Behavior Research Methods, Instruments, & Computers
1998, 30 (2), 232-237

Simulation-based Bayesian inference using BUGS

CHING-FAN SHEU and SUZANNE L. O'CURRY
DePaul University, Chicago, Illinois

We illustrate the application of BUGS, a Bayesian computer program, with two examples. The algorithm used in the program is a popular Markov chain Monte Carlo procedure called Gibbs sampling. Bayesian analysis based on simulation has been applied to a wide range of complex problems (Gilks, Richardson, & Spiegelhalter, 1996). The availability of a general purpose program like BUGS should facilitate important applications of Bayesian inference in psychological research.

More than three decades ago, Edwards, Lindman, and Savage (1963) published a ground-breaking paper on Bayesian statistical inference in psychological research. Bayesian data analysis, however, rarely appears in mainstream journals in the behavioral sciences.

One of the major limiting factors in applying Bayesian methods is computational difficulty. Given data, Bayesian inference proceeds from prior distributions to posterior distributions of the model parameters. In real applications involving many parameters, Bayesian computation requires the evaluation of complex, high-dimensional integrals. Because commonly used statistical packages cannot accommodate these difficult calculations, practitioners of Bayesian inference must rely on their ability to adapt specialized routines to the problem at hand. In our view, this lack of user-friendly software has hampered the adoption of Bayesian statistics by researchers in the social and behavioral sciences.

In recent years, the Bayesian computational problem has been addressed by taking a sampling approach. References on a number of simulation tools for drawing samples from the posterior distribution are found in Tanner (1993). So remarkably simple are these computational algorithms, they can be described in just a few lines in any computer language. With the wide availability of high-speed computing, simulation-based methods have become increasingly important in data analysis.

The purpose of this paper is to give an accessible account of the sampling-based Bayesian computation through a language called BUGS. Unlike other computer programs compiled by Press (1989), BUGS provides an integrated computing platform for Bayesian inference. Programming BUGS is rather similar to implementing statistical routines in an object-oriented language like S (Chambers & Hasties, 1992). By means of the two illustrations here, we hope to facilitate important applications of Bayesian data analysis in psychological research.

This paper is organized as follows. We describe the BUGS software in Section 2. We use a simple example to explain in detail the principles embodied in this software. Section 3 describes how the example is translated to BUGS language, and Section 4 demonstrates how to run the software. Section 5 provides a Bayesian analysis of a one-way random effects analysis of variance (ANOVA). This model is chosen in order to highlight the application of BUGS in solving a statistical problem that is familiar to psychologists, as well as the occasionally unsatisfactory results arising from the classical frequentist approach. We close with a brief discussion.

THE BUGS SOFTWARE

Several computer programs are available for Bayesian analysis. The public domain statistical software library (lib.stat.cmu.edu) at Carnegie-Mellon University is a useful resource.

Albert (1996) presents a collection of MINITAB macros for Bayesian computation. O'Hagan's First Bayes is another software package (www.math.nott.ac.uk/personal/aoh/). These two packages were written mainly for teaching and learning Bayesian statistics. In contrast, Spiegelhalter, Thomas, Best, and Gilks (1997) developed BUGS to solve complex statistical problems. BUGS is a quasi-acronym for Bayesian inference Using Gibbs Sampling. The program and documentation (including installation instructions and examples manual) can be freely downloaded from the World Wide Web (www.mrc-bsu.cam.ac.uk/bugs). The software runs under UNIX and Windows. The version described here is BUGS 0.6 for Windows. Conceptually, BUGS is built from four components:

1. The dependencies (more precisely, lack of dependence) between variables in a statistical problem can be formulated by a set of conditional independence assumptions. These assumptions represent the qualitative knowledge we have about the problem.

2. The numerical values of these dependencies are specified in terms of prior probabilities and conditional prob-

The authors thank Ira Bernstein, Rich Chechile, and an anonymous referee, whose comments led to an improved version of this paper. The authors are grateful to George Michel for his support of this research. Correspondence concerning this article should be addressed to C.-F. Sheu, Department of Psychology, DePaul University, 2219 North Kenmore Avenue, Chicago, IL 60614-3504 (e-mail: csheu@condor.depaul.edu).

Table 1
Conditional Probabilities for Symptoms Given Disease

	D = no	D = yes
R = no	.3	.4
R = yes	.7	.6
S = no	.5	.1
S = yes	.5	.9

Note—D, disease; R, rash; S, stiffness.

abilities. These probabilities, together with the data, specify our quantitative input to the problem.

3. A set of rules manipulates these probabilities (essentially rules of probabilities and Bayes' theorem) in a routine manner. This is the inferential engine that derives the posterior summaries for drawing conclusions.

4. The computations are done by stochastic simulation (Gibbs sampling). In complicated problems, exact solutions using analytical methods are not feasible.

We use a simple example to illustrate how these components emerge in solving an inferential problem.

The Disease Example

Suppose that a disease (D) infects 20% of a population. It has two symptoms, rash (R) and stiffness (S). It is assumed that the symptoms are independent, given the disease D. This means that, if the state of the disease is known, knowing whether or not a rash is present will not influence the chance of having stiffness (and vice versa). This conditional independence assumption is a qualitative statement concerning how we model the relationships among the relevant variables in a situation—the first component.

Suppose that a person presents the rash symptom (datum). What is the chance that the person is infected? That is, we wish to find $Pr\{D, S|R = \text{yes}\}$. The base rate of the infected population is known. Yet, how likely is it that we will observe either of the two symptoms in the presence (absence) of the disease? These conditional probabilities are given in Table 1. Typically, the quantitative input to a problem (second component) comes from the past and present observations or judgments of experts.

For a simple problem such as this, the desired solution can be found exactly by a simple manipulation of the rules of probabilities. Consider, however, the same problem with a disease having more than a dozen symptoms, each of which has several possible states. One will still have to apply the rules of probabilities and Bayes' theorem to find a solution (the third component). But it is impractical to attempt a direct calculation of the exact probabilities. In such cases, it is more efficient to approximate the solution by simulation. The idea is to draw randomly all possible configurations of the variables a sufficient number of times and to compute the frequency of the desired configuration (event). By taking a large enough sample, one can compute the desired probabilities to any accuracy required (the law of large numbers).

Gibbs Sampling

BUGS uses an iterative simulation scheme called Gibbs sampling (Gelfand & Smith, 1990; Geman & Geman, 1984). For the technique, we pick a variable from the configuration and sample from the distribution of this variable conditionally on the rest of the variables. This is done sequentially for all of the variables in the configuration. A good introductory account of this computer-intensive algorithm can be found in Casella and George (1992). The technique fits naturally to cases where our knowledge about the dependencies among variables can be represented in conditional terms. The general purpose computational algorithm (the fourth component) completes BUGS as an automated system for Bayesian analysis.

We return to the example in order to illustrate the Gibbs sampling technique. The goal is to obtain samples of the posterior distribution of $\{D, S|R\}$. We begin with this formula for joint probabilities of $\{D, S, R\}$:

$$\begin{aligned} Pr\{R, S, D\} &= Pr\{R, S|D\}Pr\{D\} \\ &= Pr\{R|D\}Pr\{S|D\}Pr\{D\}. \end{aligned} \quad (1)$$

The first step in the equation uses the multiplicative rule of probability, and the second step uses the assumption of conditional independence between symptoms, given the disease. Using Equation 1, we compute all of the joint probabilities presented in Table 2, although we will only need the two rows that are consistent with the presence of rash (R = yes).

For the example, we perform the Gibbs sampling as follows: First, we sample the value of disease D from the prior distribution of D. Second, given the current value of D, we sample the value of S from the conditional distribution of $\{R = \text{yes}, D = \text{current value}\}$. For instance, if the current value of D is no, the value of S is yes, with probability .5 = .28/.56. Third, given the current value of S, we sample the value of D from the conditional distribution of $\{R = \text{yes}, S = \text{current value}\}$. For instance, if the current value of S is yes, then the value of D is yes, with probability $0.278 \approx .108/.388$. The procedure alternates between the second and third steps until a long sequence of yes's and no's for (D, Y) pairs has been collected. Since we are concerned with the disease, given symptoms, we compute the proportion of yes's in the D sequence. If the Gibbs sequence converges, this proportion is an estimate of the true probability $Pr\{D = \text{yes}, S|R = \text{yes}\}$.

It is obvious that adjacent values of a Gibbs sequence are dependent. Casella and George (1992) explained the

Table 2
Joint Probabilities for Disease and Symptoms

	D = no	D = yes
R = no, S = no	.120	.008
R = no, S = yes	.120	.072
R = yes, S = no	.280	.012
R = yes, S = yes	.280	.108

Note—D, disease; R, rash; S, symptom.

Listing 1
BUGS Model File For Disease Example

```
model disease;
var
   d, r, s,       # d = disease, r = rash, s = stiffness
   p.d[2],        # prior of disease
   p.r[2,2],      # prob.  of rash given disease
   p.s[2,2];      # prob.  of stiffness given disease
data in "disease.dat";   # data and conditional prob.
{
   d ~ dcat(p.d[]);
   r ~ dcat(p.r[d,]);
   s ~ dcat(p.s[d,]);
}
```

convergence of the sequence to long-run distributions on the basis of the theory of Markov chains. The Markov chain Monte Carlo methods, of which Gibbs sampling is a special case, have become very popular in the applications of Bayesian statistics (Gilks, Richardson, & Spiegelhalter, 1996). These methods are not foolproof and should be used with care. This cautionary remark extends to any responsible use of BUGS. In general, the Markov chain Monte Carlo methods face two problems. First, we have to run the simulation long enough to ensure that the long-run distribution is reached before collecting samples. This is usually achieved by discarding an initial run of the samples, a procedure called *burn-in*. The second problem is that the sampler can move very slowly or even get stuck in a certain configuration. This often happens when the parameters of the distribution are highly correlated. In practice, this difficulty is monitored by running several diagnostics on convergence. Cowles and Carlin (1996) give a comparative review on this issue.

Although the Gibbs algorithm can be implemented in computer languages such as C or Fortran, to write and debug a Gibbs sampler for a statistical application requires a considerable amount of effort. The time cost can be high when one has to devise a scheme for sampling from arbitrary nonstandard densities. BUGS is designed to shield the users from the chore of implementing these computational procedures. The users can instead focus on statistical modeling. BUGS offers a set of concise expressions for specifying a large class of models, a sampler and a compiler for constructing and sampling from conditional distributions automatically, and a command shell for running a data analysis session, including built-in diagnostics.

PROGRAMMING BUGS

This section explains a BUGS program for the disease example. Two files, disease.dat and disease.bug, are created. The former contains the input of the example, whereas the content of the latter, presented in Listing 1, is the full model specification of the disease example.

BUGS syntax mimics closely the conventional model specification in statistics. The keyword model in the first line of the code indicates the name of the model. The next statement declares variables d, r, s, following the keyword var. The array p.d[2] has two elements. The variable p.r[2,2] is a (2 × 2) two-dimensional array, with row and column subscripts enclosed in the square bracket. BUGS ignores the comments preceded by the # sign. Note that the variables are separated by commas and the statements are terminated by colons. The statement data in "disease.dat" tells BUGS the search path leading to the data file. Here, it points to the same folder where BUGS application programs are kept. The main body of the model specification is enclosed within a parenthesis. The ~ sign indicates a stochastic relationship, and the following expressions

```
d ~ dcat(p.d[]);
r ~ dcat(p.r[d,]);
s ~ dcat(p.s[d,]);
```

declare the variables d, r, s to be categorical random variables. The dcat is used to sample values 1 or 2, according to the relevant entries in the probability tables. These are the prior probability and conditional probabilities (Table 1) input in the data file in a (S-Plus) list format:

```
list(r =2, p.d = c(.8, .2),
  p.r = c(.5, .5, .1, .9),
  p.s = c(.3, .7, .4, .6))
```

A list is used to collect items of different type. The c function combines objects into a vector. The variable r is given value 2, indicating that rash has been observed. Recall that 20% of the population is infected; this is entered by setting the second element of p.d to .2. The entries of p.r[,] are input row first, then column, in this order: p.r[1,1], p.r[2,1], p.r[2,1], p.r[2,2].

BUGS automatically constructs the necessary conditional distributions from the model specification. Therefore, the joint probabilities in Table 2 are not required in disease.bug or disease.dat.

RUNNING BUGS

In this section, we first describe a simulation session using BUGS in general. The disease example is then used to illustrate the concrete detail. A typical BUGS run consists of the following steps.

1. Compile the code. If an error is found during compilation, BUGS displays an error message, saves it to the file bugs.log, ends the session, and passes the control back to the system.

2. Update the simulation run in order to reach convergence. This is the preconvergence run of the sampler, to eliminate the influence of the initial values of the parameters. The number of iterations needed to achieve convergence depends on the problem under investigation. For moderate size data sets involving standard statistical models, a few thousand iterations should suffice.

3. Select parameters of interest for monitoring. This is up to the data analyst.

Listing 2
A BUGS Session for Disease Example

```
Bugs> compile("disease.bug")
Bugs> update(5000)
time for      5000   updates was   00:00:00
Bugs> monitor(d)
Bugs> update(10000)
time for      10000  updates was   00:00:01
Bugs> stats(d)
mean      sd         2.5% :    97.5% CI  median    sample
1.175E+0  3.800E-1   1.000E+0  2.000E+0  1.000E+0  10000
Bugs> diag(d)
mean   sd        mean   sd        Z         sample
1.18   1.55E-3   1.17   6.44E-4   3.20E-1   10000
Bugs> q()
```

4. Run a second updating to collect sample values. The number of iterations depends on the desired accuracy of the results. Typically, a few thousand samples are collected.

5. Summarize the posterior distributions of the parameters monitored. BUGS displays basic summary statistics for the selected parameters.

6. Check the convergence diagnostics. For a single long chain, a Z-test statistic, indicating whether an early part of a run differs substantially from the latter part, is computed. This is a rough approximation to the diagnostic measure suggested by Geweke (1992). Z values greater than 2 or smaller than -2 point to a lack of convergence.

7. Save the results of the simulation. If unspecified, BUGS saves output to four files by these default names: bugs.out, bugs.ind, bugs1.out, and bugs1.ind. The file bugs.out contains the monitored samples; bugs.ind keeps the variable names and counts for analyzing the data in bugs.out. The file bugs1.out contains the output of the stats command, with indicators given in bugs1.ind. A complete log of the session is written to bugs.log.

To begin a simulation session, click on the BUGS icon. A window appears with a prompt Bugs>. We assume the files disease.bug and disease.dat are in the BUGS folder. For the example, we use a burn-in of 5,000 iterations, monitor and collect 10,000 values of the d (disease) variable, check the summary statistics and the convergence diagnostic, and then quit BUGS. An edited BUGS window session is presented in Listing 2.

On a Pentium II 233 MHz personal computer, BUGS takes fractions of a second to compile and run this example. The output from the diag command shows a Z-value within the range $(-2,2)$, indicating that the run has probably converged. The command stats(d) yields a (posterior) mean of 1.175 for the 10,000 samples. Recalling that the categorical variable d is defined on the domain (1,2), we obtain .175 as an estimated value for $Pr\{D = \text{yes}, S|R = \text{yes}\}$ by subtracting 1 from the mean. This value may be compared to the direct solution $.176 \approx (.012+.108)/.68$, based on the entries of Table 2 (note that the odds of disease are about 1.56 times higher without a rash symptom than with a rash). We refer the reader to the BUGS manual for further details on command options.

RANDOM EFFECTS ANALYSIS OF VARIANCE

Many models of interest to psychologists can be implemented in BUGS, notably the latent class models and hierarchical linear models. For our second example, we consider a single-factor random effects ANOVA. This statistical model is widely used in psychological research. Less well known is the fact that the classical (frequentist) estimation of variance components can sometimes give unreasonable results (Box & Tiao, 1973).

We adapt a textbook example from Neter, Kutner, Nachtsheim, & Wasserman (1996). A consulting firm provides management services to many business companies nationwide. The firm employs a large number of personnel officers, who interview job applicants at its many regional offices. For a typical job interview, the personnel officer gives a rating between 0 and 100 to indicate the applicant's qualification for the job. A study is conducted in order to compare the mean ratings among all personnel officers in the firm. Five personnel officers in the firm were selected at random, and four applicants were assigned at random to each officer. The data for the study are shown in a 5×4 rectangular format below. Each row represents the ratings of a personnel officer for four job applicants.

76	65	85	74
59	75	81	67
49	63	61	46
74	71	85	89
66	84	80	79

The one-way normal random effects model is now described. Let Y_{ij} be the rating assigned to the jth applicant by the ith personnel officer ($i = 1, \ldots, 5; j = 1, \ldots, 4$). We assume Y_{ij} to be normally distributed with mean μ_i, and a within-personnel variance σ_w. The true mean μ_i for personnel officer i is also assumed to be normally distributed with mean μ_a (true mean for all personnel officers) and between-personnel variance σ_b.

In a Bayesian ANOVA, we need to specify prior distributions for the parameters μ_a, σ_w, and σ_b. In BUGS, precision (1/variance) parameters are used. So we define $\tau_b = 1/\sigma_b$ and $\tau_w = 1/\sigma_w$, respectively. For mathematical convenience, the true overall mean μ_a is given a normal prior, and the precisions are given gamma priors. These are conjugate priors, having the property that the posterior distribution follows the same parametric form as the prior distribution. Typically, we set huge values for the standard deviations of these conjugate prior distributions in order to make them *noninformative*. To complete the model, we assume independence between the observed ratings given all model parameters, between the means μ_i, given μ_a, τ_w, and τ_b, and between the parameters μ_a and τ_b themselves.

BUGS Program of the Job Example

We specify the random factor effects model in BUGS. Listing 3 presents the content of job.bug model file. The

Listing 3
BUGS Model File for Job Example

```
model job;
const
    rows = 5,          # number of personnel officers
    cols = 4;          # number of job applicants
var
    Y[rows,cols],      # rating for an applicant
    mu[rows],          # personnel officer's true mean
    sigma.w, tau.w,    # within-personnel variance, precision
    sigma.b, tau.b,    # between-personnel variance, precision
    mu.a;              # true overall mean
data Y in "job.dat";
inits in "job.in";
{
    # Model
    for (i in 1:rows) {
        for (j in 1:cols) {
            Y[i,j] ~ dnorm(mu[i], tau.w);
    }
    mu[i] ~ dnorm(mu.a, tau.b);    # random effects
}
# Prior distributions
tau.w ~ dgamma(.0001,.0001);
sigma.w ← 1/tau.w;              # variance = 1/precision
tau.b ~ dgamma (.0001,.0001);
sigma.b ← 1/tau.b;
mu.a ~ dnorm(.0,.0001);
}
```

data given earlier are input in job.dat. The job.in file contains the initial values of the model parameters. The correspondence between the syntax in Listing 3 and the model should be clear. We comment on a few new features.

The dimensions of data are constants in a program; hence, rows and cols are so declared, following the keyword const. The statement inits in "job.in" tells BUGS where to find the initial values (0 for mean, 1 for precision) of model parameters. The content of job.in is in a list format:

list(mu = c(0,0,0,0,0), tau.w = 1, mu.a = 0, tau.b = 1)

(The initial value file was not needed in the disease example. The values were set by sampling forward.) To specify each element of the two-dimension array Y[i,j] as a normal distribution with mean mu[i] and precision tau.w, we put the expression

Y[i,j] ~ dnorm(mu[i], tau.w);

inside a for loop (row index *i*) nested within another for loop (column index *j*). The expression for (i in 1:rows) begins the row count at one and ends the iteration at the (constant) value assigned to rows. Each element of mu[i] is declared a normal distribution with mean mu.a and precision tau.b in the outer for loop. The statement

tau.w ~ dgamma (.0001, .0001);

declares tau.w to be a gamma distribution with a mean of 1 and a standard deviation of 100, representing noninformativeness. The gamma distribution is a conjugate prior distribution for the inverse of the normal variance (precision). The form of the gamma density function can be found in the BUGS manual, indexed under distribution dgamma. Lastly, the sign ← specifies a deterministic relationship. In

sigma.w ← 1/tau.w;

the within variance sigma.w is assigned the inverse of precision tau.w.

Analysis of the Job Example

In the usual presentation of one-way random effects ANOVAs, the hypothesis regarding the equality of the (true) means is tested by examining σ_b, the variability of the means. For the job example, if the between-personnel variance is not different from zero, the mean ratings for all personnel officers are the same. What is less often discussed is that a confidence interval for σ_b cannot be computed exactly, because of its complicated sampling distribution. Several classical (frequentist) procedures for constructing approximate confidence intervals for σ_b have been developed. However, it is known that the estimated lower limit of the confidence interval for σ_b can become negative, which is not acceptable. For this data, Neter et al. (1996) reported two approximately 90% confidence intervals for σ_b, (30.9, 685.2) and (20.2, 536.4), respectively, on the basis of two different classical procedures. In contrast, the sampling-based Bayesian approach estimates σ_b by drawing from the posterior distribution of $\sigma_b|Y_{i,j}, \mu_i, \mu_a, \sigma_w$.

We run the job example in BUGS with a burn-in of 5,000 iterations. The between-personnel variance sigma.b is monitored. A further 2,200 iterations collects the samples. Figure 1 is a histogram of the central 90% of the

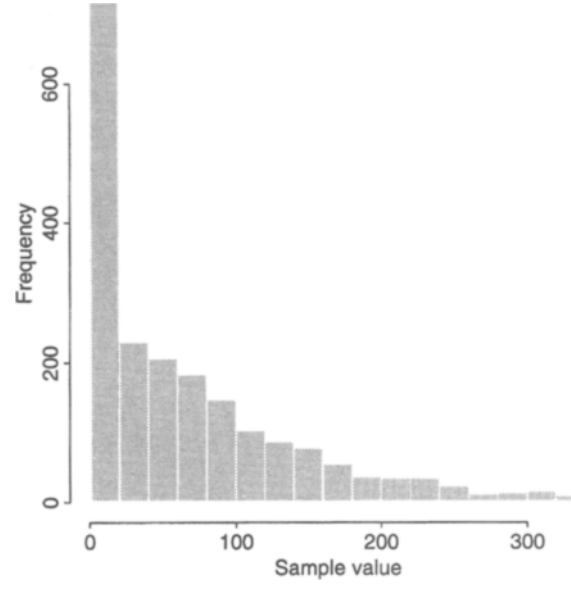

Figure 1. Histogram of the middle 90% of the 2,200 simulation draws of the between-personnel variance.

Table 3
Posterior Estimates of Between-Personnel Variance

Quantile	Estimated value
10%	0.004
25%	0.486
50%	43.990
75%	110.967
90%	219.192
95%	335.767
99%	896.294

simulated values of posterior distribution for σ_b. Omitted from this histogram are the lower and upper 5% of the simulation draws. The extreme tails are truncated in order to make the histogram visible. The values of σ_b for the middle 90% of the samples have a range of (0.000678, 335.767). This differs substantially from the estimates obtained by the classical ANOVA. The estimates of selected posterior quantiles for the entire samples are shown in Table 3.

The results of simulation indicate a great deal of uncertainty concerning σ_b. With an insufficient amount of data, there will also be a considerable sensitivity to the prior chosen in a Bayesian analysis. It reminds us that we cannot hope for an accurate estimate from such small samples. The lower limits of the two approximate confidence intervals given by the classical procedures are too big for this data. In simulation, extremely large or small values often point to a problem with the specified model that might not be noticed when the results are obtained in analytical form. We emphasize that, in addition to making Bayesian computations feasible, this is another distinct advantage of the simulation-based approach.

DISCUSSION

The Markov chain Monte Carlo methods, such as Gibbs sampling, together with computer programs like BUGS, enable researchers to get an approximately correct answer to a complex question, when analytical solutions are unavailable. This means that researchers can focus on models of empirical relevance instead of on models of convenient mathematical form.

The sampling approach cannot eliminate many objections to the use of Bayesian inference, but it has made Bayesian data analysis a realistic option for applied statisticians. We suspect that the ground-breaking work done by Edwards et al. (1963) three decades ago would have had a much greater impact on psychological research had it come bundled with computer software.

REFERENCES

ALBERT, J. H. (1996). *Bayesian computation using MINITAB*. Belmont, CA: Duxbury.

BOX, G. E. P., & TIAO, C. C. (1973). *Bayesian inference in statistical models*. Reading, MA: Addison-Wesley.

CASELLA, G., & GEORGE, E. (1992). Understanding the Gibbs sampler. *American Statistician*, **46**, 167-174.

CHAMBERS, J. M., & HASTIES, T. J. (Eds.) (1992). *Statistical models in S*. New York: Chapman & Hall.

COWLES, M. K., & CARLIN, B. P. (1996). Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of American Statistical Association*, **91**, 883-905.

EDWARDS, W., LINDMAN, H., & SAVAGE, L. J. (1963). Bayesian statistical inference in psychological research. *Psychological Review*, **70**, 193-242.

GELFAND, A. E., & SMITH, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, **85**, 398-409.

GEMAN, S., & GEMAN, D. (1984). Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, **6**, 721-741.

GEWEKE, J. (1992). Evaluating the accuracy of sampling-based approaches to calculating posterior moments. In J. M. Bernado, J. O. Berger, A. P. Dawid, & A. F. M. Smith (Eds.), *Bayesian statistics 4* (pp. 169-193). Oxford: Oxford University Press, Clarendon Press.

GILKS, W. R., RICHARDSON, S., & SPIEGELHALTER, D. J. (Eds.) (1996). *Practical Markov chain Monte Carlo*. New York: Chapman & Hall.

NETER, J., KUTNER, M. H., NACHTSHEIM, C. J., & WASSERMAN, W. (1996). *Applied linear statistical models* (4th ed.). Chicago: Irwin.

PRESS, S. J. (1989). *Bayesian statistics*. New York: Wiley.

SPIEGELHALTER, D. J., THOMAS, A., BEST, N. G., & GILKS, W. (1997). *BUGS: Bayesian inference using Gibbs sampling, Version 0.60*. Cambridge: Medical Research Council Biostatistic Unit.

TANNER, M. A. (1993). *Tools for statistical inference* (3rd ed.). New York: Springer-Verlag.

(Manuscript received October 13, 1997;
revision accepted for publication January 7, 1998.)