Some thoughts on the BUGS package for Bayesian analysis[1]

Andrew Gelman, Dept of Statistics and Dept of Political Science, Columbia University

19 May 2009

I first saw BUGS in a demonstration version at a conference in 1991, but I didn't take it seriously until over a decade later, when I found that some of my Ph.D. students in political science were using Bugs to fit their models. It turned out that Bugs's modeling language was ideal for students who wanted to fit complex models but didn't have a full grasp of the mathematics of likelihood functions, let alone Bayesian inference and integration. I also learned that the modular structure of BUGS was a great way for students, and researchers in general, to think more about modeling and less about deciding which conventional structure should be fit to data.

Since then, my enthusiasm for BUGS has waxed and waned, depending on what sorts of problems I was working on. For example, in our study of income and voting in U.S. states [1], my colleagues fit all our models in BUGS. Meanwhile we kept running into difficulty when we tried to expand our model in different ways, most notably when going from varying-intercept multilevel regressions, to varying-intercept, varying-slope regressions, to models with more than two varying coefficients per group. Around this time I discovered lmer [2], a function in R which fits multilevel linear and generalized linear models allowing for varying intercepts and slopes. The lmer function can have convergence problems and does not account for uncertainty in the variance parameters, but it is faster than Bugs and in many cases more reliable—so much so that Jennifer Hill and I retooled our book on multilevel models to foreground lmer and de-emphasize Bugs, using the latter more as a way of illustrating models than as a practical tool.

What does BUGS do best and what does it do worst? In short, BUGS excels with complicated models for small datasets. For example, we fit a fairly elaborate discrete choice model to data from a laboratory economics experiment with 30 trials on each of about 100 participants [3]. The sample size was small enough that we wanted to use a hierarchical model to obtain stable inference for all the people, so we programmed something up in BUGS and it worked right away.

Well, not right away. We first had to devise a work-around because BUGS was crashing on a logistic model. We added a couple lines in the model to bound the probabilities between 0.01 and 0.99. As with many situations in which a specification is set up for computational reasons [4], it turned out that the model made statistical sense as well: bounding the probabilities allowed for the occasional outlier, an issue we had never previously thought about in the context of binary data. (see [5] for a cleaner solution to this problem using the t distribution in place of the logistic.) Anyway, the point is that in this example, BUGS worked well, it was the direct solution to a good answer, and even its problems led to a new solution which would actually have been difficult to implement in non-BUGS software.

When does BUGS not work so well? With large datasets, multivariate structures, and regression coefficients. For one thing, it can be difficult to keep track of regression coefficients in the BUGS modeling language, which allows no subroutines or macros.

For example, instead of:

[1] Discussion of "The BUGS project: evolution, critique and future directions," by David Lunn, David Spiegelhalter, Andrew Thomas, and Nicky Best, for *Statistics in Medicine*.

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.hat[i] <- a[county[i]] + b[county[i]]*x[i]
  e.y[i] <- y[i] - y.hat[i]
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 1000)
```

we want something like:

```
y ~ norm (a[county] + b[county]*x, sigma.y)
```

This is all algorithmic and could be programmed, but Bugs is not set up so as to allow this.

Convergence can be slow, and, in fact, each iteration can take a long time to run. When generalizing our model of income and voting to allow four coefficients for each state (corresponding to individual income, religious attendance, their interaction, and a constant term), BUGS basically ground to a halt, to the extent that I wouldn't have trusted its results, even had I decided to be patient and run it all night.

Winston Churchill said that sometimes the truth is so precious, it must be attended by a bodyguard of lies. Similarly, for a model to be believed, it must, except in the simplest of cases, be accompanied by similar models that either give similar results or, if they differ, do so in a way that can be understood. In the example of voting by income and religion, we simply took a step back and fit the model using lmer. This didn't always work either, but when lmer had problems we stripped down the model in various ways until it worked. It ran fast enough that we were able to experiment.

What are the great things about BUGS?

1. It really works! I can use it in my applied research. (For software to be useful to me in this way, it doesn't have to work all the time, it only has to solve some problems that can't easily be solved in other ways, and to break down in recognizable ways, so that it doesn't pretend to work when it doesn't.)
2. BUGS is easy to use and to teach, with intuitive syntax.
3. It is free.
4. It can be called directly from R (see [6] for many illustrations of the advantages of running BUGS in this way, most notably for preprocessing of data and postprocessing of inferences.)

What are some problems with BUGS?

1. It often needs lots of hand-holding adjustment of the model to work.
2. Efficiently-programmed models can get really long, ugly, and bug-prone, as I have learned in implementing redundant parameterizations in hierarchical models [7].
3. You can't debug it by running interactively, as you can in R, Matlab, C, Fortran, etc.
4. There's no easy way to go inside and improve the sampling, either by writing shortcuts in the BUGS language or by inserting your own jumping rules inside the inference engine. Attempts to "trick" BUGS into using efficient updates can sometimes backfire.
5. As far as I can tell, BUGS does not understand hierarchical models but rather thinks of scalar parameters as individual entities. This may be partly a historical issue—before the mid-1990s, hierarchical modeling was generally considered a special topic rather than central to concepts of Bayesian inference and prior information—but, whatever the reason, the current implementation of BUGS does not allow easy setup or fast computation with hierarchical models.

There is also a problem—not really the fault of BUGS itself—that its users typically focus on inference as the only goal, ignoring the other two stages of Bayesian data analysis: model building and model checking. Often it can take awhile to get the model to converge, and once that happens there's a tendency to just stop, relax, and present the results. I believe that an important future development should be automatic implementations of model checking using posterior simulations. This can be done—for example, by creating a "shadow" replicated variable for each variable in the model, and then allowing user-specified or default graphical comparisons—but, like everything else, it requires work.

I shouldn't really complain—after all, BUGS is free and, as noted in the article under discussion, has had an incredibly beneficial influence, especially considering that it was put together by just a few dedicated people. I hope that the next twenty years of BUGS are as successful as the first.

References

1. Gelman, A., Shor, B., Bafumi, J., and Park, D. Rich state, poor state, red state, blue state: What's the matter with Connecticut? *Quarterly Journal of Political Science* 2007; 2, 345-367.

2. Bates, D. Fitting linear models in R using the lme4 package. *R News* 2005; 5 (1), 27-30.

3. Casella, A., Gelman, A., and Palfrey, T. R. An experimental study of storable votes. *Games and Economic Behavior* 2005; 57, 123-154.

4. Gelman, A. Parameterization and Bayesian modeling. *Journal of the American Statistical Association* 2004; 99, 537-545.

5. Liu, C. Robit regression: a simple robust alternative to logistic and probit regression. In *Applied Bayesian Modeling and Causal Inference from Incomplete-Data Perspectives*, ed. A. Gelman and X. L. Meng, 2004; 227-238. London: Wiley.

6. Gelman, A., and Hill, J. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, 2007.

7. Gelman, A., Huang, Z., van Dyk, D., and Boscardin, W. J. Using redundant parameters to fit hierarchical models. *Journal of Computational and Graphical Statistics* 2007; 17, 95-122.