Using PROC MCMC in a Process Control Setting: An Illustrative Example with PROC IML

Austin Brown, M.S., M.M.P., University of Northern Colorado

Abstract
When an experienced observer in a specific discipline examines some process or phenomenon, it is likely that they have prior knowledge about the characteristics and behaviors of what it is they are monitoring. This knowledge could be useful in calculating estimates for the parameters of this process. Leveraging the prior knowledge of parameter behavior along with observed data from the process or phenomenon being observed is the idea upon which Bayesian Estimation is based. In this presentation, attendees will be shown, by example, basic functionality of PROC MCMC to obtain Bayesian estimates, including examining diagnostics for the estimators. Attendees will also be shown an example of how to use Bayesian Estimation via PROC MCMC and PROC IML in a process control environment. This example will highlight some of the advantages of Bayesian Estimation over traditional Maximum Likelihood Estimation, especially in small sample size cases, as is often the case in process control settings.

Introduction
In quantitative analysis, researchers are frequently interested in estimating some unknown parameter or vector of parameters of interest by collecting numeric data and calculating point estimates. These estimates can be used for interval estimation as well as hypothesis testing. Classically, these estimates are found via frequentist estimation techniques, such as maximum likelihood (ML). ML estimation relies solely upon the random sample from the population being studied. This means for some population with a probability density function (PDF), $p(y|\theta)$, that only information gathered from the random variable itself is considered when estimating the parameter vector, θ. If a researcher has specific knowledge about the probabilistic behavior of θ, such as a likely range of values between which it may fall, or a specific distributional assumption (i.e., a PDF for θ, $p(\theta)$), this cannot be used to help estimate θ. Despite the many desirable qualities of ML estimators, the inability to incorporate prior knowledge of the parameters being estimated into the calculation may be quite limiting. The idea of using prior knowledge in the estimation of and inference with respect to a population's parameters is the basis upon which Bayesian Statistics rests.

In Bayesian Statistics, researchers assume prior knowledge about θ. As previously mentioned, this prior knowledge is quantified probabilistically in a PDF for θ, denoted $p(\theta)$.

Using the classical Bayes' Rule found in probability, researchers are able to find a "posterior" distribution for θ which is conditioned upon data gathered in the random sample. Mathematically:

$$p(\theta|y) = \frac{p(y,\theta)}{p(y)} = \frac{p(\theta)p(y|\theta)}{p(y)} \tag{1}$$

Often times in practice, the denominator in Equation 1 can be ignored since the general structure of $p(\theta|y)$ can be determined by the product in the numerator. Thus, proportionality is often sufficient to determine the posterior distribution. That is:

$$p(\theta|y) \propto p(\theta)p(y|\theta) \tag{2}$$

In SAS, many procedures incorporate Bayesian estimation into regular statistical models, such as GENMOD, LIFEREG, PHREG, and FMM (Stokes, Chen, and Gunes, 2014). By using the "BAYES" statement in each of these procedures, SAS will use Bayesian estimation via the "Markov chain Monte Carlo" (MCMC) technique (2014). SAS also has a direct method of finding Bayesian estimates which provides users more control over choices of priors, possible usage of hyperparameters (parameters of the prior distribution which have their own distributions as well), in addition to some more technical control over how the MCMC is performed (i.e., burn-in, thinning, tuning parameters, etc.). This procedure is aptly named PROC MCMC. In some practical settings, such as a process control setting, use of PROC MCMC may be appropriate if the user desires to incorporate Bayesian estimation into their quality program.

PROC MCMC

The intent of this paper is to be illustrative rather than an in-depth explanation of the technical aspects of how PROC MCMC functions. A more robust explanation can be found in the SAS documentation ("The MCMC Procedure," n.d.). Below is example code where $y|\mu,\sigma \sim N(\mu = 10, \sigma = 10), \mu \sim N(\mu_0 = 10, \sigma_0 = 2), \sigma^2 \sim Scaled - Inv - \chi^2(\nu_0 = 10, s_0^2 = 8)$:

```
proc mcmc data=x outpost=simout1 nmc=10000 maxtune=0
nbi=0 statistics=(summary interval) diagnostics=all;
parms mu sigma;
prior mu ~ normal(15,sd=2);
prior sigma ~ sichisq(10,scale=8);
model x ~ normal(mu,var=sigma);
run;
```

Here, `x` is a dataset of some observations from $p(y|\theta)$. In this example:

```
%let n = 5;
data x;
do i = 1 to &n;
x = rand("normal",10,10); output;
end;
keep x;
run;
```

Without delving into the technicalities of how the MCMC procedure derives estimates of μ and σ^2, it is sufficient to say that it is necessary for the user to require PROC MCMC to have a fairly large number, and the default is 10,000 as given by `nmc=10000`. Depending upon the complexity of the models and the distributions of the parameters, hyperparameters, and likelihood, it may be necessary to increase this number to 20,000 or larger in order to achieve converging estimates. These 10,000 estimates can be worked with directly by giving PROC MCMC the `outpost=simout1` call. PROC MCMC follows the general form of other SAS procedures as users are able to output the coda (i.e., the individual estimates discussed above), as well as call specific statistics and diagnostics, the latter being important prior to drawing any conclusions about the posterior estimates. While it will not be discussed here, interested readers will be directed to SAS documentation on PROC MCMC for information regarding tuning parameters and burn-in, both of which are important for complex models ("The MCMC Procedure," n.d.).

After the PROC MCMC line, users can specify the parameters from the likelihood desired to be estimated. Here, `parms mu sigma;` provides PROC MCMC with this information. Next, much like in the classical WinBUGS programming language for Bayesian estimation, users specify the prior distributions placed upon the parameters named in the `parms` statement. Finally, the form of the likelihood is given by the `model` statement. PROC MCMC has many predefined distributions which can be used to model parameters, but it also worthy of note that the procedure has flexibility to work with any distribution via the `general` statement.

Above a basic PROC MCMC code is given that can be easily ran, and in this instance, Figure 1 gives the Bayesian estimates provided by SAS output. Here, PROC MCMC gives $\hat{\mu} = 11.4940$, and $\hat{\sigma^2} = 34.8973$, as shown in the "MEAN" column in the posterior summaries table. The output also provides users with the standard deviation and the first, second, and third quartiles. Notice here, the standard deviation for `sigma` is 17.4092, which seems somewhat large. Generally, it is better for this number to be smaller, but this can be evaluated in a couple of ways. The practical use of PROC MCMC requires assessing the validity of the estimates given. SAS gives a variety of statistical tests and graphs which can be used in conjunction with each other, similar to assessing normality or constant variance in classical normal linear models. Here, focus will be upon graphical assessment. Figure 2 gives the trace, autocorrelation, and density plots for $\hat{\mu}$ and $\hat{\sigma}^2$, respectively.

Figure 1: PROC MCMC Parameter Estimates

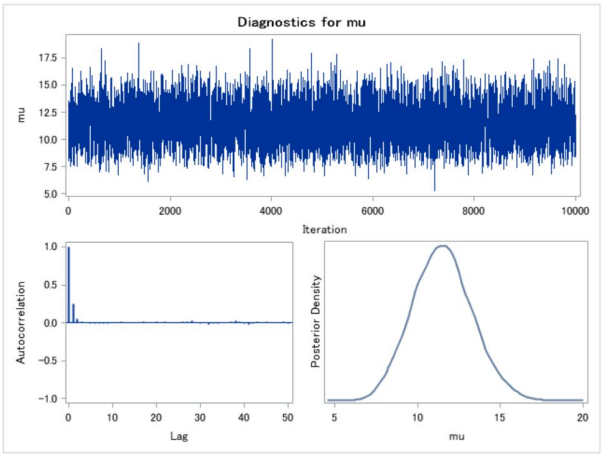
Posterior Summaries

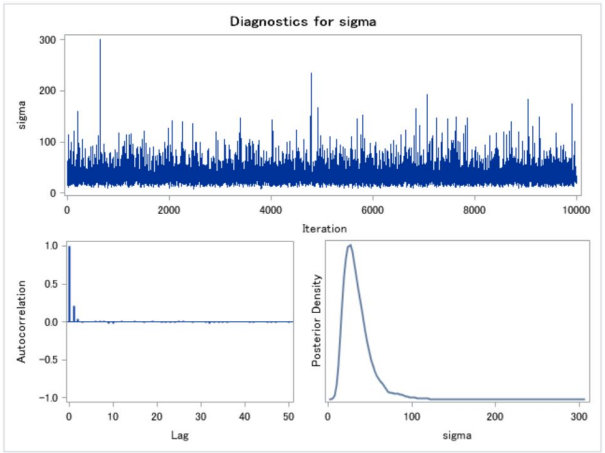
Parameter	N	Mean	Standard Deviation	Percentiles		
				25	50	75
mu	10000	11.4940	1.7800	10.2452	11.4549	12.6608
sigma	10000	34.8973	17.4092	23.4198	30.9324	41.6448

Posterior Intervals

Parameter	Alpha	Equal-Tail Interval		HPD Interval	
mu	0.050	8.1447	15.1556	7.8571	14.8231
sigma	0.050	14.4106	80.4767	11.1706	67.2450

Figure 2: Trace, Autocorrelation, and Density Plots for $\hat{\mu}$ & $\hat{\sigma}^2$





The top plot on both graphs is referred to as a "trace plot." For a posterior estimate with low variance, it is desired to observe what is shown by $\hat{\mu}$'s trace plot. Most of the observations stay between 7.5 and 15, and they all seem to be fairly randomly distributed about a center line of approximately 12.5, which is relatively close to the posterior mean estimate. On the other hand, inspecting $\hat{\sigma}^2$'s trace plot, we graphically observe what the large standard deviation in Figure 1 was showing. There are large outliers, which increases the standard deviation. This is an example of the power of the choice of prior distributions, and their associated hyperparameter values. If the user is sure of their use of a particular prior distribution, it is recommended to appropriately consider modifying the values of the hyperparameters or perhaps including considering hyperprior distributions. Otherwise, reconsideration of the use of a particular prior distribution may be warranted.

The bottom two graphs for each parameter are the autocorrelation plot and the density plots, from left to right. The autocorrelation plot shows correlations on the y-axis and "lags" on the x-axis. It is desirable for there to be little to no correlation between a particular estimate (i.e., one of the 10,000 estimates) and the other estimates any distance (in terms of lags) away from it. That is, it is most desirable to have the coda produced by the MCMC procedure

to be a white-noise time series. Observing the plot, we see the correlations quickly going to zero after only two lags, which is a promising result. Finally, we can observe the bottom right graph, which is the density plot from the coda. Here, we are mostly interested in observing smooth lines. If there are many "lumps" in the empirical density, this may be indicative of too much variation and high correlation between the estimates. In such cases, carefully considering prior distribution choices as well as the number of iterations performed is recommended.

Statistical Process Control Example

Statistical process control is a set of techniques organizations can implement with the ultimate goal of lowering variability in the product or service being provided. One such technique is quality control charting, where numeric data is aggregated into sample statistics, such as mean, and if an estimate falls above or below some threshold (called "control limits") calculated via the knowledge of the distribution of the sample statistic itself, the process being observed is deemed "out-of-control," and this signals to operators that investigation of some sort ought to take place. Because of the central limit theorem, it is well known that as $n \to \infty$, $\bar{x} \dot{\sim} N(\mu, \sigma^2/n)$. However, when only a small sample is taken, and it cannot necessarily be assumed that the underlying x_i's are normally distributed, it may be inappropriate to implement a charting technique which is based on normality. Additionally, because of the small sample size and because of the experience of business managers with a particular process, Bayesian estimation may be an attractive option.

While there are many instances where such techniques could be applied, below is code that could be easily manipulated for specific applications when implementation of a Bayesian quality control chart is desired. This code takes advantage of the functionality of PROC IML. First, an amply sized sample is taken in order to calculate the control limits.

```
proc iml;
x = j(1000,1,.);
do i = 1 to 1000;
x[i,1] = rand("normal",10,10); end;
create x var{x};
append;
close x;
```

Using the `submit` functionality of PROC IML, we can directly calculate the Bayesian estimates without having to "leave" the IML environment.

```
submit;
proc mcmc data=x outpost=simz nmc=10000 maxtune=0
nbi=0 statistics=summary diagnostics=none ;
parms mu sigma;
prior mu ~ normal(15,sd=2);
prior sigma ~ sichisq(10,scale=8);
model x ~ normal(mu,var=sigma);
run;
endsubmit;
```

Now, we are able to call the output dataset into IML defined vectors in order to work with the coda directly to calculate the upper and lower control limits. Here, we are able to use the `qntl` function which writes out matrices (here, `z1, z2`) of quantiles of a particular vector of observations.

```
use simz;
read all var {mu} into T1;
read all var {sigma} into T2;
p = {0.005, 0.995};
call qntl(z1, T1, p); /* compute 0.005th & 99.5th percentiles */
call qntl(z2, T2, p); /* compute 0.005th & 99.5th percentiles */
```

Now, Figure 3 shows our computed upper and lower control limits that can be used in a quality control charting technique.

Figure 3: Upper and Lower Control Limits for μ and σ^2

z1	z2
9.2943184	88.835915
10.888375	111.97474

At this point, we have obtained enough information to be able to actually determine if a given sample taken is above or below one of these limits. What we are interested in testing is if one of the parameters of the likelihood, in this case, $y \sim N(10, 10)$, has shifted. For the sake of clarity, we will keep σ^2 at its original value of 10, but we will see what happens if μ shifts by a multiplicative magnitude of 2. As shown in the code, we are making a very simple comparison of estimates given by the shifted mean distribution, and seeing if they fall above or below the upper and lower control limits, respectively. If a "1" is given, that means the process may be out-of-control for that given parameter. If a "0" is given, that means the process may still be in-control. Assuming the data collected during the control limit computation phase is accurate, and assuming proper prior and likelihood distributions

are assigned, this code provides a simple framework to examine if a process' parameters are likely in or out-of-control.

```
use simz1;
read all var mu into B1;
read all var sigma into B2;
Result = j(1,2,.);
B11 = j(1,1,.);
B21 = j(1,1,.);
B11[1,1] = mean(B1);
B21[1,1] = mean(B2);
if B11[1,1] < z1[1,1] then Result[1,1] = 1;
else if B11[1,1] > z1[2,1] then Result[1,1] = 1;
else Result[1,1]= 0;
if B21[1,1] < z2[1,1] then Result[1,2] = 1;
else if B21[1,1] > z2[2,1] then Result[1,2] = 1;
else Result[1,2] = 0;
print Result;
quit;
```

Conclusion

To conclude, PROC MCMC is an effective tool to construct Bayesian estimates. While the procedure has more than ample features for complex estimation techniques, its structure is still approachable and intuitive. With its variety of outputs, convergence of the parameter estimates can be easily assessed. Even without strong knowledge of the inner-workings of MCMC, it has been demonstrated that users can still gain the benefit of Bayesian estimation in practical settings. For experienced Bayesian statisticians, PROC MCMC has taken the familiar aspects of WinBUGS, but greatly enhanced them by allowing more flexibility in the model statements as well as producing many excellent statistical and graphical diagnostics. In short, PROC MCMC is scalable to the user's knowledge, experience, and needs.

References

Stokes, M., Chen, F., and Gunes, F. (2014). *An Introduction to Bayesian Analysis with SAS/STAT® Software*. Available at:
https://support.sas.com/resources/papers/proceedings14/SAS400-2014.pdf

SAS/STAT(R) 9.2 User's Guide, Second Edition. (2010, April 30). Available at:
https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htmmcmc_toc.htm

Contact Information

Austin Brown
University of Northern Colorado
austin.brown@unco.edu