Markov Chain Monte Carlo in Practice: A Roundtable Discussion

Moderator: Robert E. Kass[1]

Panelists: Bradley P. Carlin, Andrew Gelman, and Radford M. Neal

July 29, 1997

Abstract. Markov chain Monte Carlo (MCMC) methods make possible the use of flexible Bayesian models that would otherwise be computationally infeasible. In recent years, a great variety of such applications have been described in the literature. Applied statisticians who are new to these methods may have several questions and concerns, however: How much effort and expertise are needed to design and use a Markov chain sampler? How much confidence can one have in the answers that MCMC produces? How does the use of MCMC affect the rest of the model-building process? At the Joint Statistical Meetings in August, 1996, a panel of experienced MCMC users discussed these and other issues, as well as various "tricks of the trade". This paper is an edited recreation of that discussion. Its purpose is to offer advice and guidance to novice users of MCMC – and to not-so-novice users as well. Topics include building confidence in simulation results, methods for speeding and assessing convergence, estimating standard errors, identification of models for which good MCMC algorithms exist, and the current state of software development.

[1]Robert E. Kass is Professor and Chair, Department of Statistics, Carnegie Mellon University, Pittsburgh, PA 15213. Bradley P. Carlin is Associate Professor, Division of Biostatistics, School of Public Health, University of Minnesota, Minneapolis, MN 55455. Andrew Gelman is Associate Professor, Department of Statistics, Columbia University, New York, NY 10027. Radford M. Neal is Assistant Professor, Departments of Statistics and Computer Science, University of Toronto, Toronto, ON M5S 3G4. The authors thank Ming-Hui Chen for participating in a second, similar panel discussion held at the 1997 Joint Statistical Meetings, and Jim Albert for organizing both the 1996 and 1997 sessions.

Introduction

The 1990's have witnessed a burst of activity in applying Bayesian methods. Most of these applications have used *Markov chain Monte Carlo* (MCMC) methods to simulate posterior distributions. The simulation algorithm is, in its basic form, quite simple and is becoming standard in many Bayesian applications (see e.g. Gilks, Richardson, and Spiegelhalter, 1996). Furthermore, it has been around for a long time (dating at least to Metropolis et al., 1953), and the essential theory is in place (see Tierney, 1994, for a review). Nonetheless, newcomers often run into substantial difficulties. For this reason we felt it would be worthwhile to discuss some of the most pressing issues at the 1996 Joint Statistical Meetings in Chicago. Here we offer a recreation of that discussion.

Before launching into our discussion, a quick review of some terminology and standard notation will be helpful. The problem is to simulate observations from a *posterior* distribution, obtained via Bayes' Rule as

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\, d\boldsymbol{\theta}} ,$$

where $p(\mathbf{y}|\boldsymbol{\theta})$ denotes the likelihood and $p(\boldsymbol{\theta})$ the *prior* density for the vector of k model parameters $\boldsymbol{\theta}$. The practical virtue of simulation methods in general, including MCMC, is that, given a set of random draws $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(G)}$ from the posterior distribution, one can estimate virtually all summaries of interest from the posterior distribution directly from the simulations. For example, means, variances, and posterior intervals for a quantity of interest $h(\boldsymbol{\theta})$ can be estimated using the sample mean, variance, and central intervals of the values $h(\boldsymbol{\theta}^{(1)}), h(\boldsymbol{\theta}^{(2)}), \ldots, h(\boldsymbol{\theta}^{(G)})$. MCMC methods have been successful because they allow one to draw simulations from a wide range of distributions, including many that arise in statistical work, for which simulation methods were previously much more difficult to implement.

For statistical users, there are two basic methods of MCMC. The *Gibbs sampler* sequentially

samples from the collection of *full* (or *complete*) *conditional distributions* $p(\theta_i|\theta_{j\neq i}, \mathbf{y})$, $i = 1, \ldots, k$, and it does, under fairly broad conditions, produce a Markov chain with the joint posterior density $p(\boldsymbol{\theta}|\mathbf{y})$ as its stationary distribution. The algorithm was named by Geman and Geman (1984); Gelfand and Smith (1990) showed how the method could be applied to a wide variety of Bayesian inference problems. An excellent recent tutorial was given by Casella and George (1992).

The second method applies when it is difficult to simulate from the full conditionals. In this case, one may instead simulate from a different Markov chain, having some other stationary distribution, but then modify it in such a way so that a new Markov chain is constructed that has the posterior as its stationary distribution. This magic is performed by the *Metropolis-Hastings algorithm*. It samples from a prespecified *candidate* distribution for each parameter (or group of parameters), and subsequently uses an accept-reject step. A key feature is that it involves only the unnormalized posterior density $p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})$ in the sampling chain. Here the seminal references are Metropolis et al. (1953) and Hastings (1970); Chib and Greenberg (1995) offer a fine tutorial. The required selection of an appropriate candidate density makes the Metropolis-Hastings algorithm more involved than the Gibbs sampler, but it has the advantage of being more general, and is particularly helpful for sampling parameters that lack closed, easily-recognizable forms for their full conditional distributions.

A properly derived and implemented MCMC method will produce draws from the joint posterior density $p(\boldsymbol{\theta}|\mathbf{y})$ once it has converged to stationarity. As a result, a primary concern in applying such a method is determining that it has in fact essentially *converged*, i.e., that after an initial *burn-in* period (designed to remove dependence of the simulated chain on its starting location), all further samples may be safely thought of as coming from the stationary distribution. This determination is complicated by two factors. First, since what is produced at convergence is not a single value but a *random sample* of values, we must somehow distinguish between the natural variability in

the converged chain and the (typically greater) variability in the pre-convergence samples. Second, since the sampled results come from a Markov chain, they will typically be serially correlated. Early MCMC attempts often retained only every k^{th} value from the chain, where k is an estimate of the lag at which the sample autocorrelation function "dies out," in an attempt to produce an approximately independent sample. Very high autocorrelations will lead to little movement in the full observed chain, perhaps making a sampler operating far from its stationary distribution *appear* as if it has converged. Various diagnostics (the most common of which are *trace plots* of the sampled MCMC values versus iteration) are often used to estimate the degree of *mixing* in a simulation, which is the extent to which a simulated chain traverses the entire parameter space. Analogous to the use of multiple starting points for traditional optimization algorithms, many MCMC convergence diagnostics involve the use of *multiple sampling chains*, started at disparate points in the parameter space.

Panel discussion

Routine MCMC

Kass: This roundtable was organized largely because, on the one hand, MCMC methods are often easy to apply while, on the other, there remain a number of subtleties. In many situations it's either hard to get them working well or, worse yet, it may be hard to know how well they're working. Can we begin by identifying some classes of models where MCMC is easy to use – e.g., via standard software such as `BUGS` (Gilks et al., 1994) – and is very likely to give reliable answers?

Gelman: Hierarchical linear regression models and GLM's work pretty well. Of course, nonhierarchical models are even easier, but if the data make it possible to fit hierarchical models, I'll almost always do so (see Carlin and Louis, 1996, and Gelman et al., 1995, for much more elaboration on

this point). Mixture models (including Student t's) are tougher because the posterior distributions typically have multiple modes.

Carlin: Certainly the simpler the model is, the better. I like the approach taken by the `BUGS` people to offer not just a manual (Spiegelhalter et al., 1995a), but also a book of examples (Spiegelhalter et al., 1995b) that can serve as prototypes for users. This book currently covers the usual range of GLM's, as well as a surprising number of nonstandard models and special cases of interest (spatial models for disease mapping, conditional inference in case-control studies, etc.). This approach by the way is reminiscent of that used by most `SAS` programmers I know: you don't read the manual; instead, you find the example that most nearly matches your situation, copy it, and modify it.

Neal: I'd like to inject a bit of doubt into this discussion. Even in fairly simple cases (e.g., simple random effects models), it is possible for the posterior to be multimodal, which could cause the unwary user to get the wrong answer. There may be classes of models where some MCMC method can be shown to work reliably (either theoretically or empirically), but I'm not aware of any good demonstrations of this sort. Even if a model has worked well in the past, it is possible that it will work much less well on a new dataset.

This isn't meant to be excessively discouraging – after all, it's also hard to guarantee that a maximum likelihood procedure is really finding all the modes – but some caution is almost always prudent.

Kass: It's certainly good to be reminded that problems with maximization as well as simulation can arise whenever a posterior is not log-concave, which is to say in nearly every data analysis problem to which posterior sampling methods are going to be applied. However, this isn't very satisfying to the novice user. Let me go on to my next question and return later to this more difficult issue.

Assessing convergence

Kass: Together with Kate Cowles, Brad has written a nice review of convergence diagnostics (Cowles and Carlin, 1996). But it left me thinking that knowledge about this topic is not as great as the number of papers that have been written about it might lead one to believe. Indeed, some experienced users simply examine the trace plots (for some collection of parameters) informally. So, what do each of you do to assess convergence?

Gelman: I automatically monitor $\widehat{R}$ (Gelman and Rubin, 1992) for all parameters in the model and anything else that might be of interest. For any given parameter, $\widehat{R}$ is the estimated posterior variance of the parameter, based on the mixture of all the simulated sequences, divided by the average of the variances within each sequence. Thus $\widehat{R} = 1$ means that the sequences have mixed, at least according to this criterion. I try to start the different simulations overdispersed, with the intention that the sequences will not all be stuck in some unrepresentative small region of parameter space. I don't stop until $\widehat{R}$ is at some low value (e.g., less than 1.2) for all the parameters. If I get to that point in a reasonably short time, I don't look at any simulation trails (the trace plots with several parallel simulations overlaid) at all.

Carlin: I like Andrew's idea of using a few (say, 3 or 5) initially overdispersed sampling chains, but my experience with convergence diagnostics makes me wary of relying on only one. Instead of monitoring $\widehat{R}$ for every parameter, I actually plot the sample traces of my chains for a "representative subset" of the parameter space. For a standard hierarchical model, this might include most of the fixed effects, some of the variance components, and a few well-chosen random effects – say, corresponding to two individuals who are at opposite extremes relative to the population (and who are thus likely to have random effects distributions not concentrated near zero). I decorate these plots with a few very simple convergence diagnostics – typically $\widehat{R}$, and the lag-1 sample

autocorrelation in the middle chain. A collection of pictures is slower to look at (and of course, impossible to automate), but much more reliable and can even reduce run times in situations where summary diagnostics are artificially inflated (e.g., $\widehat{R}$ will always be large for a bimodal posterior if at least one chain is visiting each mode).

Neal: I don't use any formal convergence diagnostics, other than to compute autocorrelation estimates now and then. I just look at trace plots of various quantities that seem to be of central importance, such as crucial hyperparameters, and almost always, the log of the posterior probability density of the current state. If the log posterior density is going up and up, you haven't reached the main mode yet; if it's going down and down, you started near a mode that was tall but narrow, containing little probability mass, and are working your way to a more representative part of the distribution.

Looking at a few chains is usually a good idea, as it can certainly reveal problems that you can't see by looking at only one chain. If convergence is very slow, however, I sometimes run just one chain, as this can often be better than running several chains for correspondingly less time. I wouldn't really trust results found with a single chain, though.

If you start your chains at points that aren't typical of the posterior distribution, you will certainly need to discard early states, from the "burn-in" period. These atypical states will often be obvious from the trace plots. If you are using only one chain, the amount you discard should be at least as large as your estimate of the maximum lag at which any of the quantities that you are monitoring have a substantial amount of autocorrelation. With a few chains, the same rule should probably apply, even though with many chains you could in theory diagnose convergence before this. If after discarding "burn-in" states, you are left with less than half the run, you clearly haven't run for long enough. Typically, one would discard only a small fraction of the run.

Kass: What about the case in which marginals of some parameters seem to converge but others don't?

Gelman: Start by setting the parameters that are not converging to reasonable fixed values; then do the inference for the others conditional on them. I think it's necessary to do serious thinking in these situations to understand why the problem is occurring and whether it is a real statistical problem or just slow convergence.

Carlin: Yes, the usual acceleration tricks apply, such as reparametrizations, blocking (updating parameters in medium-dimensional groups), collapsing (generating from partially marginalized distributions), and using cycles or mixtures of MCMC algorithms (Tierney, 1994; Gelfand and Carlin, 1995). But careful thought is required. It's easy to create examples where the "parameters of interest" appear to have converged, but in fact have not due to slowly converging nuisance parameters. For instance, suppose Y has a Normal distribution with mean $\theta_1 + \theta_2$ and variance 1, and the parameter of interest is $\mu = \theta_1 + \theta_2$. If we put proper priors on θ_1 and θ_2, these parameters become identified, but if these priors have large variances the parameters are "just barely" identified. (See the exercise on p.203 of Carlin and Louis, 1996.) The resulting slow convergence for θ_1 and θ_2 causes a correspondingly slow convergence for μ, but the problem is apparent *only* from the θ output; no plot or diagnostic for μ suggests any convergence failure. So simply stopping the sampler when the parameters of interest appear to have settled down is a terrible idea.

Neal: That's a nice example. Unless you are very sure you know what's going on (and offhand I can't see how you could be), you should never use a chain to estimate the expectation of one function of the parameters if another function has clearly not converged.

How useful are transformations?

Kass: Brad mentioned transformations. How important are they and what general guidelines are there for choosing them?

Gelman: Very important. Two kinds of transformations I like are: (a) setting nonidentified or poorly identified parameters to fixed values (a special case of introducing general constraints), and (b) rotating/scaling to get approximate posterior independence.

Carlin: Even simple transformations, such as taking the log of a parameter with a heavily right-skewed marginal distribution, can be surprisingly helpful. Also, hierarchical centering (Gelfand, Sahu and Carlin, 1995, 1996), which involves centering random effects around their means in hierarchical models, can be very effective. These transformations don't even involve a Jacobian, and can lead to surprising reductions in parameter correlation, hence accelerating convergence.

Neal: Transformations can be very beneficial. In high-dimensional problems, they can also be hard to find, and possibly hard to apply even if you can find them. Doing a general rotation in order to decorrelate parameters is out of the question if you have hundreds or thousands of parameters. Finding problem-specific transformations such as Brad mentioned may be more promising than trying to apply some general technique.

Gelman: Once again, I think Radford is the skeptic here because he works with more complicated models. For the relatively simple problem of hierarchical linear regression models, Boscardin (1996) shows rotation to be effective (reducing computation time by a factor of 10 to 40) even with hundreds of parameters.

Starting values

Kass: What general guidelines are there on finding good starting values?

Gelman: I use estimates from simpler models (e.g., setting hyperparameters to fixed values), estimates using less information (e.g., discarding missing data), or simpler methods (e.g., maximum likelihood). With Gibbs sampling, you often only have to specify either the main parameters or the hyperparameters – the others automatically get updated in the first step. I try to make sure the chain is overdispersed (this is easiest when the prior distribution is informative).

Carlin: I typically use prior distributions that are vague but not arbitrarily so, and centered near a value that I feel the data could logically support. So a very simple rule for initializing 5 parallel chains for θ_i is to start chain j at $\mu_i + (j-3)\sigma_i$, $j = 1, \ldots, 5$, where μ_i and σ_i are the prior mean and standard deviation of θ_i. Note that this doesn't really create much "overdispersion," since the 5 starting values lie on a single ray in the full parameter space, rather than being randomly distributed throughout it. But it seems to work well in most practical settings.

Kass: When you say "work well" you mean that you are sometimes able to detect multiple modes?

Carlin: Yes, though in the posterior surfaces I see, ridges seem to be more common than multiple modes. Even more commonly, this crude initialization procedure allows me to detect convergence failure.

Gelman: Let me echo Brad here. I've encountered slow convergence in distributions that are essentially unimodal but are full of ridges in high dimensions. I've also found obvious programming and modeling bugs using multiple starting points (see, e.g., p. 134 of Gelman, 1996).

Neal: The problems I usually work with (e.g., neural network models, mixture models) may be a bit different from those that Andrew and Brad work with. The maximum likelihood estimates for

these models are often ridiculous. For example, maximum likelihood for a mixture of normals will place components having zero variance right on the data points. Accordingly, I often just start all the chains in the same place, perhaps the prior mode. I also often use a different Markov chain for the first few iterations than I use later; in this chain, I sometimes fix some of the hyperparameters, in order to prevent them from taking on strange values in the period before the other parameters have adopted reasonable values.

Poor behavior of the chain

Kass: What do you usually do when you have difficulties with convergence?

Gelman: Here are four problems that can lead to MCMC convergence difficulties, listed in increasing order of difficulty of diagnosis and repair:

1. *Problem:* poor convergence (slow, too slow, or really too slow). *Method of checking:* multiple starting points, check $\widehat{R}$, examine sequences if $\widehat{R}$ remains far from 1. *Potential solutions:* (a) if slow, run longer; (b) if too slow, tune algorithm (e.g., reparameterize Gibbs, alter scales of Metropolis jumps, etc.); (c) if really too slow, alter algorithm in a serious way, e.g., by putting in jumps between modes, using auxiliary variables (Swendsen and Wang, 1987; Besag and Green, 1993), simulated tempering (Geyer and Thompson, 1995; Neal, 1996b), and so on.

2. *Problem:* mistake in implementation of MCMC or coding the model. *Methods of checking:* (a) rerun with simulated data; (b) work up from a simpler model; (c) try to fit a smaller, or better-behaved, data set. *Potential solution:* debug the code!

3. *Problem:* nonidentified/underidentified model (includes improper posterior distribution) or poorly-understood model (can occur when allowing parameters that were previously fixed to

be estimated from data). *Methods of checking:* (a) rerun with simulated data (altering prior distribution to be proper, if necessary); (b) fitting the model with one or two parameters (those believed to be nonidentified) held fixed at reasonable values. *Potential solutions:* (a) sensitivity analysis (perform inference conditional on the poorly identified/understood parameters rather than averaging over them, as in full Bayes); (b) more realistic joint prior distribution on all the parameters in the model. (Adding an informative prior distribution can make sense statistically and also make the computation easier, as Brad mentioned earlier.)

4. *Problem:* poor fit of model to data. *Methods of checking:* posterior predictive checks, cross-validation, and so on. *Potential solution:* fit a different model!

It is not always possible to distinguish among these problems at first. For example, suppose you run the simulations, and $\widehat{R}$ is stuck at a high value (i.e., your Markov chain sequences are not mixing). This could be caused by any of the problems above. So it is generally necessary to do all these sorts of checks to have confidence in the model and simulation results.

An analogous situation arises in traditional statistical analysis via maximum likelihood. For example, in regression analysis, the four problems mentioned above translate to (1) numerical instability, (2) mistake in implementation, (3) ill-posed problem (multicollinearity), and (4) poor fit (perhaps because there are important nonlinear terms or interactions not included in the model). Nowadays, diagnosing (1) and (3) are automatic with a good computer program, and regression software is standard enough that (2) almost never occurs. Thus the user can concentrate on (4), for which we have standard tools, such as residual plots. Things are more complicated with GLM's, but we're getting there. Problems (1) and (3) are rare (although there are some tricky points with, e.g., identifiability in multinomial probits); (2) is becoming much less of a problem with software such as `SAS` and the modeling notation in `S`; and lots of research has gone into automatic diagnostics

for (4), though of course a common approach nowadays seems to be to examine numerous high dimensional models, with the consequent converse problem of *overfitting* the data.

With MCMC for Bayes posterior distributions, I think we're in pretty good shape on (1), and lots of work is going on with (4), of course (see e.g. Gelman, Meng, and Stern, 1996). I think that the good MCMC programs in the future will have automatic features to increase efficiency, fake data simulation, sensitivity analysis (fixing hard-to-fit parameters rather than automatically averaging over everything), fitting simpler models or the same model with less data, posterior predictive checks, and cross-validation.

Neal: This is a good categorization of problems, and Andrew is right that you can't assume at the start that you know the source of a problem. The problems aren't even exclusive: poor convergence could be due to an implementation mistake that makes things slower, while still delivering the correct answer if you wait long enough (e.g., using a Metropolis proposal distribution that wasn't what you meant to use, but which had the symmetry required for correctness).

Andrew lists poor convergence as the easiest problem to diagnose. This is often the case, but perhaps the most worrying aspect of MCMC is that poor convergence can be present without any signs of it being evident — for instance, if a mode with substantial probability has never been visited by any of the chains. To really check for this, you need to run many chains with an initial state distribution that you somehow know is adequate to find all modes. Alternatively, you can use one of the tempering schemes that Andrew mentioned, which are the only methods currently available for getting the chain to move between modes when you don't know where the modes are located.

I do a lot of work on methods for speeding up convergence, so I might well view poor convergence as an opportunity. For users more interested in getting an actual answer, a judicious transformation

could be the solution, as could use of a different MCMC method. There are several methods that are not well known in the statistics community that can speed up convergence by huge factors in some problems, by suppressing the random walk that Gibbs sampling and simple forms of the Metropolis algorithm take; see my review (Neal, 1993), a recent technical report of mine (Neal, 1995), and for an example involving neural network models, my book (Neal, 1996a).

In addition to Andrew's suggestions for detecting implementation mistakes, I would add another: Compare with a completely different implementation, preferably a simpler one. One implementation that is often easy to get right is rejection sampling from the prior. Of course, this is hopelessly slow for real problems, but it can be adequately fast for a small data set, with a fairly narrow prior in the vicinity of the true parameter values. The idea is to check that the MCMC method gets the same answer in such an easy situation. Using standard software, one would hope that such checks are not usually necessary, though the possibility of bugs should always be kept in mind.

Kass: It sounds like you are all in agreement on this issue: the more checking you do, the better. Multiple chains, multiple models, and multiple algorithms are all valuable. Let's return to the problem of non-identifiability, or near-non-identifiability, which Brad illustrated with the $N(\theta_1 + \theta_2, 1)$ example. Although in this simple case the problem is obvious, it is often encountered in more subtle forms, correct?

Gelman: Yes, there are lots of cases where you run into non-identifiability or near-non-identifiability.

Neal: But provided the posterior is proper, this is not a problem for MCMC methods — assuming you've determined that the nonidentifiability isn't due to a bug. It might be a statistical problem, but in my opinion, concern over this is often misplaced. In mixture models, for example, there is a natural nonidentifiability involving re-labelings of the mixture components. This is harmless, since this sort of multimodality is just due to relabelling the various components; the modes all have the

same effect. You should not try to get rid of it by imposing constraints on the parameters values, as this can do serious harm to the convergence of your Markov chain.

Carlin: I agree that certain types of nonidentifiability are harmless, but correctly implementing samplers under such models is certainly more difficult, since it requires a firm understanding of which parametric functions are well-identified, and which are not. While some types of nonidentifiability will be immediately apparent, in more complicated settings (e.g., hierarchical random effects models), failures in identifiability can be very subtle, muddying convergence diagnosis.

Gelman: In the Bayesian framework, one way to look at identifiability is a parameter is not identified if its posterior distribution is the same as its prior. This just puts the burden back upon us to understand our models.

Improper or very diffuse posteriors

Kass: In a couple of places already we've touched on the possibility that the posterior is improper. Sometimes a proper prior is used, but it is so widely dispersed that the posterior *effectively* (for numerical purposes) becomes improper even though it is mathematically proper. What can be done to check for or avoid these situations?

Gelman: A basic way to check is to play around with informative prior distributions. For example, in a toxicokinetic modeling problem (Gelman, Bois, and Jiang, 1996), we had prior distributions cut off at $\pm 3\sigma$, where σ was the prior standard deviation. Then if the simulations converged to the boundary (which could be caused by slow convergence, programming error, model misfit, or just because the prior bounds were not appropriate), we'd see the problem (not a foolproof method, but it actually revealed a problem to us in this case). Then we checked things by changing the cut-off to $\pm 2\sigma$ and $\pm 4\sigma$ to see if anything changed. Seems like a lot of work but maybe much of it could

be automated.

Carlin: It's important to remember that improper posteriors are sometimes created *deliberately* to make the sampling process easier (see e.g. Besag et al., 1995, for several examples). The unidentified parameters will of course never converge, but the identifiable ones (say, contrasts in an ANOVA-type model) may be very well behaved. Of course, it's critically important that the user understand the precise nature of the unidentifiability, before he or she ignores the former sort of "convergence failure" and proceeds with the analysis. (The very recent paper by Gelfand and Sahu, 1996, contains a nice discussion of this issue.) So the beginner may be well-advised to avoid such models, always starting with one that ought to be well-identified by the data, and using some of the remedies we have mentioned if the MCMC output suggests it is not.

Neal: I like to use informative proper priors, since then I can be sure that the posterior is proper, though I might sometimes use an improper prior if I had a proof that the posterior will nevertheless be proper. I can't see offhand why one would want to live dangerously by sampling from an improper posterior, nor would I be comfortable if I wasn't sure whether or not the posterior was proper.

Gelman: I agree that you don't want to be simulating from a posterior distribution when you don't know if it's proper. But it can be okay to have an improper posterior distribution for an auxiliary parameter that has been added solely for computational purposes. For example, consider the mixed effects model, $y \sim \text{glm}(X\beta + W\alpha)$, $\alpha \sim N(0, \tau^2 I)$. This can be rewritten as $y \sim \text{glm}(X\beta + \theta W\alpha)$, $\alpha \sim N(0, \phi^2 I)$, where $\tau = \theta\phi$. Using the new parameterization, Gibbs can go a lot faster (see the discussions of Meng and Van Dyk, 1997), even if both θ and ϕ have improper posterior distributions.

Checking results

Kass: How can you check results? I presume you think it wise to compare with maximum likelihood when possible, for example.

Gelman: Yes. Compare with anything and everything. Of course, if they disagree, you then have to decide whether it's worth putting in the effort to understand why. Usually it is. Different computational methods are complementary, not competitive.

Carlin: I agree completely. Applied Bayesians love the maximum likelihood estimate; after all, it's often nothing but the posterior mode under a flat prior! Also, we should keep in mind that for low-dimensional problems, alternative methods often serve as good checks and may be more efficient. Traditional quadrature methods (even newer adaptive ones) have been almost forgotten in the recent rush to MCMC; Evans and Swartz (1995) provide a nice recent summary focusing on such methods.

Kass: Along these lines let me mention my colleague Alan Genz's collection of Fortran routines for numerical integration and importance sampling, which he calls `BAYESPACK`. Based on methods described in Genz and Kass (1997), this collection is available from the website `http://www.math.wsu.edu/math/faculty/genz/homepage`.

Software

Kass: I want to turn to software for MCMC, but as a lead-in let me try to rephrase the question with which we began this discussion. In standard situations, such as generalized linear models or generalized mixed models, if all appears to have gone well with the simulation, that is, if the Markov chain appears to have converged satisfactorily, how much more do you worry?

Carlin: Assuming the results appear reasonable, not much more. Most of the MCMC convergence "horror stories" I've seen have either been the result of authors failing to check their answers (e.g., by running even one more sampling chain), or applying the technology to models certain to cause trouble (e.g., attempting to sample from the so-called "witch's hat" distribution, a mixture of a bivariate uniform with an extremely highly peaked bivariate normal). Most of the models encountered in statistical practice are far less pathological, so if you've checked things out and they look good, they probably are.

Gelman: I agree.

Kass: Fine. Then on software, certainly the most useful package so far for MCMC is `BUGS`. What sorts of things should a `BUGS` user keep in mind?

Carlin: I think I'm the only one here with much `BUGS` experience, so I'll try this one. I'm a big fan of `BUGS`; indeed the U.S. mirror for the program's (U.K.) web site at the Medical Research Council Biostatistics Unit at Cambridge University is on my machine at Minnesota (`http://www.biostat.umn.edu/mirror/methodology/bugs/`). Initially, I think some people dismissed the program as just another piece of freeware, since early versions of the program could handle mostly just "toy" problems, and were fairly buggy (making the program's name a double entendre!). However, the latest release (Version 0.5) is general and reliable enough to be used as a tool for both teaching and research; more and more scholarly papers (including some of my own) list `BUGS` as their computational engine. Its `S`-like syntax is readily accessible to statisticians, and there is certainly nothing else like it on the market, commercial or otherwise.

When running the program, one thing to keep in mind is that its error messages occasionally refer to the line in the code where it first *noticed* the problem, not where the error actually occurred, so debugging `BUGS` code can be a challenge. Parallel chains are also still a bit awkward in `BUGS`,

but I think the user should resist the temptation to give up and just run a single chain. While `BUGS` itself contains only crude convergence diagnosis abilities, the accompanying post-sampling menu-driven `S-plus` function for this purpose, `CODA` (a musical analogy, and also an anagram of *Convergence Diagnosis and Output Analysis*), provides a wealth of diagnostic and summarization tools that are fully equipped to handle parallel chains (Best, Cowles and Vines, 1995). Finally, `BUGS` at present features no Metropolis-Hastings updating capability, so all full conditional distributions that are not log-concave must be discretized onto a grid and sampled by brute force, a somewhat inelegant (though often adequate) solution.

Gelman: Although I'm also a fan of the `BUGS` project, whenever my collaborators and I have tried to use it for our applied problems, it's always turned out to be easier to write our own programs than to get `BUGS` to work — but I expect that'll change in a few years.

Kass: The `BUGS` writers emphasize directed acyclic graphs (DAGs) in formulating models. Do you find yourself using DAGs in your MCMC work?

Gelman: No.

Carlin: No, I still do it the old-fashioned way, using stacks of algebraic symbols like $Y_i|\theta_i \sim N(\theta_i, \sigma^2)$. But I can certainly see why the DAG approach might be more attractive to a statistician working with a subject-matter specialist, who may have some idea about causation in the system but no stomach for algebra. Of course, this is all modulo the usual worries about causation and correlation not being the same thing; in the specific context of using DAGs with the `BUGS` language, see the last paragraph of Fienberg (1996).

Neal: Yes, I like DAGs. What else would you scribble on a blackboard?

Kass: How do you make sure you've gotten the formulation set up correctly?

Gelman: By expanding from previous models that I understand: starting simple (e.g., setting hyperparameters to reasonable values), getting that to work, then making the model more complex and realistic.

Carlin: And by checking your code carefully at every step!

Standard errors

Kass: As Brad mentioned, one nice thing about importance sampling is that the observations are uncorrelated, so that simulation standard errors are easy to compute. There are a variety of ways to compute standard errors with MCMC. What method do you use?

Gelman: For Bayesian inference, I never assess MCMC standard errors because I am interested in inference about parameters and predictions, not functionals. For instance, suppose my 95% posterior inference for a parameter is [2.4,3.9], with $\widehat{R} = 1.05$. Then if the simulation were run forever, I would expect the interval to shrink by as much as a factor of about 5%. There is no reason to compute MCMC standard errors here.

Carlin: But there is still Monte Carlo error associated with your interval endpoints – how shall we measure it, given that the samples are correlated? I think MCMC standard errors are relevant no matter what posterior feature you're interested in. For example, writing the post-convergence samples from a single sampling chain for the i^{th} parameter as $\theta_i^{(g)}$, we could certainly estimate the posterior mean as $\hat{E}(\theta_i|\mathbf{y}) = \bar{\theta}_i = \frac{1}{G}\sum_{g=1}^{G}\theta_i^{(g)}$, the sample mean of all our MCMC samples. But we can't simply use the sample variance, $s_i^2 = \frac{1}{G-1}\sum_{g=1}^{G}(\theta_i^{(g)} - \bar{\theta}_i)^2$, divided by G as our estimate of the MCMC standard error of $\bar{\theta}_i$, since it would very likely be an underestimate due to positive autocorrelation in the samples (though this problem could be ameliorated by combining the draws from a collection of initially overdispersed sampling chains, as discussed earlier). Relatedly, the

sample variance s_i^2 will be a slightly negatively biased estimate of the posterior variance of θ_i, though it should be a reasonably good estimate whenever the sample mean is a good estimate of the posterior mean.

An early remedy suggested keeping only every k^{th} sample to achieve approximate independence, but MacEachern and Berliner (1994) proved the intuitive result that this wasteful approach is always suboptimal. A simple approach that seems to produce suitably conservative (larger) standard error estimates is *batching*, described for example in Section 6.2 of Ripley (1987), or in a more specifically Gibbs context on pp.194–5 of my book. Time series methods, as described for example by Ripley (1987, Sec 6.3) or Geyer (1992, Sec 3.1), seem a bit more complicated to implement but may well produce superior answers given the proper tuning.

Neal: I usually compute standard errors for an estimate of a posterior expectation using estimates for the autocorrelations. Partly, this is because I often want to look at the autocorrelations anyway, just to see what's going on. The idea (see e.g. Ripley, 1987; Neal, 1993) is to base the standard error on an "effective sample size", found by dividing the number of points used from the chain (G in Brad's comment) by the autocorrelation time, τ, which is defined to be $1 + 2\sum_{k=1}^{\infty} \rho(k)$, where $\rho(k)$ is the autocorrelation at lag k for the parameter of interest, θ_i. The standard error for $\bar{\theta}_i$ is then $\sqrt{s_i^2/(G/\tau)}$. Of course we have only estimates for the autocorrelations, and hence only an estimate for τ. It is necessary when estimating τ to cut off the sum at a value for k where the autocorrelations seem to have fallen to near zero, as including estimates for lots of higher lags adds too much noise. Choosing this cut-off may seem a bit subjective, but batching methods have similar fudge parameters.

A remark on a comment by Brad: $s_i^2 = \frac{1}{G-1}\sum_{g=1}^{G}(\theta_i^{(g)} - \bar{\theta}_i)^2$ is actually a consistent estimator for the posterior variance of θ_i, even in the presence of correlations, though it would be better to

divide by $G-\tau$ rather than $G-1$.

Gelman: There's an interesting distinction here between *researchers* and *users* of computational methods. All four of us are researchers in statistical computation, but then when we work on our applied problems, we become mere "users." As a researcher, I, like Radford, am interested in autocorrelations and anything else that will help me understand the multivariate Markov chain that I'm working with. But as a user, it's enough for me to know that the sequences are mixed and I don't need to run any more simulations. As a user, I'm wary of involved data-analytic methods of estimating the autocorrelation time of simulations—if approximate convergence has been reached, the properties of the simulation process seem irrelevant.

What I'm reacting to is the all-too-common practice of an MCMC user spending lots of time and ingenuity in studying autocorrelations and so forth, but then spending no time actually checking the fit of the model to the data.

Kass: So with all this technology you don't want people to forget about doing statistics. Perhaps that's an appropriate thought on which to end this discussion.

References

Besag, J. and Green, P.J. (1993), "Spatial Statistics and Bayesian Computation" (with discussion), *J. Roy. Statist. Soc., Ser. B*, **55**, 25–37.

Besag, J., Green, P., Higdon, D. and Mengersen, K. (1995), "Bayesian Computation and Stochastic Systems" (with discussion), *Statistical Science*, **10**, 3–66.

Best, N.G., Cowles, M.K. and Vines, K. (1995), "CODA: Convergence Diagnosis and Output Analysis Software for Gibbs Sampling Output, Version 0.30," Technical report, Medical Research

Council Biostatistics Unit, Institute of Public Health, Cambridge University.

Boscardin, W.J. (1996), "Bayesian Analysis for some Hierarchical Linear Models," unpublished Ph.D. thesis, Department of Statistics, University of California – Berkeley.

Carlin, B.P. and Louis, T.A. (1996), *Bayes and Empirical Bayes Methods for Data Analysis*, London: Chapman and Hall.

Casella, G., and George, E. (1992), "Explaining the Gibbs Sampler," *The American Statistician*, **46**, 167–174.

Chib, S. and Greenberg, E. (1995), "Understanding the Metropolis-Hastings Algorithm," *The American Statistician*, **49**, 327–335.

Cowles, M.K. and Carlin, B.P. (1996), "Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review," *J. Amer. Statist. Assoc.*, **91**, 883–904.

Evans, M. and Swartz, T. (1995), "Methods for Approximating Integrals in Statistics with Special Emphasis on Bayesian Integration Problems," *Statistical Science*, **10**, 254–272 (discussion and rejoinder: **11**, 54–64).

Fienberg, S.E. (1996), Comment on "Computation on Bayesian Graphical Models," by D.J. Spiegelhalter, A. Thomas, and N.G. Best, in *Bayesian Statistics 5*, eds. J.M. Bernardo, J.O. Berger, A.P. Dawid and A.F.M. Smith, Oxford: Oxford University Press, p.422.

Gelfand, A.E. and Carlin, B.P. (1995), Comment on "Bayesian Computation and Stochastic Systems," by J. Besag, P. Green, D. Higdon, and K. Mengersen, *Statistical Science*, **10**, 43–46.

Gelfand, A.E. and Sahu, S.K. (1996), "Identifiability, Propriety, and Parametrization with regard to Simulation-Based Fitting of Generalized Linear Mixed Models," Technical Report 96–36,

Department of Statistics, University of Connecticut.

Gelfand, A.E., Sahu, S.K. and Carlin, B.P. (1995), "Efficient Parametrizations for Normal Linear Mixed Models," *Biometrika*, **82**, 479–488.

Gelfand, A.E., Sahu, S.K. and Carlin, B.P. (1996), "Efficient Parametrizations for Generalized Linear Mixed Models" (with discussion), in *Bayesian Statistics 5*, eds. J.M. Bernardo, J.O. Berger, A.P. Dawid, and A.F.M. Smith. Oxford: Oxford University Press, pp. 165–180.

Gelfand, A.E. and Smith, A.F.M. (1990), "Sampling-Based Approaches to Calculating Marginal Densities," *J. Amer. Statist. Assoc.*, **85**, 398–409.

Gelman, A. (1996), "Inference and Monitoring Convergence," in *Markov Chain Monte Carlo in Practice*, eds. W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, London: Chapman and Hall, pp. 131–143.

Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B. (1995), *Bayesian Data Analysis*, London: Chapman and Hall.

Gelman, A., Meng, X.-L. and Stern, H.S. (1996), "Posterior Predictive Assessment of Model Fitness via Realized Discrepancies" (with discussion), *Statistica Sinica*, *6*, 733–807.

Gelman, A., Bois, F.Y., and Jiang, J. (1996), "Physiological Pharmacokinetic Analysis using Population Modeling and Informative Prior Distributions," *Journal of the American Statistical Association* **91**, 1400–1412.

Gelman, A. and Rubin, D.B. (1992), "Inference from Iterative Simulation using Multiple Sequences" (with discussion), *Statistical Science*, **7**, 457–511.

Geman, S., and Geman, D. (1984), "Stochastic Relaxation, Gibbs Distributions and the Bayesian

Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721-741.

Genz, A. and Kass, R. (1997), "Subregion Adaptive Integration of Functions Having a Dominant Peak," *J. Comp. Graph. Statist.*, **6**, 92–111.

Geyer, C.J. (1992), "Practical Markov Chain Monte Carlo" (with discussion), *Statistical Science*, **7**, 473–511.

Geyer, C.J. and Thompson, E.A. (1995), "Annealing Markov Chain Monte Carlo with Applications to Ancestral Inference," *Journal of the American Statistical Association*, **90**, 909–920.

Gilks, W.R., Richardson, S. and Spiegelhalter, D.J., eds. (1996), *Markov Chain Monte Carlo in Practice*, London: Chapman and Hall.

Gilks, W.R., Thomas, A. and Spiegelhalter, D.J. (1994), "A Language and Program for Complex Bayesian Modelling," *The Statistician*, **43**, 169–177.

Hastings, W.K. (1970), "Monte Carlo Sampling Methods using Markov Chains and Their Applications," *Biometrika*, **57**, 97–109.

MacEachern, S.N. and Berliner, L.M. (1994), "Subsampling the Gibbs Sampler," *The American Statistician*, **48**, 188–190.

Meng, X.L. and Van Dyk, D. (1997), "The EM Algorithm – An Old Folk-Song Sung to a Fast New Tune" (with discussion), to appear *Journal of the Royal Statistical Society, Ser. B.*

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E. (1953), "Equations of State Calculations by Fast Computing Machines," *J. Chemical Physics*, **21**, 1087–1091.

Neal, R.M. (1993), "Probabilistic Inference Using Markov Chain Monte Carlo Methods," Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto. (Available from the author's home page, at `http://www.cs.toronto.edu/~radford/`)

Neal, R.M. (1995), "Suppressing Random Walks in Markov Chain Monte Carlo Using Ordered Overrelaxation," Technical Report No. 9508, Dept. of Statistics, University of Toronto. (Available from the author's home page, at `http://www.cs.toronto.edu/~radford/`)

Neal, R.M. (1996a), *Bayesian Learning for Neural Networks*, New York: Springer-Verlag.

Neal, R.M. (1996b), "Sampling from Multimodal Distributions Using Tempered Transitions," *Statistics and Computing*, **6**, 353–366.

Ripley, B.D. (1987), *Stochastic Simulation*, New York: Wiley.

Spiegelhalter, D.J., Thomas, A., Best, N. and Gilks, W.R. (1995a), "BUGS: Bayesian Inference Using Gibbs Sampling, Version 0.50," technical report, Medical Research Council Biostatistics Unit, Institute of Public Health, Cambridge University.

Spiegelhalter, D.J., Thomas, A., Best, N. and Gilks, W.R. (1995b), "BUGS Examples, Version 0.50," technical report, Medical Research Council Biostatistics Unit, Institute of Public Health, Cambridge University.

Swendsen, R.H. and Wang, J.-S. (1987), "Nonuniversal Critical Dynamics in Monte Carlo Simulations," *Phys. Rev. Letters*, **58**, 86–88.

Tierney, L. (1994), "Markov Chains for Exploring Posterior Distributions" (with discussion), *Ann. Statist.*, **22**, 1701–1762.