)} = \theta^*$
   If $u > r$ then $\theta^{(t)} = \theta^{(t-1)}$

The Metropolis algorithm is a special case of the Metropolis-Hastings algorithm. Since the proposal distribution is symmetric and thus $J\left(\theta^{(t-1)}|\theta^*\right) = J\left(\theta^*|\theta^{(t-1)}\right)$, then in the Metropolis sampler the factor $\frac{J\left(\theta^{(t-1)}|\theta^*\right)}{J\left(\theta^*|\theta^{(t-1)}\right)} = 1$ which reduces the ratio in (2) to the ratio in (1).

As explained before, in Metropolis-based algorithms it might be the case that the proposed values θ^* are rejected in more than one consecutive draw and that we keep the same value $\theta^{(t)}$ for several consecutive iterations, causing autocorrelation among the simulated draws. This is due to the random walk behavior of the Metropolis-based algorithms and can make such sampling algorithms inefficient, especially when estimating high dimension parameter vectors. As such, there are situations in which more efficient algorithms are needed.

HAMILTONIAN MONTE CARLO ALGORITHM

Gelman et.al. (2013) define the Hamiltonian Monte Carlo algorithm as a generalization of the metropolis algorithm that subdues its random walk behavior and allows HMC to move faster in the parameter space by including momentum variables, resulting in faster mixing and convergence, especially for high dimensional parameter vectors. This algorithm is most used when there are several parameters to be estimated and the Metropolis algorithms moves slowly through the target distribution. It depends on a vector of auxiliary variables $\boldsymbol{\phi}$ of the same dimension as the vector of parameters $\boldsymbol{\theta}$ and the proposal distribution for $\boldsymbol{\theta}$ is based on $\boldsymbol{\phi}$.

In Hamiltonian Monte Carlo the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$ is combined with an independent distribution of the auxiliary variables $p(\boldsymbol{\phi})$ resulting in a joint distribution $p(\boldsymbol{\theta},\boldsymbol{\phi}) = p(\boldsymbol{\phi})p(\boldsymbol{\theta}|y)$. Random draws are obtained using this joint distribution, but interest is placed only on $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ values are discarded, since they are only added to move across the parameter space faster. Apart from the posterior distribution, this algorithm also requires the gradient of its log,

$$\frac{\partial \log p(\boldsymbol{\theta}|\boldsymbol{y})}{\partial \boldsymbol{\theta}} = \left(\frac{\partial \log p(\boldsymbol{\theta}|\boldsymbol{y})}{\partial \theta_1}, \frac{\partial \log p(\boldsymbol{\theta}|\boldsymbol{y})}{\partial \theta_2}, \dots, \frac{\partial \log p(\boldsymbol{\theta}|\boldsymbol{y})}{\partial \theta_k}\right)$$

The Hamiltonian Monte Carlo sampling algorithm is described in Gelman et.al (2013) as follows. For iteration t $(t = 1, \dots, T)$ steps taken are:

1. Sample $\boldsymbol{\phi}$ from $p(\boldsymbol{\phi})$, where $\boldsymbol{\phi} \sim multivariate\ normal(\mathbf{0}, \boldsymbol{M})$ and $\boldsymbol{M}$ is usually a diagonal matrix
2. Update $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ simultaneously repeating L leapfrog steps which consist of the following:
    a. Let $\boldsymbol{\phi} = \boldsymbol{\phi} + \frac{\varepsilon}{2}\frac{\partial \log p(\boldsymbol{\theta}|\boldsymbol{y})}{\partial \boldsymbol{\theta}}$
    b. Let $\boldsymbol{\theta} = \boldsymbol{\theta} + \varepsilon M^{-1}\boldsymbol{\phi}$
    c. Update $\boldsymbol{\phi}$ one more time by letting $\boldsymbol{\phi} = \boldsymbol{\phi} + \frac{\varepsilon}{2}\frac{\partial \log p(\boldsymbol{\theta}|\boldsymbol{y})}{\partial \boldsymbol{\theta}}$
3. Let $\boldsymbol{\theta}^{(t-1)}$ and $\boldsymbol{\phi}^{(t-1)}$ be the values that $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ had before starting the leapfrog process and $\boldsymbol{\theta}^*$ and $\boldsymbol{\phi}^*$ their values after the L leapfrog steps, the calculate the ratio

$$r = \frac{p(\boldsymbol{\theta}^*|\boldsymbol{y})}{p(\boldsymbol{\theta}^{(t-1)})}\frac{p(\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{(t-1)})}$$

4. Draw $u \sim uniform\ (0,1)$

    If $u < r$ then $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^*$

    If $u > r$ then $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$

In this algorithm ε is a scaling factor for the probability distribution of the momentum variables and is known as stepsize. Gelman et.al. (2013) recommend that the product of the number of leapfrog steps and the scaling factor $L\varepsilon = 1$. If convergence is not achieved, the number of leapfrog steps should be increased and the scaling factor should be reduced. Depending on the choices on distribution for the momentum variables, the scaling factor and the number of leapfrog step, the Hamiltonian Monte Carlo sampling algorithm might converge to the posterior distribution faster than the Metropolis-based algorithms, making it more efficient in Bayesian computation, but that might not always be the case.

COMPARING PERFORMANCE OF METROPOLIS-BASED AND HAMILTONIAN MONTE CARLO SAMPLING ALGORITMS

We compared the performance of default sampling algorithms in PROC MCMC which are based in the Metropolis sampler and the Hamiltonian Monte Carlo. We used data from the Adolescent to Adult Health Longitudinal Study (Add Health) to do this comparison. This dataset was collected longitudinally in four waves. Thus, observed outcomes were nested within respondents creating correlation. We fit a hierarchical logistic regression to model the impact of gender, depression scale and physical activity levels on obesity status while accounting for correlation among outcomes coming from same subject, through random effects. The model fit was:

$$logit\big(P(OS = 1|\boldsymbol{X}, \gamma_i)\big) = \beta_0 + \beta_1 * gender + \beta_2 * depression + \beta_3 * physicalact + \gamma_i$$

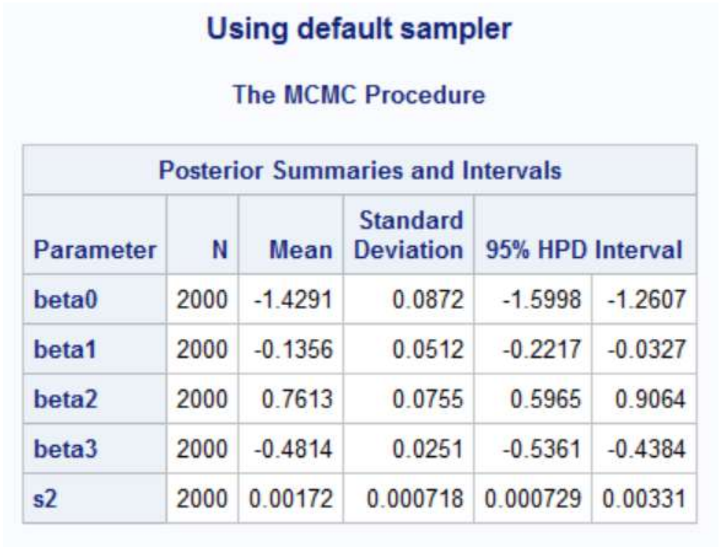
$$\gamma_i \sim normal(0, \sigma^2)$$

Where γ_i represents the random effect due to subject i and comes from a normal distribution with mean 0 and variance σ^2, gender is a dichotomous variable taking value 1 for males and 0 for females and depression and physical activity level are continuous variables. Using this model, we estimate 5 parameters β_0, β_1, β_2, β_3 and σ^2.

It is important to note that the structure of the code to fit a hierarchical logistic regression model is the same regardless of the sampling algorithm used. However, in the PROC MCMC statement there is an option ALG where one can specify the sampling algorithm to be used, if no ALG option is provided, the default sampler is utilized. The following code is used to fit this model using default sampler in PROC MCMC:

```
title 'Using default sampler';
proc mcmc data=add_health nbi=2000 nmc=2000 seed=189;
parms beta0 beta1 beta2 beta3 s2;
prior s2 ~ igamma(0.001, s=0.001);
prior beta0 ~normal(-1.5578, var=0.1330);
prior beta1 ~normal(-0.1784, var=0.08663);
prior beta2~ normal(0.8966, var=0.1068);
prior beta3~normal(-0.6632, var=0.03235);
mu=beta0+beta1*gender+beta2*feelingscale+beta3*activityscale;
random delta~normal(0, var=s2) subject=id;
pi=logistic(mu+delta);
model bmi~binary(pi);
run;
```

We use 2,000 burn-in iterations for the default sampler and obtained 2,000 iterations after burn-in without thinning resulting in a chain length of 2,000. We used results from fitting a frequentist hierarchical model to get priors for all parameters in our model. Output 1 shows the estimated means, as well as the 95% highest posterior density (HDP) credible intervals for the parameter in our model. After running this code, we noticed that the trace plots for the Markov Chains indicated that there had been convergence for the regression parameters, but not for the variance of random effects, Figure 1. We also observed that the effective sample size for each of the regression coefficients was higher than 100, but for the parameter related to the random effects' variance it was much lower, Output 2.


Using default sampler

The MCMC Procedure

Posterior Summaries and Intervals					
Parameter	N	Mean	Standard Deviation	95% HPD Interval	
beta0	2000	-1.4291	0.0872	-1.5998	-1.2607
beta1	2000	-0.1356	0.0512	-0.2217	-0.0327
beta2	2000	0.7613	0.0755	0.5965	0.9064
beta3	2000	-0.4814	0.0251	-0.5361	-0.4384
s2	2000	0.00172	0.000718	0.000729	0.00331

Output 1. Output from PROC MCMC with default sampler for estimated parameters and credible intervals

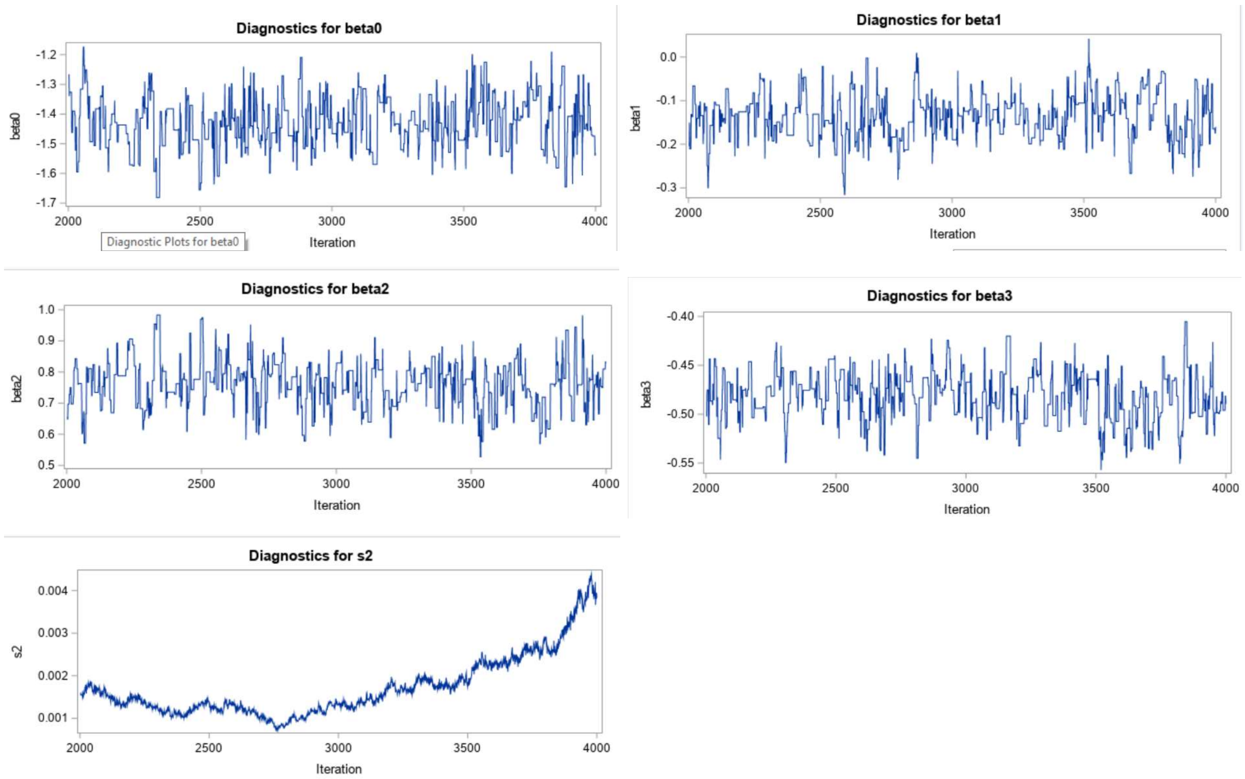
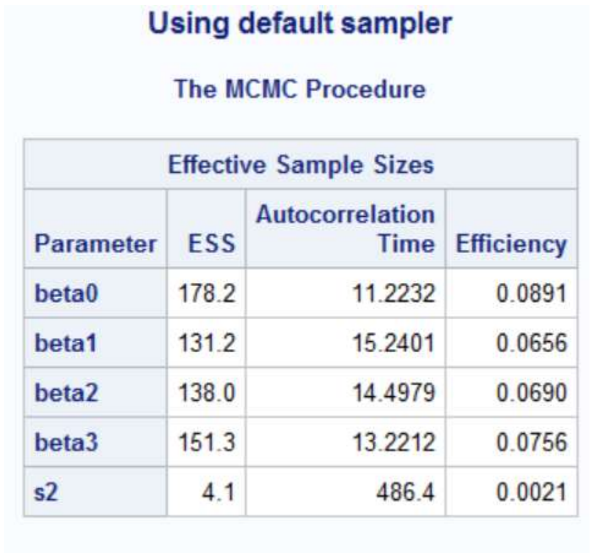


Figure 2. Trace plots for Markov Chains for parameters when estimated using default sampler



Using default sampler

The MCMC Procedure

Effective Sample Sizes			
Parameter	ESS	Autocorrelation Time	Efficiency
beta0	178.2	11.2232	0.0891
beta1	131.2	15.2401	0.0656
beta2	138.0	14.4979	0.0690
beta3	151.3	13.2212	0.0756
s2	4.1	486.4	0.0021

Output 2. Output from PROC MCMC with default sampler for estimated parameters' ESS and autocorrelation

We fit the hierarchical logistic regression model with the Hamiltonian Monte Carlo algorithm with the following code. Here, we specified that we wanted to use to this sampler using the ALG option as follows ALG=HMC (*nsteps*= *stepsize*=), where *nsteps* indicates the number of leapfrog steps and *stepsize* indicates the scaling factor for the probability distribution of the momentum variables:

```
title 'Using Hamiltonian Monte Carlo';
proc mcmc data=add_health alg=hmc(nsteps=1000 stepsize=0.001) nbi=2000
nmc=2000 seed=189;
parms beta0 beta1 beta2 beta3 s2;
prior s2 ~ igamma(0.001, s=0.001);
prior beta0 ~normal(-1.5578, var=0.1330);
prior beta1 ~normal(-0.1784, var=0.08663);
prior beta2~ normal(0.8966, var=0.1068);
prior beta3~normal(-0.6632, var=0.03235);
mu=beta0+beta1*gender+beta2*feelingscale+beta3*activityscale;
random delta~normal(0, var=s2) subject=id;
pi=logistic(mu+delta);
model bmi~binary(pi);
run;
```

We run a total of 2,000 burn-in iterations followed by 2,000 sampling iterations without thinning resulting in a chain length of 2,000. We tuned the Hamiltonian Monte Carlo algorithm using 1,000 leapfrog steps and a step size of 0.001. We used the same prior distributions that we used when fitting the model with the default samplers for all the parameters of interest. The estimated means of the parameters of interest and their 95% highest posterior density credible intervals are shown in Output 3. After using the Hamiltonian Monte Carlo sampler with the above specifications, we noticed that the trace plots for the Markov Chains indicated that there had been convergence for the regression parameters, but not necessarily for the variance of random effects, Figure 2. The trace plots for this Markov Chains also indicated that the Hamiltonian Monte Carlo sampler did not move through the parameter space that fast, leading to parameter values staying the same for several continuous iterations. We also observed that the effective sample sizes for each of the regression coefficients and the variance of the random effects were lower than 100, Output 4. The effective sample sizes indicated that we could not make reliable inferences about the parameters that we were interested on.

Output 3. Output from PROC MCMC with Hamiltonian Monte Carlo sampler for estimated parameters and credible intervals

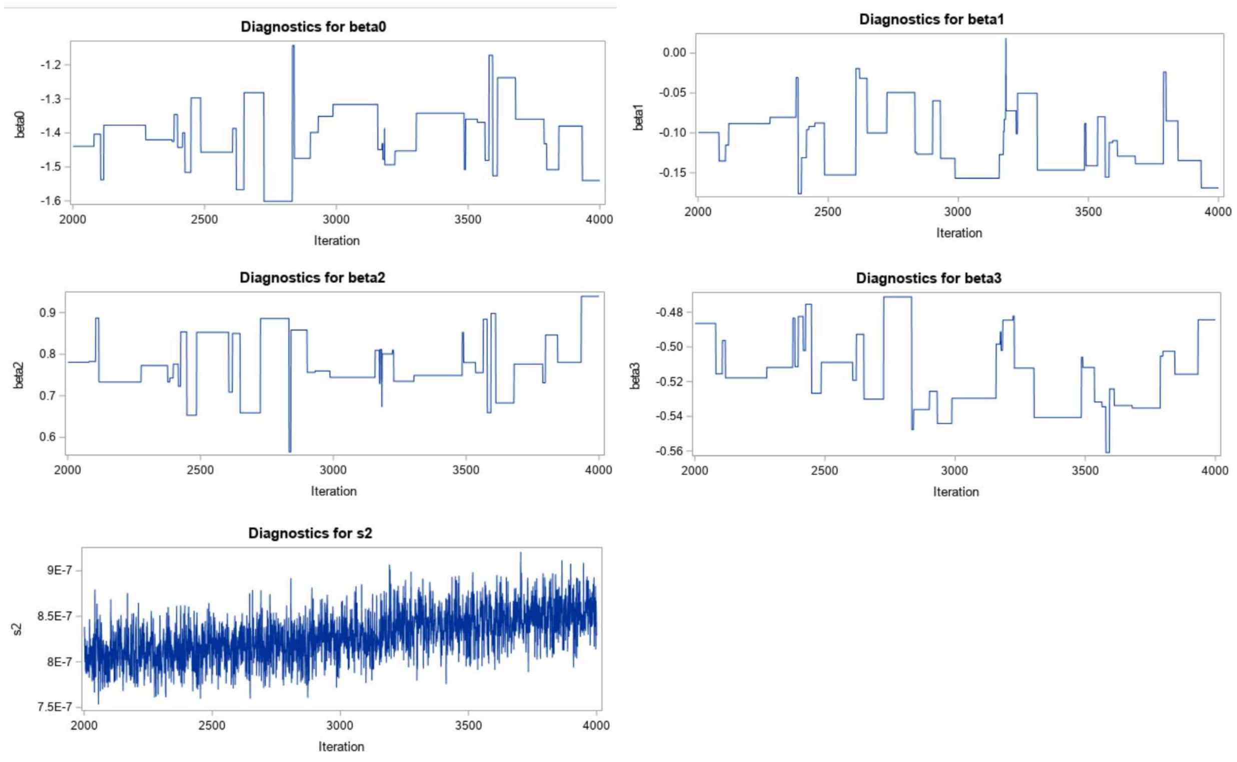
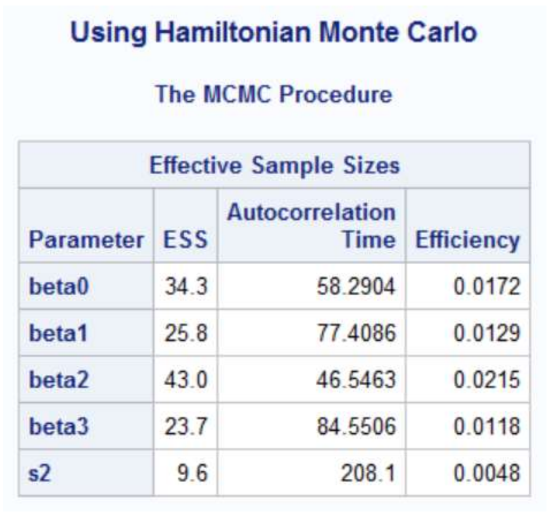


Figure 3. Trace plots for Markov Chains for parameters when estimated using Hamiltonian Monte Carlo sampler



Using Hamiltonian Monte Carlo

The MCMC Procedure

Effective Sample Sizes			
Parameter	ESS	Autocorrelation Time	Efficiency
beta0	34.3	58.2904	0.0172
beta1	25.8	77.4086	0.0129
beta2	43.0	46.5463	0.0215
beta3	23.7	84.5506	0.0118
s2	9.6	208.1	0.0048

Output 4. Output from PROC MCMC with Hamiltonian Monte Carlo sampler for estimated parameters' ESS and autocorrelation

After using both the default sampler in PROC MCMC and the Hamiltonian Monte Carlo sampling algorithm with the same number of burn-in and sampling iterations and with the same prior distributions for all parameters, we found that the default sampler outperformed the Hamiltonian Monte Carlo sampler in terms of effective sample sizes for the regression coefficients. However, for the variance of the random effects, the Hamiltonian Monte Carlo sampler had higher effective sample sizes than the default sampler and for both algorithms the effective sample sizes for this parameter were smaller 100. For the regression coefficients, the default sampler had effective sample sizes higher than 100 suggesting that we could make trustful inference about these parameters, that was not the case for the Hamiltonian Monte Carlo sampler where effective sample sizes were smaller than 100. Also, when looking at the trace plots for the Markov Chains for both sampling algorithms, we noticed that the Hamiltonian Monte Carlo sampler got stuck in the same values of the regression parameters longer than the default sampler, making it more inefficient. Through this data example, where we fit a hierarchical logistic regression model using both samplers, we observed that although the Hamiltonian Monte Carlo algorithm is designed to be more efficient and converge faster than the Metropolis and Metropolis-Hastings algorithms that might not always be the case. The ALG option in PROC MCMC lets users utilize the Hamiltonian Monte Carlo sampler, however there might be cases were the default sampler is more efficient.

CONCLUSION

Bayesian statistical methods provide opportunities to obtain models that cannot be fit or do not provide reliable answers under frequentist approaches because of small sample sizes or complexity of the models. Inferences under Bayesian approaches are done through simulated draws from the posterior distribution of the parameters which are obtained using different Markov Chain Monte Carlo simulation methods. SAS users may fit Bayesian models using PROC MCMC. This procedure uses Metropolis or Metropolis-Hasting sampling algorithms by default, but can also be adapted to fit models using other algorithms. We showed how to modify the PROC MCMC statement to use the Hamiltonian Monte Carlo algorithm and used this procedure to fit a hierarchical logistic regression model. We compared results obtained using default and Hamiltonian Monte Carlo sampling algorithms and found that in this particular case the default sampler outperformed the Hamiltonian Monte Carlo algorithm. We recommend that when using the Hamiltonian Monte Carlo sampler and finding that it is not converging as fast as expected or that the effective sample sizes are not high enough to make trustful inferences, researchers also use the default sampler in PROC MCMC and check how the results change.

REFERENCES

Hoff, Peter D. 2009*. A First Course in Bayesian Statistical Methods*. New York, NY: Springer

Givens, Geof H., Hoeting, Jennifer A. 2013. *Computational Statistics Second Edition*. Hoboken, NY: John Wiley & Sons, Inc.

Gelman, Andrew, Carlin, John B., Stern, Hal S., Dunson, David B., Vehtari, Aki and Rubin, Donald B. 2014. *Bayesian Data Analysis Third Edition*. Boca Raton, FL: CRC Press Taylor & Francis Group.

Geweke, J. Evaluating the accuracy of sampling-based approaches to calculating posterior moments. In *Bayesian Statistics 4* (ed JM Bernado, JO Berger, AP Dawid and AFM Smith). Clarendon Press, Oxford, UK.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Elsa Vazquez Arreola
Arizona State University
maria.vazquez@asu.edu

Jeffrey R. Wilson
Arizona State University
jeffrey.wilson@asu.edu