M. Plummer. *JAGS 0.90 User Manual*. IARC, Lyon, September 2005. URL http://www-ice.iarc.fr/~martyn/software/jags. 9

B. J. Smith. The boa package. 2005. URL http://www.public-health.uiowa.edu/boa/. 8

D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. *BUGS 0.5: Bayesian inference Using Gibbs Sampling - Manual (version ii)*. Medical Research Council Biostatistics Unit, Cambridge, Cambridge, 1995. 8

D. Spiegelhalter, A. Thomas, N. Best, and D. Lunn. *WinBUGS user manual, version 2.0*. Medical Research Council Biostatistics Unit, Cambridge, Cambridge, June 2004. URL http://www.math.stat.helsinki.fi/openbugs. 8

Martyn Plummer
International Agency for Research on Cancer
Lyon, France
plummer@iarc.fr

Nicky Best
Department of Epidemiology and Public Health
Faculty of Medecine
Imperial College
London, UK

Kate Cowles
Department of Biostatistics, College of Public Health
The University of Iowa, USA

Karen Vines
Department of Statistics
The Open University
Milton Keynes, UK

Bayesian Software Validation

by Samantha Cook and Andrew Gelman

BayesValidate is a package for testing Bayesian model-fitting software. Generating a sample from the posterior distribution of a Bayesian model often involves complex computational algorithms that are programmed "from scratch." Errors in these programs can be difficult to detect, because the correct output is not known ahead of time; not all errors lead to crashes or results that are obviously incorrect. Software is often tested by applying it to data sets where the "right answer" is known or approximately known. Cook et al. (2006) extend this strategy to develop statistical assessments of the correctness of Bayesian model-fitting software; this method is implemented in **BayesValidate**. Generally, the validation method involves simulating "true" parameter values from the prior distribution, simulating fake data from the model, performing inference on the fake data, and comparing these inferences to the "true" values. Geweke (2004) presents an alternative simulation-based method for testing Bayesian software.

More specifically, let $\theta^{(0)}$ represent the "true" parameter value drawn from the prior distribution $p(\theta)$. Data y are drawn from $p(y|\theta^{(0)})$, and the posterior sample of size L to be used for inference, $\theta^{(1)}, \ldots, \theta^{(L)}$, is drawn using the to-be-tested software. With this sampling scheme, $\theta^{(0)}$ as well as $\theta^{(1)}, \ldots, \theta^{(L)}$ are, in theory, draws from $p(\theta|y)$. If the Bayesian software works correctly, then, $\theta^{(0)}$ should look like a random draw from the empirical distribution $\theta^{(1)}, \ldots, \theta^{(L)}$, and therefore the (empirical) posterior quantile of $\theta^{(0)}$ with respect to $\theta^{(1)}, \ldots, \theta^{(L)}$ should follow a Uniform(0, 1) distribution. Testing the software amounts to testing that the posterior quantiles for scalar parameters of interest are in fact uniformly distributed.

One "replication" of the validation simulation consists of: 1) Generating parameters and data; 2) generating a sample from the posterior distribution; and 3) calculating posterior quantiles. Performing many replications creates, for each scalar parameter whose posterior distribution is generated by the model-fitting software, a collection of quantiles whose distribution will be uniform if the software works correctly. If N_{rep} is the number of replications and $q_1, q_2, \ldots, q_{N_{rep}}$ are the quantiles for a scalar parameter, the quantity $\sum_{i=1}^{N_{rep}} \left(\Phi^{-1}(q_i)\right)^2$ will follow a $\chi^2_{N_{rep}}$ distribution if the software works correctly, where Φ^{-1} represents the inverse normal cumulative distribution function (CDF). For each scalar parameter, a p-value is then obtained by comparing the sum of the transformed quantiles with the $\chi^2_{N_{rep}}$ distribution. **BayesValidate** analyzes each scalar parameter separately, but also creates combined summaries for each vector parameter; these scalar results and summaries are in the graphical output as well.

BayesValidate performs a specified number of replications and calculates a p-value for each scalar parameter. The function returns a Bonferroni-adjusted p-value and a graphical display of the z_θ statistics, which are the inverse normal CDFs of the p-values. Figures 1 and 2 show the graphical output for two versions of a program written to fit a simple hierarchical normal model with parameters σ^2, τ^2, μ, and $\alpha_1, \alpha_2, \ldots, \alpha_6$; one version correctly samples from the posterior distribution and one has an error.

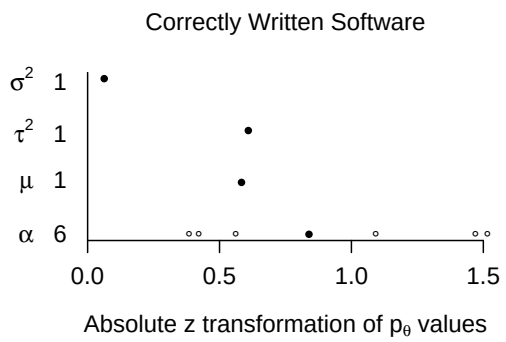

Figure 1: *z_θ statistics: Correctly written software. Each row represents a scalar parameter or batch of parameters; the circles in each row represent the z_θ statistics associated with that parameter or batch of parameters. Solid circles represent the z_θ statistics associated with the mean of that batch of parameters. The numbers on the y axis indicate the number of parameters in the batch. The z_θ statistics are all within the expected range for standard normal random variables.*

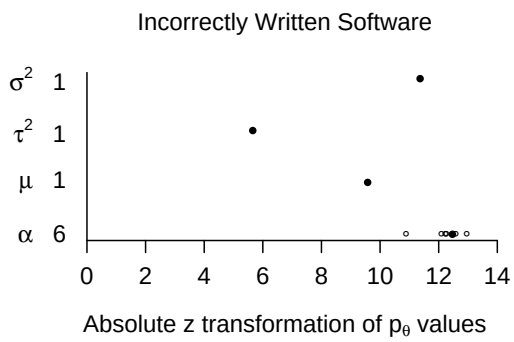

Figure 2: *z_θ statistics: Incorrectly written software (error sampling the parameter α). Each row represents a scalar parameter or batch of parameters; the circles in each row represent the z_θ statistics associated with that parameter or batch of parameters. Solid circles represent the z_θ statistics associated with the mean of that batch of parameters. The numbers on the y axis indicate the number of parameters in the batch. Values of z_θ larger than 2 indicate a potential problem with the software; this plot provides convincing evidence that the software has an error.*

Bibliography

S. Cook, A. Gelman, and D. B. Rubin. Validation of software for Bayesian models using posterior quantiles. *Journal of Computational and Graphical Statistics*, 2006. To appear. 11

J. Geweke. Getting It Right: Joint Distribution Tests of Posterior Simulators. *Journal of the American Statistical Association*, 99:799–804, 2004. 11

Samantha Cook, Andrew Gelman
Department of Statistics
Columbia University, NY, USA

Making BUGS Open

by Andrew Thomas, Bob O'Hara, Uwe Ligges, and Sibylle Sturtz

BUGS[1] (Bayesian inference Using Gibbs Sampling, Spiegelhalter et al., 2005) is a long running software project aiming to make modern Bayesian analysis using Markov Chain Monte Carlo (MCMC) simulation techniques available to applied statisticians in an easy to use Windows package. With the growing realization of the advantages of Open Source software we decided to release the source code of the BUGS software plus full program level documentation on the World Wide Web[2]. We call this release OpenBUGS (Thomas, 2004). We hope the BUGS user community will be encouraged to correct, improve and extend this software.

We follow a brief outline of how the BUGS software works with a more detailed discussion of the software technology used during the development of BUGS. We then try and explain why BUGS was developed using non-standard tools. We hope to convince the reader that although unfamiliar, our tools are very powerful and simple to use.

[1] http://www.mrc-bsu.cam.ac.uk/bugs/
[2] http://mathstat.helsinki.fi/openbugs/