6

Inference from Simulations and Monitoring Convergence

Andrew Gelman and Kenneth Shirley

Constructing efficient iterative simulation algorithms can be difficult, but inference and monitoring convergence are relatively easy. We first give our recommended strategy (following Section 11.10 of Gelman et al., 2003) and then explain the reasons for our recommendations, illustrating with a relatively simple example from our recent research: a hierarchical model fit to public-opinion survey data.

6.1 Quick Summary of Recommendations

1. Simulate three or more chains in parallel. We typically obtain starting points by adding random perturbations to crude estimates based on a simpler model or approximation.
2. Check convergence by discarding the first part of the simulations—we discard the first half, although that may be overly conservative—and using within-chain analysis to monitor stationarity and between/within chains comparisons to monitor mixing.
3. Once you have reached approximate convergence, mix all the simulations from the second halves of the chains together to summarize the target distribution. For most purposes there is no longer any need to worry about autocorrelations in the chains.
4. Adaptive Markov chain Monte Carlo (MCMC)—for example, tuning the jumping distribution of a Metropolis algorithm—can often be a good idea and presents no problems for convergence if you restart after adapting. For example, if you have already run 400 iterations and have not reached approximate convergence, you can adjust your algorithm and run another 400 steps, discarding the earlier simulations. At the next step of adaptation, you can run another 400, and so forth, possibly adapting the adaptation time itself to balance the goals of rapid convergence and computational efficiency. (Newer, more sophisticated algorithms have the promise of allowing continuous adaptation and do not necessarily require discarding early iterations.)
5. If you have run your simulations for a while and they are not close to convergence, stop, look at plots of simulations from different chains, and go back and improve your algorithm, for example, by adding new kinds of jumps to get faster

mixing (see, e.g., Sections 11.8 through 11.9 of Gelman et al., 2003, for some simple approaches, or some of the chapters in this handbook for more advanced ideas for tackling harder problems). It is not generally a good idea to wait hours for convergence, and in many practical examples there is a real gain from getting an answer in ten seconds, say, rather than two minutes. Faster computation translates into the ability to fit more models and to do more real-time data analysis.

6. When all is done, compare inferences to those from simpler models or approximations. Examine discrepancies to see whether they represent programming errors, poor convergence, or actual changes in inferences as the model is expanded. (Here we are talking about using these comparisons as a way to diagnose potential problems in a simulation. Other chapters in this handbook discuss ways of combining MCMC runs from different models to perform more efficient computations, using techniques such as parallel tempering and path sampling.)

Another useful debugging technique is the fake-data check: Choose or simulate some "true values" of the parameters and simulate data given these parameters. Then run the MCMC algorithm and check that it converges to a distribution consistent with the assumed true parameter values.*

To illustrate the concepts in this chapter, we introduce a model fit using MCMC that comes from a political science application: modeling state-level attitudes on the death penalty over time using national survey data (Shirley and Gelman, 2010). The model is a multilevel logistic regression for the binary response representing support ($y = 1$) or opposition ($y = 0$) to the death penalty for people convicted of murder (this is how the question was phrased in repeated polls given by Gallup and the National Opinion Research Center during the time span 1953–2006). The predictors in the model include demographics such as race, sex, and age, as well as the state of residence of the respondent (nested within one of four regions of the United States), so that we can model opinion trends in different parts of the country.

6.2 Key Differences between Point Estimation and MCMC Inference

Markov chain Monte Carlo methods are wonderfully convenient and flexible but, compared to simpler methods of statistical computation, they involve two difficulties: running the Markov chains long enough for convergence, and having enough simulation draws for suitably accurate inference.

- The distribution of simulations depends on starting values of the algorithm. The user must correct for starting-value bias or else run simulations long enough that starting values have essentially been forgotten.
- Inferences are based on simulations rather than deterministic estimates; as a result the user must account for Monte Carlo error or else average over enough simulation draws that such error is negligible.

* The basic idea is that, over many simulations, 50% of the 50% posterior intervals should contain the true value, 95% of the 95% intervals should contain the true value, and so forth. Cook et al. (2006) provide a more formal procedure along these lines.

The first item above is sometimes called the problem of monitoring convergence of the sampler and is commonly assessed in two ways: by studying time trends *within* chains (thus detecting movement away from the starting points) and by examining mixing *between* chains (thus detecting influence of the starting values of the different chains).

The second item above arises because, even if all chains were started from random draws from the target distribution, we would still need to think about their speed of mixing: the iterative simulation must cycle through the distribution enough times to give us the equivalent of many independent draws. In practice, though, once chains have been run long enough that the distribution of each of them is close to the distribution of all of them mixed together, we usually have created enough simulation draws that Monte Carlo error is not a problem. So typically we simply monitor convergence and then stop. It can also be a good idea to examine movement within chains (via trace plots or time series summaries) to catch the occasional situation when a group of chains have mixed but still have not converged to a stable distribution.

Beyond this, there can be convergence problems which are essentially undetectable from output analysis alone, for example if a target distribution has multiple, well-separated modes and all the chains are started from within a single mode. Here there may be specific workarounds for particular models, but in general the only solution is the usual combination of subject-matter understanding, comparisons to previous fitted models, and mathematical analysis: the usual set of tools we use in any data analysis. In the words of Brooks et al. (2003):

> Diagnostics can only reliably be used to determine a lack of convergence and not detect convergence *per se*. For example, it is relatively easy for a sampler to become stuck in a local mode and naively applied diagnostics would not detect that the chain had not explored the majority of the model/parameter space. Therefore, it is important to use a range of techniques, preferably assessing different aspects of the chains and each based upon independent chains started at a range of different starting points. If only a single diagnostic is used and it detects no lack of convergence, then this provides only mild reassurance that the sampler has performed well. However, if a range of diagnostics can be used and each detects no lack of convergence, then we can be far more confident that we would gain reliable inference from the sampler output.

Ultimately, MCMC computation, and simulation in general, is part of a larger statistical enterprise.

In the case of our example, we aim to summarize patterns and trends in public opinion on the death penalty for political scientists by fitting a model to survey data. To see how knowledge of the problem leads to better decision-making regarding inferences via simulation, consider the situation in which we encounter multiple modes in the target distribution of some parameter, such as the time trend for the coefficient of a particular state. Given that we are modeling survey data, we might hypothesize that the multiple modes represent a mixture of distributions that correspond to different subgroups of the population in that state, and we would then want to add an interaction term in the model between state of residence and some demographic variable, such as sex, to see if the multimodality disappears. Such situations highlight that the convergence of MCMC algorithms depends strongly on whether the model actually fits the data: these are never totally separate, and convergence problems are often related to modeling issues.

6.3 Inference for Functions of the Parameters vs. Inference for Functions of the Target Distribution

It is sometimes said that simulation-based inference is all about the problem of estimating expectations $E(\theta)$ under the target distribution, $p(\theta)$.* This is not correct. There are actually *two* sorts of inferential or computational task:

Task 1. Inference about θ or, more generally, about any quantity of interest $g(\theta)$. Such inference will typically be constructed using a collection of 1000 (say) simulations of the parameter vector, perhaps summarized by a mean and standard deviation, or maybe a 95% interval using the empirical distribution of the simulations that have been saved. Even if these summaries could be computed analytically, we would in general still want simulations because these allow us directly to obtain inferences for any posterior or predictive summary.

Task 2. Computation of $E(\theta)$ or, more generally, any function of the target distribution. For example, suppose we are interested in a parameter θ and we plan to summarize our inference using a posterior mean and standard deviation. Then what we really want are $E(\theta)$ and $E(\theta^2)$, which indeed are expectations of functions of θ. Or suppose we plan to summarize our inference using a 95% central posterior interval. These can be derived from posterior expectations; for example, the lower endpoint of the interval is the value L for which $\Pr(\theta < L) = 0.025$.

The precision we need depends on our inferential goals. Consider a scalar parameter θ whose posterior distribution happens to be approximately normal with mean and standard deviation estimated at 3.47 and 1.83, respectively. Suppose you are now told that the Monte Carlo standard deviation of the mean is estimated to be 0.1. If your goal is inference for θ—Task 1 above—you can stop right there: the Monte Carlo error is trivial compared to the inherent uncertainty about θ in your posterior distribution, and further simulation will be a waste of time (at least for the purposes of estimating 50% and 95% intervals for θ).† However, if your goal is to compute $E(\theta)$—Task 2—then you might want to go further: depending on your ultimate goal, you might want to learn that $E(\theta)$ is actually 3.53 or 3.53840 or whatever.

Task 1 is by far the more common goal in Bayesian statistics, but Task 2 arises in other application areas such as statistical physics and, in statistics, the computation of normalizing constants and marginal distributions. Much of the routine use of Markov chain simulation (e.g. inferences for hierarchical models using the Bayesian software package BUGS) culminates in inferences for parameters and model predictions (i.e. Task 1). Many of the most technically demanding simulation problems have Task 2 as a goal.

---

* In Bayesian applications, the target distribution is the posterior distribution, $p(\theta \mid y)$, but more generally it can be any probability distribution. Our discussion of inference and convergence does not require that the MCMC be done for a Bayesian purpose, so we simply write the target distribution as $p(\theta)$, with the understanding that it might be conditional on data.

† In this example, the standard deviation is only estimated, not known, but our point remains. If the standard deviation is estimated at 1.83, it is highly doubtful that adding further precision to the $E(\theta)$ will tell us anything useful about θ itself. If computation is free, it is fine to run longer, but to the extent that computation time is an issue and some stopping criterion must be used, it makes sense to tie the convergence to the estimated uncertainty in θ rather than to keep going to get some arbitrary preset level of precision.

It may be that much of the confusion of the statistical literature on MCMC convergence involves methods being designed for Task 1 problems and applied to Task 2, and vice versa.

One goal of our death penalty analysis is to measure the changes in attitudes during the past half-century in each of the four regions of the United States (Northeast, South, Midwest, and West). We model changes as being linear on the logistic scale, estimating a different slope parameter for each region, β^{North}, β^{South}, β^{Midwest}, and β^{West}, and also estimating the standard deviation among regions, σ_{region}. We will take the posterior distributions of these five parameters as our target distribution of interest, and our inferential goals are of the Task 1 variety. That is, we care about basic summaries of the distributions of these parameters, and not functions of them, such as their means.

6.4 Inference from Noniterative Simulations

We first consider the simple problem of inference based on simulations taken directly from the target distribution. Let us consider specific instances of the two tasks mentioned above:

1. Inference about a parameter (or function of parameters) θ, to be represented by a set of simulations and possibly a 95% interval. We can order our simulation draws and use the 2.5% and 97.5% quantiles of these simulations.

As pointed out by Raftery and Lewis (1992), these extreme order statistics are numerically unstable. For example, Table 6.1 shows five replications of inferences from a unit normal distribution based on 100 simulations, then based on 1000 simulations. If the goal is to precisely determine the endpoints of the interval (e.g. to determine if a coefficient is statistically significant, or simply to present a replicable value for publication), then many simulations are required—even in this extremely easy problem, 1000 independent draws are not enough to pin down the interval endpoints to one decimal place. However, if the goal is to get an interval for θ with approximate 95% coverage in the target distribution, even 100 draws are reasonable.

TABLE 6.1

Simple Examples of Inference from Direct Simulation

Inferences Based on 100 Random Draws	Inferences Based on 1000 Random Draws
[–1.79, 1.69]	[–1.83, 1.97]
[–1.80, 1.85]	[–2.01, 2.04]
[–1.64, 2.15]	[–2.10, 2.13]
[–2.08, 2.38]	[–1.97, 1.95]
[–1.68, 2.10]	[–2.10, 1.97]

Simple examples of inference from direct simulation. Left column: five replications of 95% intervals for a hypothetical parameter θ that has a unit normal distribution, each based on 100 independent simulation draws. Right column: five replications of the same inference, each based on 1000 draws. For either column, the correct answer is $[-1.96, 1.96]$. From one perspective, these estimates are pretty bad: even with 1000 simulations, either bound can easily be off by more than 0.1, and the entire interval width can easily be off by 10%. On the other hand, for the goal of inference about θ, even the far-off estimates above aren't so bad: the interval $[-2.08, 2.38]$ has 97% probability coverage, and $[-1.79, 1.69]$ has 92% coverage.

Our practical advice is to use the estimated uncertainty in the target distribution to decide when simulations are sufficient; the purpose of this particular simple example is to demonstrate that an appropriate minimal number of simulations depends on inferential goals.

Many fewer draws are needed for everyday inference than for a final published result. And we see this even with direct simulations without even getting into the "Markov chain" part of MCMC.

2. Inference about the mean, $E(\theta)$. We can divide our simulations of θ into k groups, compute the sample mean for each group, and then get a standard error for the grand mean by taking the standard deviation of the k group means, divided by $\sqrt{k}$.

Dividing into groups is an arbitrary choice—presumably it would be better to use a tool such as the jackknife (Efron and Tibshirani, 1993)—but we go with a simple batch means approach here because it generalizes so naturally to MCMC with blocking and parallel chains. In any case, the appropriate number of simulation draws will depend on the inferential goal. For example, 1000 random draws from a unit normal distribution allow its mean to be estimated to within a standard error of approximately 0.03.

6.5 Burn-In

It is standard practice to discard the initial iterations of iterative simulation as they are too strongly influenced by starting values and do not provide good information about the target distribution. We follow this "burn-in" idea ourselves and generally discard the first half of simulated sequences. Thus, if we run MCMC for 100 iterations, we keep only the iterations 51–100 of each chain. If we then run another 100 iterations, we discard the 50 we have already kept, now keeping only iterations 101–200, and so forth.

Burn-in is convenient, but discarding early iterations certainly cannot be the most efficient approach; see Geyer (1998) for a general argument and Liu and Rubin (1996, 2002) for specific methods for output analysis accounting for the dependence of the simulations on the starting values. That said, we typically go with the simple burn-in approach, accepting the increased Monte Carlo error involved in discarding half the simulations.

There has been some confusion on this point, however. For example, we recently received the following question by email:

> I was wondering about MCMC burn-in and whether the oft-cited emphasis on this in the literature might not be a bit overstated. My thought was that the chain is Markovian. In a Metropolis (or Metropolis–Hastings) context, once you establish the scale of the proposal distribution(s), successful burn-in gets you only a starting location inside the posterior—nothing else is remembered, by definition! However, there is nothing really special about this particular starting point; it would have been just as valid had it been your initial guess and the burn-in would then have been superfluous. Moreover, the sampling phase will eventually reach the far outskirts of the posterior, often a lot more extreme than the sampling starting location, yet it will still (collectively) describe the posterior correctly. This implies that *any* valid starting point is just as good as any other, burn-in or no burn-in.
>
> The only circumstance that I can think of in which a burn-in would be essential is in the case in which prior support regions for the parameters are not all jointly valid (inside

the joint posterior), if that is even possible given the min/max limits set for the priors. Am I missing something?

Indeed, our correspondent was missing the point that any inference from a finite number of simulations is an approximation, and the starting point can affect the quality of the approximation. Consider an extreme example in which your target distribution is normal with mean μ and standard deviation σ; and your sampler takes independent draws directly from the target distribution; but you pick a starting value of X. The average of n simulations will then have the value, in expectation, of $(1/n)X + ((n-1)/n)\mu$, instead of the correct value of μ. If, for example, $X = 100$, $n = 100$, and $\mu = 1$, you are in trouble! But a burn-in of 1 will solve all your problems in this example. True, if you draw a few million simulations, the initial value will be forgotten, but why run a few million simulations if you do not have to? That will just take time away from your more important work.

More generally, the starting distribution will persist for a while, basically as long as it takes for your chains to converge. If your starting values persist for a time T, then these will pollute your inferences for some time of order T, by which time you can already have stopped the simulations if you had discarded some early steps. In this example, you might say that it would be fine to just start at the center of the distribution. One difficulty, though, is that you do not know where the center of the distribution is before you have done your simulations. More realistically, we start from estimates $\pm$ uncertainty as estimated from some simpler model that was easier to fit.

We illustrate with our example. Figure 6.1a contains a trace plot of β^{South}, the slope coefficient for the Southern region. We initialized three chains at values that were overdispersed relative to the estimate of this parameter from a simpler model (a linear model of the differences in the sample percentages of supporters in the South relative to the national average). The crude estimate of β^{South} from the simple model was 0.33, with a standard error of about 0.03, so we started our three chains at –0.7, 0.3, and 1.3, which are roughly centered at the crude estimate, but widely dispersed relative to the crude estimate's standard error,

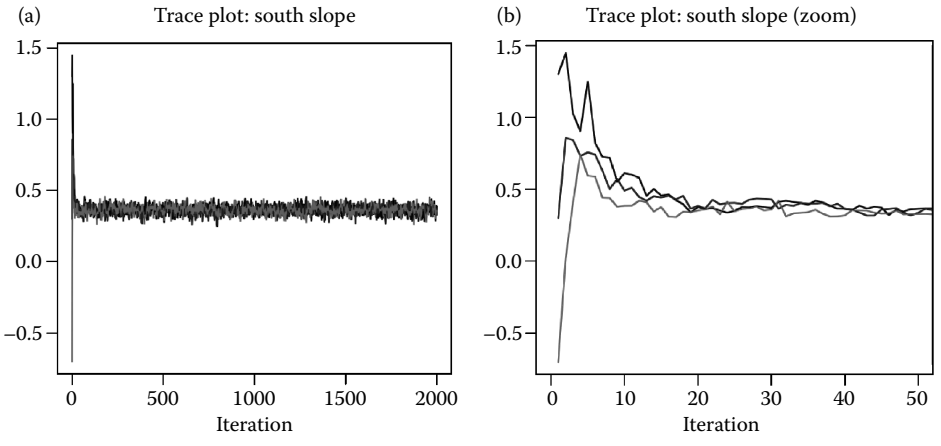

FIGURE 6.1
(a) Trace plots of β^{South}, the time trend in the logistic regression coefficients for death penalty support (per decade) for the Southern states. Three chains were initialized at –0.7, 0.3, and 1.3, respectively, and they converge to the target distribution within about 20 or 30 iterations. (b) The first 50 iterations, showing the movement away from the starting values.

so that we would be unlikely to miss a local mode in the potentially multimodal posterior distribution. The model converges very quickly, so that the initial values have been essentially forgotten after about 25 iterations (see Figure 6.1b). We ran the MCMC for 2000 iterations, because not every parameter converged as quickly as β^{South}, and also because it only took a few minutes to obtain chains of this length. Had the algorithm been much slower, we could have stopped partway through, inspected trace plots, and then made a decision about continuing the algorithm.

6.6 Monitoring Convergence Comparing between and within Chains

We never reach exact convergence; as a result it does not make sense to try to check convergence using statistical hypothesis tests of the null hypothesis of perfect mixing. Instead, we use statistical estimation—postprocessing of simulation results—to estimate how far current simulations are from perfect mixing.

We typically monitor the convergence of all the parameters and other quantities of interest separately. There have also been some methods proposed for monitoring the convergence of the entire distribution at once (see, e.g. Brooks and Gelman, 1998), but these methods may sometimes represent overkill: individual parameters can be well estimated even while approximate convergence of simulations of a multivariate distribution can take a very long time.

Our usual approach is, for each parameter or quantity of interest, to compute the variance of the simulations from each chain (after the first halves of each have been discarded, as explained in our discussion of burn-in), to average these within-chain variances, and compare this to the variances of all the chains mixed together. We take the mixture variance divided by the average within-chain variance, compute the square root of this ratio, and call it R.hat or the "potential scale reduction factor" (Gelman and Rubin, 1992, following ideas of Fosdick, 1959). R.hat is calculated in various MCMC software including BUGS (Spiegelhalter et al., 1994, 2003) and the `R2WinBUGS` and `coda` packages (Plummer et al., 2005; Sturtz et al., 2005) in R, and the underlying idea has also been applied to transdimensional simulations—mixture of models with different parameter spaces (see Brooks and Giudici, 2000; Brooks et al., 2003).

At convergence, the chains will have mixed, so that the distribution of the simulations between and within chains will be identical, and the ratio R.hat should equal 1. If R.hat is greater than 1, this implies that the chains have not fully mixed and that further simulation might increase the precision of inferences. In practice we typically go until R.hat is less than 1.1 for all parameters and quantities of interest; however, we recognize that this rule can declare convergence prematurely, which is one reason why we always recommend comparing results to estimates from simpler models. It can also be useful to check other convergence diagnostics (Cowles and Carlin, 1996). In our death penalty example, Figure 6.2a illustrates that convergence happens quickly—if we recompute R.hat every 50 iterations, discarding the first half of the iterations as burn-in in our computations, we see that it is less than 1.05 for every batch of such samples after 200 iterations and is less than 1.02 for every batch of such samples after about 500 iterations.

When problems show up, we typically look at time series plots of simulated chains to see where the poor mixing occurs and get insight into how to fix the algorithm to run more efficiently. Multivariate visual tools can make this graphical process more effective (Venna et al., 2003; Peltonen et al., 2009).

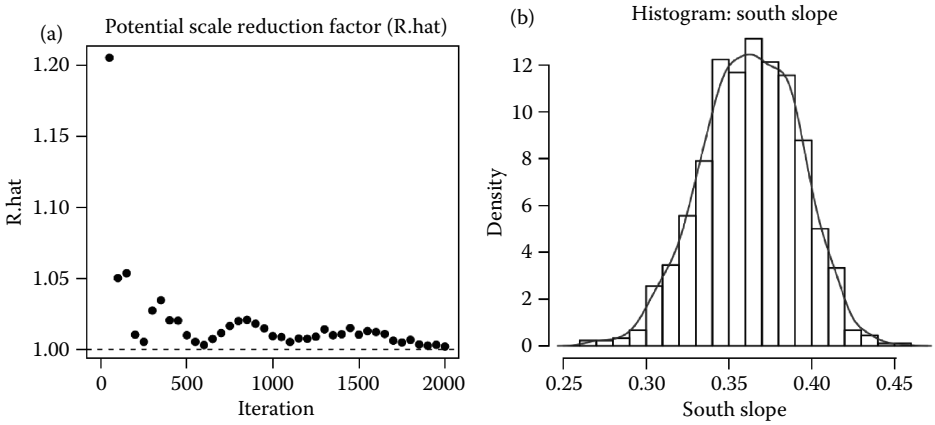

FIGURE 6.2
(a) R.hat, the potential scale reduction factor, for β^{South}, calculated repeatedly every 50 iterations using only the last half of each set of iterations. That is, it was calculated using iterations 26–50, then using iterations 51–100, then using iterations 76–150, and so on. R.hat was less than 1.05 for all batches of samples after 200 iterations and was less than 1.02 for all batches of samples after about 500 iterations. (b) The 900 posterior samples formed by taking iterations 301–600 from each of the three chains, with a density estimate overlain.

Mixing of chains can also be monitored nonparametrically, for example by computing the 80% (say) central interval from each chain and then determining its coverage with respect to the empirical distribution of all the other chains combined together (as always, after discarding the early burn-in iterations). At convergence, the average coverage of these 80% intervals should be 80%; a much lesser value indicates poor mixing. Brooks and Gelman (1998) and Brooks et al. (2003) discuss this and other methods for monitoring convergence by measuring mixing of multiple sequences, along with problems that can arise.

6.7 Inference from Simulations after Approximate Convergence

In considering inferential summaries from our simulations, we again separately consider our two tasks:

1. Inference about a parameter (or function of parameters) θ, to be summarized by a set of simulations and possibly a 95% interval.

Here we can use the collection of all our simulation draws (after discarding burn-in), or we can "thin" them by saving every nth iteration. The purpose of thinning (i.e. setting n to some integer greater than 1) is computational, not statistical. If we have a model with 2000 parameters and we are running three chains with a million iterations each, we do not want to be carrying around 6 billion numbers in our simulation. The key is to realize that, if we really needed a million iterations, they must be so highly autocorrelated that little is gained by saving them all. In practice, we find it is generally more than enough to save 1000 iterations in total, and so we thin accordingly. But ultimately this will depend on the size of the model and computational constraints.

2. Inference about an expectation, $E(\theta)$.

To estimate expectations, we can use the batch means method, dividing the chains (again, after discarding burn-in) into k batches and then computing the mean and standard error based on the average of the batch means. Each chain should be divided into batches, so that k is the number of chains multiplied by the number of batches per chain. The number of batches per chain should be set large enough that the standard error can be estimated reasonably precisely (the precision is essentially that of a chi-squared distribution with $k - 1$ degrees of freedom) while still having the batch means be approximately statistically independent (so that the $1/\sqrt{k}$ standard error formula is applicable). If necessary, the standard error can be adjusted upward to correct for any autocorrelation remaining even after batching.

In the death penalty example, our inference for β^{South}, as noted above, is of the Task 1 variety. According to Figure 6.2a, we can assume that the three chains have converged after 300 iterations, and we can use the next 300 iterations from each of the three chains to form a posterior sample of size 900 for our inference. From this we estimate the posterior mean of β^{South} to be 0.363 and the standard deviation of β^{South} to be 0.029. A 95% interval for β^{South} is obtained by taking the 2.5th and 97.5th percentiles of this sample, which are 0.31 and 0.42, respectively. Figure 6.2b contains a histogram of these 900 posterior samples, with a smooth density estimate overlain.

To estimate the uncertainty about the expectation (in this case the posterior mean, $\mathrm{E}(\beta^{\text{South}}|y)$), we can use the batch means method. We divide the 300 samples from each chain into six batches each, and compute the standard error of these $k = 18$ batch means (where each set of six batch means per chain has autocorrelation approximately zero). This approximate standard error is about 0.002. As in the toy example from earlier, the uncertainty about the mean is tiny compared to the uncertainty in the posterior distribution of the parameter β^{South}, and we conclude that these 900 samples are sufficient for our Task 1 inference for β^{South}. We can double-check our batch means calculations by computing the effective sample size of these 900 draws (which accounts for autocorrelation) using standard methods (Kass et al., 1998), and we compute that the approximate total number of independent draws from these three sets of 300 autocorrelated samples is 169. Thus, the standard error of the mean computed this way is $0.029/\sqrt{169}$, which is about 0.002, confirming our earlier batch means calculation.

In practice, we could compute more accurate estimates of summaries of the posterior distribution of β^{South}, for example the expectation, variance, and various quantiles, by including iterations 101–2000 in our posterior sample: visual inspection of the trace plot that the chains converged by iteration 100, and in fact R.hat = 1.00 when it is computed using iterations 101–2000. But more samples will not improve our Task 1 inference in any meaningful way, and since this was our goal, we could have stopped the samplers after about 600 iterations (instead of running them for 2000 iterations as we did).

We have also worked on problems that have required tens of thousands of iterations or more to reach approximate convergence, and the same inferential principles apply.

6.8 Summary

Monitoring convergence of iterative simulation is straightforward (discard the first part of the simulations and then compare the variances of quantities of interest within and between chains) and inference given approximate convergence is even simpler (just mix

the simulations together and use them as a joint distribution). Both these ideas can be and have been refined, but the basic concepts are straightforward and robust.

The hard part is knowing what to do when the simulations are slow to converge. Then it is a good idea to look at the output and put together a more efficient simulation algorithm, which sometimes can be easy enough (e.g. using redundant parameterization for the Gibbs sampler or tuning the proposal distributions for a Metropolis algorithm), sometimes can require more elaborate algorithms (such as hybrid sampling or parallel tempering), and sometimes requires development of a simulation algorithm specifically tailored to the problem at hand. Once we have an improved algorithm, we again monitor its convergence by measuring the mixing of independent chains and checking that each chain seems to have reached a stationary distribution. And then we can perform simulation-based inferences as described above.

Acknowledgments

We thank Steve Brooks, Brad Carlin, David Dunson, Hal Stern, and an anonymous reviewer for helpful comments and discussion. The National Science Foundation grants SES-1023176, ATM-0934516; National Institutes of Health; National Security Agency grant H98230-10-1-0184; Department of Energy grant DE-SC0002099; Institute for Education Sciences grants ED-GRANTS-032309-005, R305D090006-09A; and Yahoo! Research provided partial support for this research.

References

Brooks, S. and Gelman, A. 1998. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7:434–455.

Brooks, S. and Giudici, P. 2000. MCMC convergence assessment via two-way ANOVA. *Journal of Computational and Graphical Statistics*, 9:266–285.

Brooks, S., Giudici, P., and Phillipe, A. 2003. Nonparametric convergence assessment for MCMC model selection. *Journal of Computational and Graphical Statistics*, 12:1–22.

Cowles, M. K. and Carlin, B. P. 1996. Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91:883–904.

Cook, S., Gelman, A., and Rubin, D. B. 2006. Validation of software for Bayesian models using posterior quantiles. *Journal of Computational and Graphical Statistics*, 15:675–692.

Efron, B. and Tibshirani, R. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.

Fosdick, L. D. 1959. Calculation of order parameters in a binary alloy by the Monte Carlo method. *Physical Review*, 116:565–573.

Gelman, A. and Rubin, D. B. 1992. Inference from iterative simulation using multiple sequences (with discussion). *Statistical Science*, 7:457–511.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. 2003. *Bayesian Data Analysis*, 2nd edition. Chapman & Hall/CRC, Boca Raton, FL.

Geyer, C. 1998. Burn-in is unnecessary. www.stat.umn.edu/∼charlie/mcmc/burn.html

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. 1998. Markov chain Monte Carlo in practice: A roundtable discussion. *American Statistician*, 52:93–100.

Liu, C. and Rubin, D. B. 1996. Markov-normal analysis of iterative simulations before their convergence. *Journal of Econometrics*, 75:69–78.

Liu, C. and Rubin, D. B. 2002. Model-based analysis to improve the performance of iterative simulations. *Statistica Sinica*, 12:751–767.

Peltonen, J., Venna, J., and Kaski, S. 2009. Visualizations for assessing convergence and mixing of Markov chain Monte Carlo simulations. *Computational Statistics and Data Analysis*, 53:4453–4470.

Plummer, M., Best, N., Cowles, K., and Vines, K. 2005. Output analysis and diagnostics for MCMC: The coda package for R. http://pbil.univ-lyon1.fr/library/coda/html/00Index.html

Raftery, A. E. and Lewis, S. M. 1992. How many iterations in the Gibbs sampler? In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (eds), *Bayesian Statistics 4: Proceedings of the Fourth Valencia International Meeting*, pp. 765–776. Oxford University Press, Oxford.

Shirley, K. and Gelman, A. 2010. State-level trends in public opinion about the death penalty, 1953–2006. Technical report, Department of Statistics, Columbia University.

Spiegelhalter, D., Thomas, A., Best, N., Gilks, W., and Lunn, D. 1994, 2003. BUGS: Bayesian inference using Gibbs sampling. MRC Biostatistics Unit, Cambridge, UK. www.mrc-bsu.cam.ac.uk/bugs/

Sturtz, S., Ligges, U., and Gelman, A. 2005. R2WinBUGS: A package for running WinBUGS from R. *Journal of Statistical Software*, 12(3).

Venna, J., Kaski, S., and Peltonen, J. 2003. Visualizations for assessing convergence and mixing of MCMC. In N. Lavraè, D. Gamberger, H. Blockeel, and L. Todorovski (eds), *Machine Learning: ECML 2003*, Lecture Notes in Artificial Intelligence, Vol. 2837. Springer, Berlin.