

Mary Kathryn COWLES

WinBUGS, a software package that uses Markov chain Monte Carlo (MCMC) methods to fit Bayesian statistical models, has facilitated Bayesian analysis in a wide variety of applications areas. This review shows the steps required to fit a Bayesian model with WinBUGS, and discusses the package's strengths and weaknesses. WinBUGS is highly recommended for both simple and complex Bayesian analyses, with the caveat that users require knowledge of both Bayesian methods and issues in MCMC.

KEY WORDS: Bayesian analysis; MCMC.

1. INTRODUCTION

WinBUGS is general-purpose software for fitting arbitrarily complex Bayesian models using Markov chain Monte Carlo (MCMC) methods. The stated aim of its developers is to make practical MCMC methods available to applied statisticians. A brief search for recently published papers referencing WinBUGS turned up applications in food safety, forestry, mental health policy, AIDS clinical trials, population genetics, pharmacokinetics, pediatric neurology, and other diverse fields, indicating that Bayesian methods with WinBUGS indeed are finding widespread use.

In the Bayesian framework, there may be information about model parameters θ available to the user before data are collected. This information is quantified by putting a probability distribution $p(\theta)$, called the "prior," on the parameters. The information regarding model parameters contained in the data y from the new study is expressed in the "likelihood," which is proportional to the distribution of the observed data given the model parameters, $p(y|\theta)$. The information in the likelihood and prior is combined to produce an updated probability distribution—the "posterior distribution" or $p(\theta|y)$ —which is the basis of Bayesian inference. Proportional to the product of the prior and the likelihood, the posterior distribution theoretically is always available. However, in realistically complex models, the analytic computations often are intractable, particularly the required integrations to obtain the normalizing constant of the joint posterior distribution and the calculations of posterior marginal distributions of individual parameters of interest. For typical high-dimensional models, even standard numerical integration techniques are inadequate.

Markov chain Monte Carlo (MCMC) methods (Metropolis et al. 1953; Geman and Geman 1984; Gelfand and Smith 1990) enable the drawing of samples from the joint posterior distribution

of model parameters. Characteristics of the joint and marginal posterior distributions (e.g., posterior means and credible sets) can be estimated from these samples. Thus, Bayesian inference can proceed without the need for intractable analytic or numerical integrations. A Markov chain is a sequence of random variables, X_0, X_1, X_2, \dots , that is characterized by its "transition kernel." The transition kernel is a probability distribution from which the values of any element, say X_{j+1} , of the Markov chain are drawn, conditional only on the values of the preceding element, X_j . Initial values X_0 must be provided. Subject to certain conditions, the sequence of draws constituting a Markov chain will converge in distribution to draws from a limiting distribution called the "stationary" or "target" distribution. The trick in using MCMC for Bayesian analysis is the construction of the transition kernel such that the stationary distribution of the resulting Markov chain is the joint posterior distribution of interest.

WinBUGS enables the user to specify a Bayesian model, either by drawing a directed graph (see, e.g., Lauritzen and Spiegelhalter 1988) or by using an S-like language. The software then determines the transition kernel for a Markov chain to generate samples from the joint posterior distribution of the unknown quantities in the model. Using either a graphical user interface or a script, the user specifies the number of parallel MCMC chains to be run, the number of iterations, the model unknowns to monitor for analysis and reporting, and the types of convergence assessment and output summaries. The final result is numeric and graphical summaries of the estimated univariate marginal posterior distributions of the requested model quantities.

Use of MCMC methods, including WinBUGS, for Bayesian inference requires knowledge and skill beyond statistical modeling. The choice of initial values may affect convergence substantially. Other decisions include the choice to run a single Markov chain or several independent chains initialized at different values, and the selection of the number of early iterations to discard before the sampler is judged to have converged closely enough to the target distribution to provide reasonable inference. Furthermore, because the samples produced by a Markov chain are correlated, a larger number of samples are required for a desired degree of accuracy in estimation than would be the case with independent samples. Literature containing recommendations on how to deal with these issues in MCMC use includes Brooks (1998), Cowles and Carlin (1996), and Gilks, Richardson, and Spiegelhalter (1995).

2. BACKGROUND

"BUGS," an acronym for "Bayesian inference Using Gibbs Sampling," began in 1989 as a statistical research project at the Medical Research Council Biostatistics Unit in Cambridge, UK. The original form of the software, now referred to as "classic BUGS" was written in Modula-2 and could be compiled and run under DOS and some Unix implementations. To fit a model to data, BUGS requires three input files: one containing a spec-

Mary Kathryn Cowles is Associate Professor, Department of Statistics and Actuarial Science and Department of Biostatistics, The University of Iowa, Iowa City, IA (kate-cowles@uiowa.edu).

ification of the model in the BUGS language, one containing the data, and the third containing initial values for the Markov chain. A simple command-line interface then drives the BUGS session, either interactively or in background mode by means of a script. The output of BUGS is samples drawn from the joint posterior distribution of the model unknowns. Because classic BUGS provides only limited facilities for checking convergence and summarizing the distributions of the samples, a separate output postprocessor is generally used for these purposes. Classic BUGS works only for Bayesian models in which all univariate full conditionals are either standard probability distributions or log-concave functions. Although versions 0.5 and 0.6 are still available, classic BUGS no longer is undergoing development.

The first release of WinBUGS, a complete rewrite of BUGS for Windows, appeared in 1997. Its new features included a sophisticated graphical user interface (GUI), flexible variate-generation methods that enable fitting much broader classes of models than had been possible with classic BUGS, and built-in facilities for numerical and graphical univariate summarization of the samples produced and for rudimentary convergence assessment. Ongoing development of WinBUGS is conducted jointly by the MRC Biostatistics Unit and the Imperial College School of Medicine at St Mary's, London. The current release, on which this review is based, is version 1.4, dated January 2003.

3. OBTAINING WINBUGS

An "educational" version of WinBUGS, in which the number of nodes in a single model is limited to 100 (except that it will run all the examples in the on-line help) can be downloaded from <http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents>.

Users must register online in order to obtain the "key" file that transforms the educational version into the unrestricted version. However, as cheerfully stated in the WinBUGS license agreement window, "The current fee is zero dollars (\$0)."

4. FITTING BAYESIAN MODELS INTERACTIVELY IN WINBUGS

We will illustrate the use of WinBUGS by fitting a simple normal hierarchical model to the peak discharge data in Table 4-4 of Montgomery (1991). These data are measurements taken at a watershed using four different methods of measuring peak

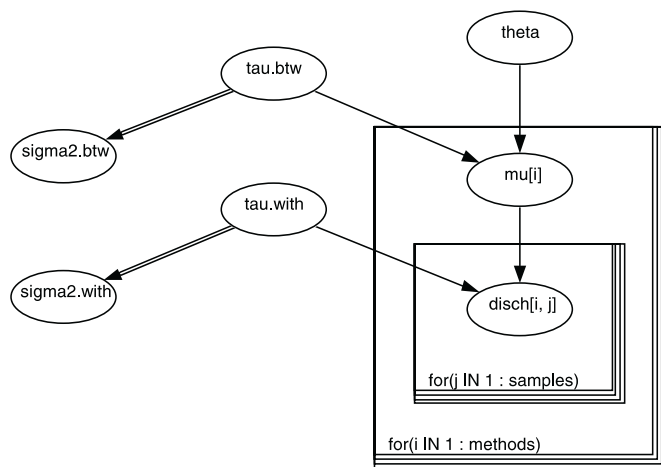


Figure 1. Simple model displayed by the Doodle facility in WinBUGS.

discharge. Six measurements were taken using each method. The response variable is peak discharge in cubic feet per second. As recommended by Montgomery, we used the square root transformation to stabilize variance.

Let $\text{disch}[i, j]$ represent the square root of the j th measurement of peak discharge ($j = 1, \dots, 6$) taken by the i th method ($i = 1, \dots, 4$). The model parameters are $\mu[i]$, $i = 1, \dots, 4$, the subpopulation mean of all possible measurements taken by method i at this watershed at this time; θ , the overall population mean of the means from all possible measurement methods at this watershed at this time; sigma2.btw , the variance between subpopulation means from different measurement methods; and sigma2.with , the variance between different measurements taken by the same method. WinBUGS requires that the normal distribution be parameterized in terms of its mean and *precision*, the latter being the inverse of the variance. We will define tau.btw and tau.with as the inverses of sigma2.btw and sigma2.with , respectively. With " \sim " denoting "is distributed as" and all normal distributions parameterized as $\text{Normal}(\text{mean}, \text{precision})$, the full Bayesian model is:

1. Likelihood

```
disch[i,j] | mu[i], tau.with
  ~ Normal( mu[i], tau.with),
i = 1 . . . 4, j = 1 . . . 6.
```

2. Second stage

```
mu[i] | theta, tau.btw
  ~ Normal( theta, tau.btw),
i = 1 . . . 4
```

3. Priors

```
theta ~ Normal(0, 10-6)
tau.with ~ Gamma( 0.001, 0.001)
tau.btw ~ Gamma( 0.001, 0.001)
```

The priors at the third stage are a vague normal prior (i.e., with very small precision) on θ , and vague gamma priors on tau.btw and tau.with . We note here that the vague prior on tau.btw works satisfactorily for this dataset because the data values within each group are well separated from those in other groups; such a prior generally does not work well for this model.

4.1 Specifying the Model

Simple models may be specified graphically using the drag-and-drop "Doodle" facility in WinBUGS. The doodle for our example is shown in Figure 1.

WinBUGS automatically generated the following model-specification code from the doodle in Figure 1:

```
model
{
  theta ~ dnorm( 0.0, 1.0E-6)
  for( i in 1 : methods ) {
    mu[i] ~ dnorm(theta, tau.btw)
  }
}
```

disch[,1]	disch[,2]	disch[,3]	disch[,4]	disch[,5]	disch[,6]
0.583	0.346	1.109	0.837	1.323	0.346
0.954	1.715	1.463	1.536	1.691	2.133
2.512	2.893	3.122	2.468	3.134	2.691
4.141	3.438	3.309	4.147	3.788	4.101

END

Figure 2. Tabular format for data for example.

```
for( i in 1 : methods ) {
  for( j in 1 : samples ) {
    disch[i , j] ~ dnorm(mu[i],
      tau.with)
  }
}
tau.btw ~ dgamma(0.0001,0.0001)
tau.with ~ dgamma(0.0001,0.0001)
sigma2.btw <- 1 / tau.btw
sigma2.with <- 1 / tau.with
}
```

Alternatively, the user may simply write model-specification code as text. (Complex models must be specified in this way.) Each stage of the Bayesian model (likelihood and all levels of priors) must be defined using WinBUGS 1.4's 23 probability distributions, 28 functions, and S-like looping structure. Probability distributions that are not part of WinBUGS' built-in set may be introduced at either the likelihood or prior stages of the model using methods documented in the section of the user manual entitled "Tricks: Advanced Use of the BUGS Language." Although WinBUGS' model-specification language has no explicit structure for if/then logic, clever use of the "step" function often is a workable substitute.

Since WinBUGS can compute transformations of parameters, it is easy to direct WinBUGS to transform simulated values of precision parameters into the corresponding values of variance parameters, as is done in the last two lines of the example code.

4.2 Providing Data and Initial Values

Fitting the model requires the following additional inputs: the data and values for all model constants, and a set of initial values for each parallel chain.

Data can be provided to WinBUGS in one of two formats, either in a list format similar to that used by S-Plus and R or in tabular (rows and columns) format. Scalar constants must be provided in the list format. Here is a data list for our example, containing both the constants and the actual data. Note that the required format for a two-dimensional array of data does not exactly match that output by S-Plus or R, so a small amount of either manual editing or special programming in S-Plus or R is required to produce it.

```
list(methods = 4, samples = 6,
      disch = structure(.Data=
c(0.583, 0.346, 1.109, 0.837, 1.323, 0.346,
  0.954, 1.715, 1.463, 1.536, 1.691, 2.133,
  2.512, 2.893, 3.122, 2.468, 3.134, 2.691,
  4.141, 3.438, 3.309, 4.147, 3.788, 4.101),
      .Dim = c(4,6)))
```

Alternatively, we could have split this input into two parts, a list for the constants:

```
list(methods = 4, samples = 6)
```

and a table for the actual data (see Figure 2). Note that column headings are required for tabular data, and that the keyword "END" followed by a carriage return must appear following the last line of data.

We wished to run five parallel chains. Below are the five widely dispersed sets of initial values we used. Note that WinBUGS requires initial values to be provided for all precision parameters, but is able to generate random initial values for most other types of parameters as long as they have proper priors. However, asking WinBUGS to generate its own initial values for parameters with vague priors can result in such poor choices of initial values that computational problems result. Thus, it often is wise for the user to provide initial values even for those parameters for which WinBUGS is capable of generating them. This was done in this example.

```
list( mu = c(1,2,3,4), tau.with = 1,
      tau.btw = 1, theta = 2.24)
list( mu = c(2.24, 2.24, 2.24, 2.24),
      tau.with = 0.01, tau.btw = 10000,
      theta = 2.24)
list( mu = c(0.75,1.5,3,4), tau.with = 5,
      tau.btw = 0.0001, theta = -2.5)
list( mu = c(-5, 0, 5, 10), tau.with = 0.001,
      tau.btw = 0.001, theta = 2.5)
list( mu = c(-100, -50, 0, 50),
      tau.with = 0.000001, tau.btw = 0.00001,
      theta = -25)
```

4.3 Running the Chains

To run the model, the user selects "Model Specification" from a pull-down menu, summoning the "Specification Tool" (see Figure 3). The user then steps through checking the model syntax, loading the data, compiling the model, and loading initial values by highlighting keywords in the appropriate input files and then clicking the respective button. If data or initial values are in multiple files, the steps to load data and to load initial values are executed repeatedly. WinBUGS may give error messages at any step in the process. The user cannot proceed to the next step until corrections have been made in the input file causing the errors.

After the model has been checked and compiled and the data and initial values loaded, the user must tell WinBUGS which quantities in the model should be "monitored"—that is, should have their samples retained for output analysis. This is done by

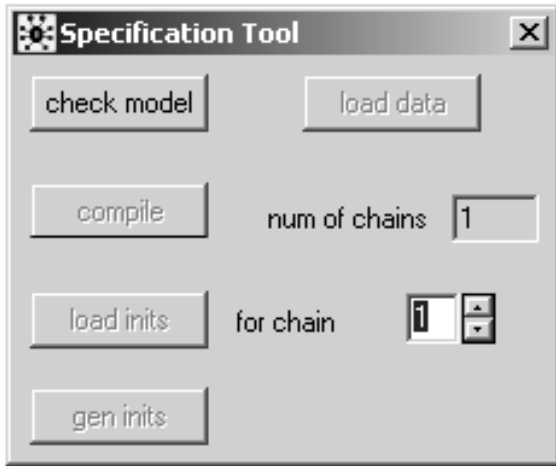


Figure 3. Screenshot of WinBUGS' Specification Tool.

typing the name of each desired node into the "node" window of the Sample Monitor Tool (Figure 4) and clicking "set."

The Update Tool is then used to request the number of iterations to be run. After the samples have been generated, output analysis is requested by selecting one of the monitored nodes (or typing an asterisk in the node window to request output for all monitored nodes) and clicking on an output button in the Sample Monitor Tool.

4.4 Analyzing the Output

History plots and the Brooks-Gelman-Rubin diagnostic may assist in determining how many initial iterations should be discarded (burn-in) and whether sufficient iterations have been run. Showing the trajectories of the samples of a particular parameter produced by each chain (different colors for different chains), the history plot enables qualitative determination of whether the chains, started at dispersed initial values, have coalesced and are drawing from a common distribution. History plots (Figure 5) may also be restricted to show one chain at a time to determine whether any one or more individual chains show aberrant behavior.

The "bgr" button produces a graphical version of the Gelman and Rubin diagnostic (Figure 6) as modified by Brooks and Gelman. See the WinBUGS on-line documentation and Brooks and Gelman (1998) for details on interpreting this diagnostic.

The "beg," "end," and "thin" fields in the Samples tool enable the user to select which iterations will be used in computing the



Figure 4. Screenshot of WinBUGS' Sample Monitor Tool.

summaries of the estimated posterior distributions of model unknowns. Among the available options for summarizing these estimated posterior distributions are density plots (smoothed kernel density plots for continuous quantities and bar graphs for discrete ones; see Figure 7) and tabular summaries (see Figure 8). The table columns represent the node name; the estimated mean and standard deviation of the posterior distribution; the autocorrelation-adjusted standard error of the estimated posterior mean; the 2.5%, 50%, and 97.5% quantiles of the posterior distribution; the iteration number of the first iteration used in the estimation (i.e., the first post-burn-in iteration); and the total number of sample values used in the estimation.

The "Coda" button on the Samples tool enables exporting the samples to external text files for further analysis outside of WinBUGS.

The "Compare" tool offers facilities for comparing the posterior distributions of sets of model parameters. Figure 9 shows boxplots to compare the posterior distributions of the mu parameters for the four different measurement methods.

WinBUGS can compute the deviance information criterion (DIC; Spiegelhalter, Best, Carlin, and van der Linde 2002), which is useful in comparing the fit of two or more models for the same data. To compute the DIC, WinBUGS evaluates the log-likelihood at each iteration of the sampler, using the parameter values from that iteration. In the DIC output table, "Dbar" is -2 times the sample average of the log-likelihoods; "Dhat" is -2 times the log-likelihood evaluated at the posterior mean of the parameters; "pD," calculated as $Dbar - Dhat$, is the effective number of parameters in the model; and the "DIC"

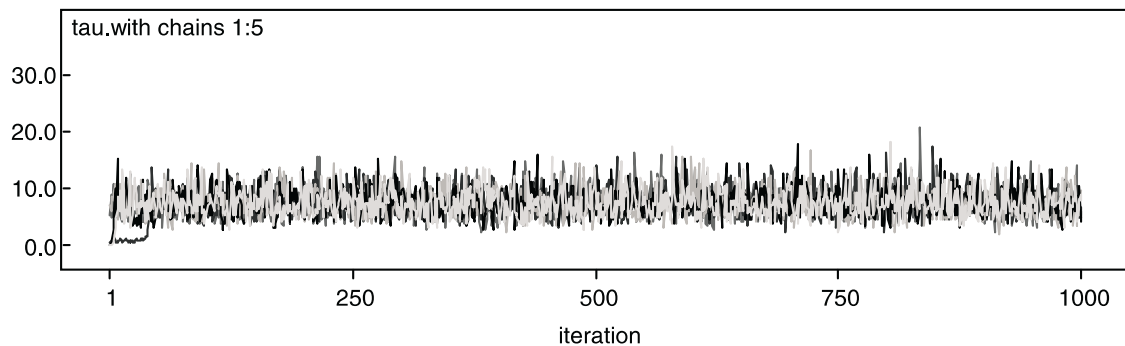


Figure 5. History plot of output for one parameter from three chains.

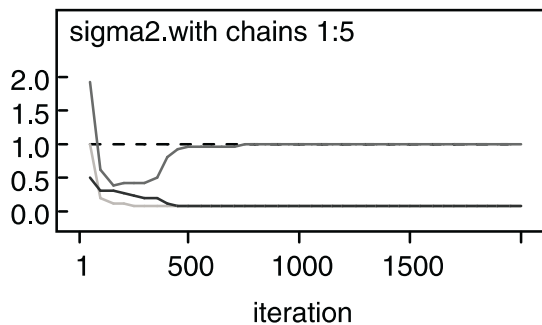


Figure 6. Graphical version of BGR diagnostic for one parameter.

(\bar{D} + pD) is the comparative index of model fit, with smaller values indicating better fit.

\bar{D} = post.mean of $-2\log L$; \hat{D} = $-2\log L$ at post.mean of stochastic nodes

	\bar{D}	\hat{D}	pD	DIC
disch	21.222	16.011	5.211	26.433

4.5 Comparing the WinBUGS Output With Other Results

Statisticians who use iterative maximization methods for frequentist statistical estimation (e.g., the algorithms employed by `proc mixed` and `proc nlin` in SAS for mixed and nonlinear models, respectively) are aware that such methods can fail to converge or can converge to minor rather than global maxima. The same types of pitfalls, and more, are possible with MCMC. Among the recommendations of Cowles and Carlin (1996), as well as other authors, to guard against erroneous inference is to check the output of any MCMC sampler against results obtained in other ways. For example, the frequentist or empirical Bayes model that is most similar to the desired Bayesian model might be fit using standard statistical software. For this purpose, we used `proc mixed` in version 9.1 of SAS to obtain empirical Bayes estimates of the parameters of our model using the REML estimation method. Because we specified such vague priors in our Bayesian model, we would expect close correspondence between the two sets of estimates. However, due to the right skewness in the posterior distributions of the two variances (refer to the plot of the estimated Bayesian posterior marginal density of `sigma2.with`; the shape of the plot for `sigma2.btw` is even more skewed), we would expect the Bayesian posterior means to be larger than the maximum likelihood or REML estimates, which correspond to the modes of the likelihood.

`Proc mixed` produced the following point estimates: $\mu[1] = 0.78$, $\mu[2] = 1.59$, $\mu[3] = 2.80$, $\mu[4] = 3.80$, $\sigma^2.btw = 0.134$, $\sigma^2.with = 1.79$, $\theta = 2.24$. As expected, the empirical Bayes point estimates of the μ s and θ are very close to the estimated Bayesian posterior means, while the REML point estimates of the two components of variance are smaller than the estimated Bayesian posterior means (only a little smaller than the Bayesian posterior medians). This is evidence that our MCMC sampler is indeed drawing from a reasonable approximation to the true posterior distribution. (Additional appropriate comparisons, such as posterior standard deviations with frequentist standard errors, are not shown.)

4.6 Using a Script to Run WinBUGS in Batch Mode

A new feature introduced in version 1.4 of WinBUGS is the scripting facility, an alternative to the menu/dialog-box interface that makes it possible to automate routine analysis, to drive WinBUGS from other programs such as S-Plus or R, and to carry out simulation studies involving WinBUGS analysis. Commands in the script language substitute for menu selections and dialog-box entries. Running WinBUGS in batch mode requires a file containing the script in addition to the model-specification file, the data file(s), and a separate file of initial values for each desired MCMC chain.

5. PLATFORMS ON WHICH WINBUGS RUNS

Written in Component Pascal, WinBUGS depends on the Black Box component, which is available only for Windows platforms. Efforts have been made to run WinBUGS under Linux using Wine and VMware. Martyn Plummer reports (<http://www-fis.iarc.fr/bugs/wine/>) that a bug first introduced into Wine in a 2001 release prevents the WinBUGS buttons from responding to clicks. However, WinBUGS can run under Wine using the script interface. The Frequently Asked Questions section of the WinBUGS Web page states, "Others have reported successful and stable running under VMWare."

6. SAMPLING METHODS IN WINBUGS

WinBUGS uses the Gibbs sampling algorithm to construct the transition kernels for its Markov chain samplers. Each iteration of a Gibbs sampler involves drawing a new value for each parameter from its "full conditional distribution"—the conditional probability distribution of that parameter given the current values of all other quantities in the model. During compilation, WinBUGS chooses a method to draw samples from each the full conditional distribution of each model parameter. Such sampling is done univariately except in the case of explicitly defined multivariate nodes and, if the user has so indicated, vectors of coefficients in generalized linear models. Samples from continuous conjugate full conditionals are drawn directly using standard algorithms. For nonstandard but log-concave full conditionals, derivative-free adaptive rejection sampling (Gilks 1992) is used.

The slice-sampling algorithm (Neal 1997) used by WinBUGS for non log-concave densities on a restricted range has a tuning phase of 500 iterations.

The random walk Metropolis algorithm (Metropolis et al. 1953) is used in WinBUGS for nonconjugate continuous full conditionals with an unrestricted range. The standard deviation of the normal proposal distribution is tuned over the first 4,000 iterations to obtain an acceptance rate of between 20% and 40%.

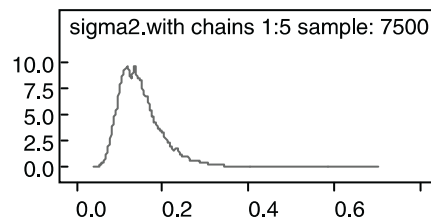


Figure 7. A posterior density plot.

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
mu [1]	0.7747	0.1594	0.001823	0.4628	0.7748	1.089	501	7500
mu [2]	1.588	0.1608	0.001802	1.27	1.587	1.916	501	7500
mu [3]	2.795	0.1583	0.00191	2.478	2.795	3.111	501	7500
mu [4]	3.798	0.1586	0.001911	3.485	3.798	4.115	501	7500
sigma2.btw	5.202	20.83	0.3268	0.5519	2.262	23.61	501	7500
sigma2.with	0.1507	0.05442	8.017E-4	0.07801	0.1401	0.2888	501	7500
tau.btw	0.5637	0.4737	0.00654	0.04236	0.4421	1.812	501	7500
tau.with	7.406	2.404	0.03509	3.463	7.136	12.82	501	7500
theta	2.231	1.104	0.01287	0.06966	2.237	4.334	501	7500

Figure 8. Tabular summary for estimated posterior distributions.

Samples from the tuning phases of both the slice-sampling and Metropolis algorithms are ignored in the calculation of all summary statistics, although they will appear in trace plots.

7. GEOBUGS

Originally developed as a spatial-analysis add-on to WinBUGS by a team at the Department of Epidemiology and Public Health of Imperial College at St Mary's Hospital London, GeoBUGS now is included as part of the WinBUGS 1.4 distribution. It enables exchanging map files between WinBUGS and S-Plus, ArcInfo, and Epimap; fitting Bayesian geostatistical and conditional autoregressive models to spatial data; and mapping the results.

8. DOCUMENTATION AND RESOURCES FOR LEARNING

The documentation built into WinBUGS is unusually good, and a multitude of other resources exist for learning WinBUGS and sharing expertise with other users.

The "Help" menu in WinBUGS provides both a printable user manual and two volumes of worked examples. The user manual includes a tutorial and clear documentation of the model-

description language, all menus and dialog boxes, and the script facility. Documentation of GeoBUGS, including a complete user manual, is provided under the "Map" pull-down menu.

Additional WinBUGS examples, links to a wealth of on-line resources concerning Bayesian analysis in general and WinBUGS in particular, and instructions for joining the users' e-mail listserv are available on the BUGS/WinBUGS Web page (see the URL under "Obtaining WinBUGS," Section 3).

9. ADD-ONS FOR WINBUGS

Several auxiliary pieces of free software facilitate the use of WinBUGS.

Two R packages, R2WinBUGS (developed by Andrew Gelman, Sibylle Sturtz, and Uwe Ligges for the Windows platform only) and rbugs (developed by Jun Yan for either Linux with Wine or Windows) permit driving script-based WinBUGS sessions from within R. Enabling R to automate generation of data files and initial values files in the format required by WinBUGS and to process the samples output by WinBUGS, these packages greatly facilitate simulation studies. Both may be downloaded from the Comprehensive R Archive Network at <http://cran.r-project.org/>.

PKBugs, developed by David Lunn, is an interface for specifying complex population pharmacokinetic/pharmacodynamic (PK/PD) models in WinBUGS software. It is available at http://www.med.ic.ac.uk/divisions/60/pkbugs_web/home.html.

The Bayesian Output Analysis system (BOA), developed by Brian Smith, is a menu-driven suite of S-Plus/R routines that offer a much wider range of convergence-assessment and output-analysis features than those built into WinBUGS. Both the S-Plus version and the R package may be downloaded from <http://www.public-health.uiowa.edu/boa/>, and the R package is also available at the CRAN Web site given above.

10. DRAWBACKS

Like any other software, WinBUGS has some drawbacks. As general-purpose software, it is not optimized for specific models, and consequently run-times may be very long. For very large datasets, memory limitations may prevent the use of WinBUGS. There are some models that even the newest version of WinBUGS cannot fit; for example, spatiotemporal models with highly structured covariance structures.

Error reporting is one of the weakest points of WinBUGS. If computational problems arise during a WinBUGS run, execu-

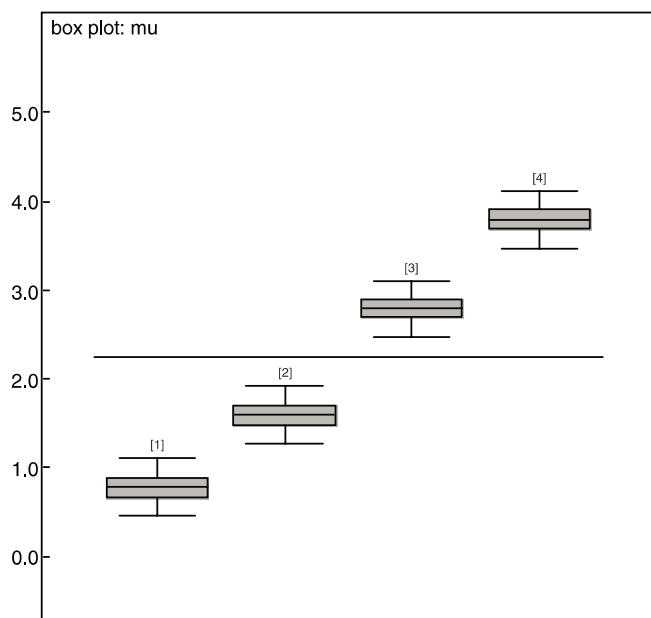


Figure 9. Boxplots for comparing posterior marginal distributions.

tion stops and a window headed “Trap” appears. The contents of the window are generally unintelligible. This can be extremely frustrating, especially for a novice user. Fortunately, the user manual gives clear suggestions on solutions to try for different types of traps, with a better choice of initial values often sufficing.

Obtaining valid results with WinBUGS requires that the user have knowledge of both Bayesian modeling and MCMC-related issues, especially selection of initial values and convergence assessment. As stated on the WinBUGS Web page, “The programs are reasonably easy to use and come with a range of examples. Considerable caution is, however, needed in their use, since the software is not perfect and MCMC is inherently less robust than analytic statistical methods. There is no in-built protection against misuse.”

11. SUMMARY

My own research involves developing Bayesian models for biomedical and environmental applications. BUGS and early releases of WinBUGS were unable to handle the models I needed, so I had to code my own MCMC samplers in C. However, the increased capabilities of WinBUGS versions 1.3 and 1.4 have enabled me to fit almost all of my models in WinBUGS, thereby both saving me time and enabling me to offer applied statisticians a more user-friendly way of using my methods. Exceptions are models with complex and highly structured covariance structures. I am convinced that, with clever programming tricks, WinBUGS can fit most of the models that are likely to be needed in applied practice in many disciplines. WinBUGS is reasonably easy to use, and its built-in output-analysis features are sufficient for most purposes.

In short, I highly recommend WinBUGS for Bayesian model fitting, with the caveat that advice from a statistician experienced with the use of MCMC is likely to be needed.

REFERENCES

- Brooks, S. (1998), “Markov Chain Monte Carlo and its Applications,” *The Statistician*, 47, 69–100.
- Brooks, S. P., and Gelman, A. (1998), “Alternative Methods for Monitoring Convergence of Iterative Simulations,” *Journal of Computational and Graphical Statistics*, 7, 434–455.
- 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.
- Gelfand, A. E., and Smith, A. F. M. (1990), “Sampling-Based Approaches to Calculating Marginal Densities,” *Journal of the American Statistical Association*, 85, 389–409.
- 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.
- Gilks, W. (1992), “Derivative-Free Adaptive Rejection Sampling for Gibbs Sampling,” in *Bayesian Statistics 4*, eds. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, UK: Oxford University Press, pp. 641–666.
- Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (eds.) (1996), *Markov Chain Monte Carlo in Practice*, London: Chapman and Hall.
- Lauritzen, S. L., and Spiegelhalter, D. J. (1988), “Local Computations With Probabilities on Graphical Structures and their Applications to Expert Systems” (with discussion), *Journal of the Royal Statistical Society, Series B*, 50, 157–224.
- Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953), “Equations of State Calculations by Fast Computing Machines,” *Journal of Chemical Physics*, 21, 1087–1091.
- Montgomery, D. C. (1991), *Design and Analysis of Experiments* (3rd ed.), New York: Wiley.
- Neal, R. (1997), “Markov Chain Monte Carlo Methods Based on ‘Slicing’ the Density Function,” Technical Report 9722, Department of Statistics, University of Toronto, Canada. Available on-line at <http://www.cs.utoronto.ca/~radford/publications.html>.
- SAS Institute, Inc. (2004), *SAS/STAT 9.1 User’s Guide*, Cary, NC: SAS Institute Inc.
- Spiegelhalter, D. J., Best, N. G., Carlin, B. R., and van der Linde, A. (2002), “Bayesian Measures of Model Complexity and Fit,” *Journal of the Royal Statistical Society, Series B*, 64, 583–561.
- Spiegelhalter, D., Thomas, A., Best, N., and Lunn, D. (2003), *WinBUGS 1.4 Manual*.