International Journal of Technology Assessment in Health Care, **17:1** (2001), 98–113.
Copyright © 2001 Cambridge University Press. Printed in the U.S.A.

AN ELEMENTARY INTRODUCTION TO BAYESIAN COMPUTING USING WINBUGS

Dennis G. Fryback
Natasha K. Stout
Marjorie A. Rosenberg
University of Wisconsin-Madison

Abstract

Bayesian statistics provides effective techniques for analyzing data and translating the results to inform decision making. This paper provides an elementary tutorial overview of the WinBUGS software for performing Bayesian statistical analysis. Background information on the computational methods used by the software is provided. Two examples drawn from the field of medical decision making are presented to illustrate the features and functionality of the software.

Keywords: Bayes theorem, Statistical models, Medical decision making, Tutorial

The purpose of this paper is to introduce software now available to support Bayesian statistical analysis. We are not experts with the software, but have found it very useful and wish to become more proficient in its use. One way to do this is to help create a community of learners pursuing similar goals with whom we can learn, and this is the ultimate intent of our presentation. What follows is directed to readers with some previous knowledge of Bayesian statistics who have been waiting for statistical packages to support their interest in using a Bayesian approach for data analysis. Introductory and intermediate texts will be cited to provide additional sources for further instruction, reference, and review. We also acknowledge valuable insights from a tutorial paper by Scollnik (11) developed for actuarial modeling.

Bayesian statistical modeling is a powerful technique to analyze data for medical decision making and healthcare technology assessment. The result of a Bayesian analysis is a probability distribution over the parameter(s) of interest given observed data and any prior information. This probability distribution can be used directly to inform decision making or to compute the probability of hypotheses about values of the parameters. The Bayesian result can also be used as input for assessing the expected value of additional information that might be collected to inform the parameter estimates (4).

Supported in part by a contract from The Bayesian Initiative, and NRSA training grant T32-HS00083, through the Agency for Healthcare Research and Quality. Portions of this paper were presented as a short course at the 21st Annual Scientific Meeting of the Society for Medical Decision Making, Reno, Nevada, October 3, 1999.

In the past, while the Bayesian technique has always been powerful, it has not always been practical. Initially Bayesian analysis was generally limited to problems involving a very small set of statistical distributions to describe the prior information and the likelihood of the observed data. These so-called "conjugate" distributions have the property that when the prior distribution and the likelihood function for the data are combined with Bayes theorem, the posterior distribution is of the same type as the prior but with updated parameters. If the analysis involved conjugate distributions, and in a limited number of other cases, the posterior distribution could be derived analytically. However, computational advances have made it possible to evaluate complex Bayesian models by using numerical approximation and simulation techniques. This has increased the range of problems and sophistication of analyses now accessible to Bayesian techniques far beyond those limited to that small set of statistical distributions accessible previously. One of these computational techniques applies Markov chain Monte Carlo (MCMC) simulation. The WinBUGS (Windows-based Bayesian Inference Using Gibbs Sampling) software package has implemented this computational method for Bayesian statistical modeling and is what we address in this elementary tutorial.

WinBUGS is a product of the BUGS (Bayesian Inference Using Gibbs Sampling) project, which is a joint program of the Medical Research Council's Biostatistics Unit in Cambridge and the Department of Epidemiology and Public Health of Imperial College at St. Mary's Hospital in London. The software is distributed at no charge by the collaborators of the BUGS Project from their Web page (10). Our discussion will be directed to the WinBUGS version 1.2 software. We note that the developers of this software warn users on their Web page that "[c]onsiderable caution is, however, needed in [use of BUGS and WinBUGS], since the software is not perfect and MCMC is inherently less robust than analytic statistical methods. There is no in-built protection against misuse."

As users of the software, we add the caveat that initially the software can be difficult and frustrating to use. Previous experience with Bayesian statistics and MCMC eases the learning process, but the learning curve is steep at the beginning. However, once the WinBUGS modeling process is understood, the software is quite accessible. Allowing for the truths of these observations, we have found WinBUGS usable and useful. It is our hope that readers will, with the advised caution, similarly find it a good entrée to Bayesian analysis.

BAYESIAN ANALYSIS AND MCMC

It is beyond our intent to provide a technical introduction to either Bayesian statistical analysis or MCMC techniques. Introductory and in-depth texts are available, and we refer readers to a few of these (2;3;5;6;7).

Bayesian analysis is about estimation of parameters. For problems in technology assessment, we are interested in parameters such as the mean cost of a medical intervention, the probability that a patient will improve with a certain intervention, or the quality of life associated with a particular health state. These are the sorts of parameters whose values we estimate by collecting and analyzing relevant data. Bayesian analysis not only provides an estimate of the parameter, it also yields a probability distribution summarizing uncertainty about the value of the parameters given the observed data. Using these distributions we can compute probabilities of higher order hypotheses, such as the hypothesis that the value of one parameter is greater than another (e.g., the probability that an individual will improve with the experimental treatment is greater than the probability that an individual will improve with the standard treatment). In traditional, frequentist statistical analysis, such probabilities cannot be computed directly and the p value for a statistic is often used implicitly as a proxy (9).

Bayes' theorem specifies that the posterior density for a parameter given the observed data is:

$$f_1(\theta \mid d) = \frac{l(d \mid \theta)\, f_0(\theta)}{g(d)},$$

where θ is the parameter (or vector of parameters) that we are estimating and d represents the (vector of) observed data. Both θ and d are considered to be random variables. The function l is the *likelihood* of possible values of the data given a fixed value of the parameter; f_0 is the *prior*, the probability density function over all values of θ prior to observing the specific observed data; and g is the unconditional probability of the observed data (g is also known as the *marginal distribution of the data* or the prior predictive distribution). The function f_1 is the *posterior* density for the parameter conditional on the observed data. In Bayesian shorthand, the posterior is the prior times the likelihood normalized by the marginal of the data.

If the prior and the likelihood are natural conjugate distributions, then the posterior is in the same family of distributions as the prior and the problem has an easy analytical solution. If the prior and the likelihood are not conjugate distributions, it may be more difficult or even impossible to express the posterior distribution analytically. The function $g(d)$ is the integral of the product of the prior and likelihood functions with respect to θ. Luckily this function does not depend on θ and therefore can be treated as a constant when making inferences about θ. Bayes' theorem tells us that the posterior is proportional to the likelihood multiplied by the prior, but we do not know the exact constant of proportionality if we cannot compute $g(d)$.

Using MCMC methods, we do not need to know the constant. Gilks et al. (8) provide a brief overview of these methods:

MCMC is essentially Monte Carlo integration using Markov chains. Bayesians, and sometimes also frequentists, need to integrate over possibly high-dimensional probability distributions to make inference about model parameters or to make predictions. Bayesians need to integrate over the posterior distribution of the model parameters given the data, and frequentists may need to integrate over the distribution of observables given parameter values. As described below, *Monte Carlo* integration draws samples from the required distribution, and then forms sample averages to approximate expectations. *Markov chain* Monte Carlo draws these samples by running a cleverly constructed Markov chain for a long time. There are many ways of constructing these chains, but all of them, including the Gibbs sampler. . . , are special cases of the general framework of Metropolis et al. . . . and Hastings. . . . (8,1)

The general mechanics of these methods entail devising a sequence of numbers whereby the nth term of the sequence is a randomly sampled value from a probability distribution whose parameters are set using the $(n-1)$st term of the sequence. Each step in this sampling process corresponds to a possible value for θ. If the first term is a relatively arbitrary initial value θ_0, then this process results in a sequence of values, $\theta_0, \theta_1, \theta_2, \ldots, \theta_\mathrm{n}$. The number of terms in the sequence, n, is prespecified by the analyst, and efficient size for n is a subject of current research. Remarkably, the theorems state that the empirical distribution of the set of sampled values occurring after a certain point in the sequence, say term k, collectively converges to the posterior distribution, $f_1(\theta \mid d)$, as n grows suitably large, relative to k, and certain other conditions are satisfied. The first k terms of the sequence are called the "burn-in" terms and can be discarded. The remaining sequence then represents a random sample from the posterior distribution we wish to compute. Analysis of this set of numbers (e.g., computation of the mean, variance, quantiles, graphing the histogram of sampled values, etc.) allows us to empirically determine the posterior.

This process, obtaining the sequence of simulated values from the posterior distribution, is the heart of MCMC and modern Bayesian computation. The conditions that guarantee convergence of the sequence to the posterior distribution should not be taken lightly. In fact, much of the art of MCMC is judging whether the observed sequence of values has converged to the posterior, and whether the observed sequence is long enough to have sampled sufficiently from the full range of the posterior. Development of diagnostic statistics to inform these judgments is another area of active research. But in practice the convergence conditions are met for many problems with which we are concerned, and picking a value in the neighborhood of 1,000 for *k* and some tens of thousands for *n* seems to do the trick. We will not deal with the problem of diagnosing convergence in this article; readers are urged to familiarize themselves with these issues in the cited texts and other references cited at the BUGS Web site.

The beauty of WinBUGS (and its predecessor, BUGS) is that the likelihood function and the prior distribution are specified—even for multiple unknown parameters and functions of these parameters—in terms of a large gallery of standard probability distributions. The software will (usually) set up the MCMC process and conduct the sampling process so that the mechanics are largely transparent to the user. (Like any software package, WinBUGS is not error free. Occasionally models cannot be processed and confusing error messages are displayed. Reformulation of one's model or tips from the user and example manuals can overcome most problems and, as a last resort, questions can be sent to the BUGS e-mail discussion list where the generally quick responses are useful.) Because the technical details of the simulation process are "hidden," users can focus on model design issues. While WinBUGS performs the MCMC simulation behind the scenes, we again strongly recommend users gain familiarity with the MCMC process, and we point to the earlier cited texts as good sources.

EXAMPLES USING WINBUGS

We introduce the basic mechanics of the WinBUGS software and the graphical modeling process using two examples. The two examples we describe are frequently encountered in technology assessment. They can serve as building blocks for more complex problems, and the basic modeling steps presented are readily generalized. All information needed for the two examples is provided in the tutorial, but we do refer readers to the WinBUGS User Manual (12) for some steps.

For the first example, we estimate a probability using the beta-binomial model. This is a commonly encountered problem in clinical and health services research studies and, even though it may be solved analytically, it is useful to illustrate the WinBUGS modeling process. We begin by specifying this model graphically with a WinBUGS "Doodle." Next, we describe how to load the observed data and prepare the model for simulation. Finally, we show how to perform the analysis and review the results.

In our second example, we extend the first model and show how to estimate the expected value of a chance node in a decision tree and how to derive a posterior distribution describing uncertainty about this expectation. This second example will show how to use functions of parameters.

Downloading and Installing WinBUGS

The WinBUGS version 1.2 installation files and documentation are located at the BUGS Project Web site (10). Documentation includes a user manual, along with 35 complex examples. When the software is downloaded and the key for unrestricted use is installed, the user is offered registration for the BUGS e-mail discussion list. We have found this mailing list to be an educational forum where novice and experienced users ask questions and exchange ideas on problems with BUGS and WinBUGS software, and more generally with Bayesian modeling issues.

Example 1: Estimating a Probability

To illustrate the basic features and functionality of WinBUGS, we will step through the WinBUGS modeling process for a simple beta-binomial model for estimating a probability. In this example, the posterior distribution can be calculated analytically; nonetheless, we will simulate it using WinBUGS.

As the hypothetical setting for the example, suppose we wish to estimate the probability, θ, that a patient undergoing a particular treatment will experience a certain complication. We observe 41 patients who each received a particular medical treatment; 16 in the case sequence experience the complication, and the remaining 25 do not. There are two equivalent ways to represent these data. First, we could consider this to be one observation of a binomial random variable with parameters (θ, N), where $N = 41$ and θ is our unknown probability of a complication. The observed value of the variable is 16. Alternatively, the binomial distribution can be thought of as the sum of independent Bernoulli trials, each taking the value 0 (no complication) or 1 (complication) with probability θ that a particular trial results in 1. We will use the Bernoulli representation because it illustrates more features and functionality in WinBUGS—in particular, it shows how to input and analyze data constituting a sample of independent observations of a random variable. Our prior uncertainty about the true value of θ is modeled using a beta distribution. Thus, in our previous notation, the likelihood function for each data point, $l(d \mid \theta)$ is a Bernoulli density, depending on θ, and the prior probability function $f_0(\theta)$ is a beta density.

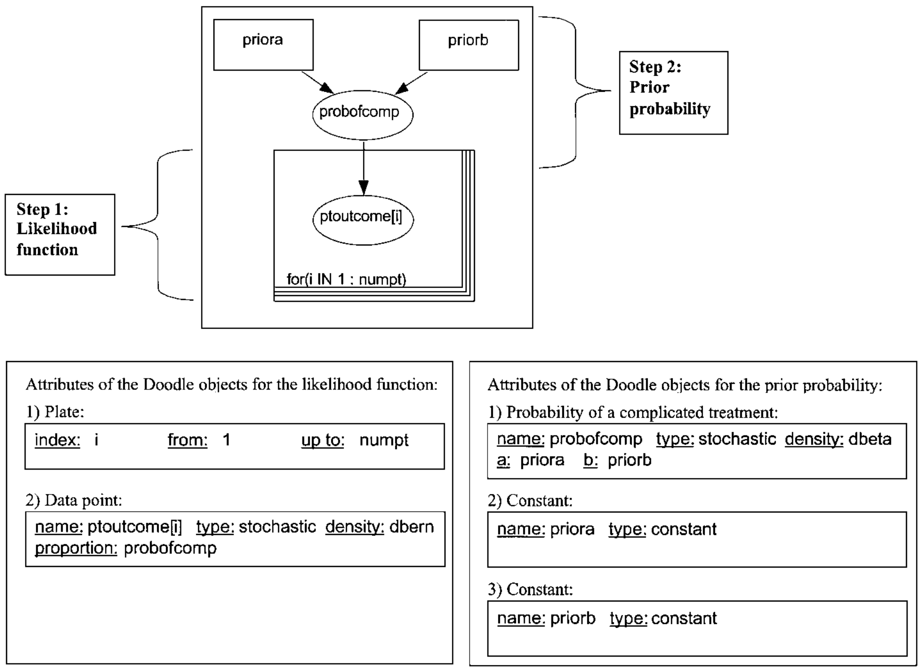
The steps in a Bayesian analysis using WinBUGS are: a) create a "Doodle" to specify the model; b) prepare the model for simulation; and c) perform the analysis and review the results. The first step is to specify the model in WinBUGS by creating a "Doodle." A Doodle is a pictorial representation of the model. It contains the information about the likelihood functions and prior probabilities that WinBUGS uses in combination with the observed data to perform MCMC simulations. A Doodle is constructed with the following three graphical objects:

1. A *node* is used to represent parameters, data, or other variables in the model and is shown on the screen as an oval object. All nodes have two main attributes: name and type. The three types of nodes are stochastic, constant, and logical. A stochastic node has additional attributes defining the associated probability distribution and parameters controlling that distribution. A constant node represents a numerical constant where the value of the constant is specified with the data rather than in the Doodle. A logical node is an algebraic function of one or more other nodes in the Doodle. If a logical node is a function of stochastic nodes, it inherits an induced probability distribution from this relationship.
2. A *plate* is used to denote repetition of identical nodes such as data observations. A plate is represented visually as a stack of rectangular cards, one for each data point. It has attributes that indicate the index to the data that is used, such as i or j, and the number of repetitions or data points that the plate represents.
3. A *link* is used to indicate dependencies between nodes. It is represented in the doodle as a unidirectional arrow from one node to another, the direction of the arrow defining the dependency between nodes.

Appendix 1 provides a quick reference for Doodle editing commands; readers are referred to the WinBUGS User Manual and the Doodle Manual distributed on-line with the software for more detailed information.

In the Doodle, nodes may be arranged in a hierarchical fashion for clarity. Figure 1 shows the Doodle for our beta-Bernoulli model and outlines the following steps we used to create it. We start by representing the observed data. We use a plate to indicate that we have 41 identically distributed data observations. When specifying the attributes of the plate, we

https://doi.org/10.1017/S0266462301104095 Published online by Cambridge University Press



Attributes of the Doodle objects for the likelihood function:

1) Plate:

index: i from: 1 up to: numpt

2) Data point:

name: ptoutcome[i] type: stochastic density: dbern
proportion: probofcomp

Attributes of the Doodle objects for the prior probability:

1) Probability of a complicated treatment:

name: probofcomp type: stochastic density: dbeta
a: priora b: priorb

2) Constant:

name: priora type: constant

3) Constant:

name: priorb type: constant

Figure 1. Complete Doodle for the beta-Bernoulli model to estimate a probability. The upper portion shows the Doodle for the beta-Bernoulli model for estimating a probability. The lower portion presents the attributes of the Doodle objects used to create the Doodle. (See Appendix 1 for specific Doodle object editing commands.) The likelihood function is represented by the oval-shaped stochastic node placed inside the rectangular plate. We specified a Bernoulli distribution as the density function for this stochastic node. The probability of a complicated treatment course is also represented by a stochastic node to which we assign a beta distribution. The two rectangular nodes are the constant parameters of the beta distribution discussed in the text.

use a placeholder for the number of data points and call it *numpt* for number of patients. By using a placeholder instead of entering the numerical value directly in the Doodle, the Doodle is made more general. The numerical value of the placeholder will be entered later in a data statement.

Within the plate, to represent the "*i*th" data point, we use a stochastic node. The default node type is stochastic, so this attribute does not need to be edited for the node we just created. We call this node *ptoutcome*[i] where "i" is the index to the data. (Nodes are very easy to create with a spurious click of the mouse and appear suddenly, seemingly at whim, in one's Doodle. In fact, the keystroke for deleting a node, ctrl-backspace or ctrl-delete, is quite possibly one of the most useful pieces of information presented in this tutorial.) Each data point is a 1 or 0 to indicate whether a patient had the complication. The likelihood function for each data point is Bernoulli. We have defined the node as Bernoulli by selecting the type, "dbern," from the dropdown menu of distributions that appears when we click on "density" in the node definition fields at the top of the Doodle window. Distributions are named "dxxxx" on this menu, where "xxxx" is a short abbreviation of the full name of the distribution. We will not go through all steps mechanistically in this text, but refer the reader to Appendix 1 where the steps are depicted. The Bernoulli distribution has one parameter, the probability that the observation takes the value 1. For our problem, this is the parameter we wish to estimate: θ, the probability of the complication.

We next use a node to represent θ and name it *probofcomp*. The Bernoulli likelihood function for our observed data depends on this parameter. To indicate this dependency, we create a “link” between the *ptoutcome* [i] node and the *probofcomp* node, where the direction of the link indicates the dependency (see Appendix 1).

The probability of complication is a random variable so we make the node, *probofcomp*, a stochastic node. We specified our prior uncertainty about this probability using a beta distribution. The beta distribution is a function of two parameters, $a > 0$ and $b > 0$; its mean, our best point estimate of θ, is $a/(a + b)$. Suppose we do not have any information or ideas about the likely values of θ before observing the data. In this case we use a noninformative beta distribution as our prior probability. Several methods exist for determining noninformative priors; however, there is no consensus as to the best method (1, 73-74). Since the beta distribution contains less and less information about θ as the two parameters simultaneously approach zero, we specify the two parameters of the beta distribution as each having value of 0.01. We could have chosen a different noninformative prior instead, such as the one that specifies both parameters of the beta distribution as having values of 1 (this beta distribution is the uniform distribution), but this specification has more impact on the posterior than does a specification with smaller parameters. In the Doodle, we use placeholders for the numerical values of these parameters. Like the placeholder for the number of data points, we will specify the values of these two parameters (and hence for our prior distribution on θ) in the data statement. Again, by using placeholders instead of entering the numerical values directly in the Doodle, it is made more general for future use.

We use two nodes to represent the two placeholders. We assign these nodes, called *priora* and *priorb*, to be constants in this problem. To indicate that the probability of a complicated treatment course depends on these two parameters, we create links from the two constant nodes, *priora* and *priorb*, to *probofcomp*. The order in which they are linked when the Doodle is interactively developed by the user determines which constant node links to which of the two parameters of the beta distribution for the *probofcomp* node.

This completes the steps for constructing a simple Doodle for the analysis model. A helpful feature is that Doodles can be copied, pasted, and edited in other WinBUGS text windows or other application windows, such as a Microsoft Word document. In WinBUGS terminology, documents containing both Doodles and text are called “compound documents.”

Once the Doodle is saved, the next steps prepare the model for simulation. To prepare the model for simulation, we have WinBUGS check the model syntax, then we enter the data, compile the model, and finally specify initial values (the starting values of the MCMC sequence) for the simulation. Appendix 2 describes these steps and discusses common error messages. The process to prepare the model is not complex, but the order of the steps is important.

Data are entered in a very specific format that is derived from the S+ statistical language. We found the example manuals that provide complete data statements for all of the examples to be helpful guides for learning the syntax. A text file, rather than a Doodle file, is used to enter data. Figure 2 shows the complete data statement for our model. We specify our observed data as well as the numerical values of the placeholders, such as *numpt*, that we created in the Doodle. Data are entered in a comma-delimited string in the parentheses. In the file the start of the data statement is indicated by the heading “data,”. The data values themselves are entered parenthetically in a comma-delimited list after the word “list” in the text file. To create the list of data, the format is: variable name = value for a single-valued variable such as a 1. For multiple-valued variables such as *ptoutcome* that describes our observed data, a simple list format can be used. The format for a list is: variable name = c(value1, value2, . . .). For multiple-valued variables with a lot of data, this list format is cumbersome. The user manual describes the syntax for entering data in vector notation as well as the syntax for higher dimensional arrays.

```
Data;
list(ptoutcome=c(1,0,0,0,1,0,0,0,1,1,1,0,0,1,0,0,0,1,0,1,1,0,0,0,1,0,0,0,1,0,0,
0,0,0,1,1,1,0,0,1,1),
priora=.1, priorb=.1, numpt=41)



Inits;
list(probofcomp=.5)
```

Figure 2. The data statement (top panel) for the beta-Bernoulli example contains the observed data, shown in list form, as well as data for parameters we created in the model. The lower panel is the initial value statement for the beta-Bernoulli example. We provide an initial or starting value to start the Markov chain for each parameter that WinBUGS will simulate. The syntax of the data statement and the initial value statement is explained in the WinBUGS User Manual.

After loading the data and compiling the model, we must provide initial values for the variables that will be simulated. "Initial values" are not "prior" information. An initial value is the starting value for the Markov chain, the numerical value for θ_0 in the sequence θ_0, $\theta_1, \theta_2, \ldots, \theta_{\mathrm{n}}$ described earlier. An initial value must be specified for each stochastic node except those representing observed data (the values of these latter nodes—*ptoutcome*[i] in the current example—are specified in the data statement where the data are input). The choice of initial value generally only affects the rate of convergence of the simulation to the posterior distribution; the farther away from the range covered by the posterior distribution, the larger the value of k for the critical θ_{k} where the sequence begins to converge on the posterior distribution. Sometimes a choice of initial value that is too far from the feasible range for θ may crash the program since it may produce a random draw for some θ_i that is arithmetically unfortunate for the computer code that WinBUGS has produced, but generally any value for θ_0 in a distantly possible range for the parameter will serve as a feasible initial value. Figure 2 also shows the completed initial value statement for this model. The format for the initial value statement is the same as the data statement.

Once the model is compiled and initialized, we are ready to perform the analysis and review the results. Appendix 3 provides a road map for these steps. We specify to WinBUGS which nodes we wish to assemble results for, how many simulated values to discard at the beginning of the sequence so we are relatively sure the sequence has converged by this point (our guess at k for θ_{k}), and how many simulations to perform (i.e., a value for n in θ_{n}). The only drawback we see to specifying quite large values for k and n is computing time. The larger n-k, the smoother the simulated posterior distribution appears. For our beta-Bernoulli model, WinBUGS simulates a posterior distribution for the probability, *probofcomp*. For this very simple model, $k = 1{,}000$ and $n = 11{,}000$ requires at most a few seconds time on a 266-MHz computer and produces a very nice result. (In WinBUGS, the beginning iteration number to include in the results, $k + 1$ or 1,001, is entered rather than the number to exclude, k.) In general, the number of iterations necessary for the simulation to converge to the posterior distribution will depend on the complexity of the model and the adequacy of the initial values; WinBUGS provides several diagnostic tools to assess convergence. The WinBUGS User Manual describes these tools, and the suggested references provide more detailed discussions of issues in convergence of MCMC simulations.

We have set *probofcomp* as a variable for which WinBUGS will store sequentially sampled values in the MCMC process. This is done under "samples" on the WinBUGS "Inference" menu. WinBUGS allows us to exclude the "burn-in" iterations on the presumption that the simulation algorithm converges to the posterior density by this point. The number we specified, 1,000, was chosen more or less arbitrarily since convergence should be very quick in this simple example, and 1,000 is a typical number in the literature. In

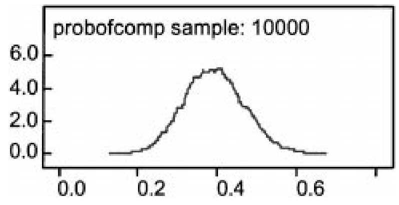
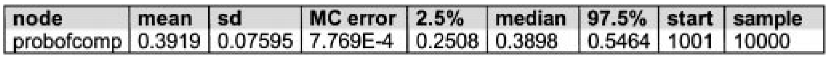


Figure 3. Posterior density plot for the beta-Bernoulli model to estimate the probability of a complicated treatment as estimated by WinBUGS. This plot shows the empirical posterior density for the probability of a complicated treatment. The plot was created from a simulation run of 11,000 iterations, excluding the first 1,000 as burn-in. As more iterations are performed, the distribution would become smoother.



node	mean	sd	MC error	2.5%	median	97.5%	start	sample
probofcomp	0.3919	0.07595	7.769E-4	0.2508	0.3898	0.5464	1001	10000

Figure 4. Sample statistics for the probability of a complicated treatment as estimated by WinBUGS. This facsimile of the WinBUGS statistics window summarizes output for the probability of a complicated treatment based on 11,000 iterations of the MCMC simulation (excluding the first 1,000). The default statistics WinBUGS computes are the mean, standard deviation, median, and 2.5th and 97.5th percentiles. Other percentiles may be selected by the user.

more complicated problems, a larger burn-in sequence, along with attention to some of the convergence diagnostics, is wise. Figure 3 shows a plot after 11,000 iterations (the plot excludes the first 1,000 as burn-in) of the simulated posterior density for *probofcomp*. From the plot, we see the range of possible values for *probofcomp* and which values are more likely than others. WinBUGS' "trace" and "history" functions also offer serial plots of the actual sequence of simulated *probofcomp* values for the purpose of diagnosing convergence and mixing of the sequence.

Figure 4 shows sample statistics from the simulated posterior distribution. WinBUGS calculates these from the stored sequence of simulated values of *probofcomp*. The default statistics are the mean, standard deviation, median, and the 2.5 and 97.5 percentiles. Additional statistics are available. WinBUGS will save the entire sequence of simulated values in a text file (termed a coda file) that can be further analyzed using other statistical analysis programs or spreadsheet programs. In the *probofcomp* example, we conclude the point estimate is *probofcomp* = 0.3919 (the mean value), there is a 0.5 probability that *probofcomp* is greater than 0.3898 (the median), and that a 95% credible interval for the value of *probofcomp* is 0.2508 to 0.5464. Since all simulated values (past the 1,000 discarded as burn-in for the sequence) can be saved, we can sort these and use them to derive a full empirical distribution function for *probofcomp*, which can be used for hypothesis testing, for stochastic sensitivity analysis in further calculations involving *probofcomp*, or for other analyses.

Example 2: Estimating an Expected Value in a Decision Tree

Using the beta-Bernoulli model of Example 1 as a building block, we now use WinBUGS to simulate a posterior distribution for the expected value at a chance node in a decision tree. The WinBUGS modeling steps for this model are the same as the previous example. This example can be extended to evaluate a more complicated decision tree in WinBUGS.

As in the previous example, suppose we have observed the results of a medical treatment in 41 patients. Further, we observe the total costs of care consumed by each patient in the 12 months following the treatment and are interested in estimating the expected cost of treatment. Figure 5 shows the chance node from a decision tree diagramming this example.

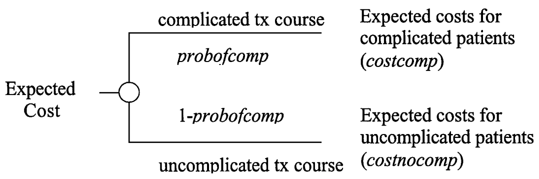
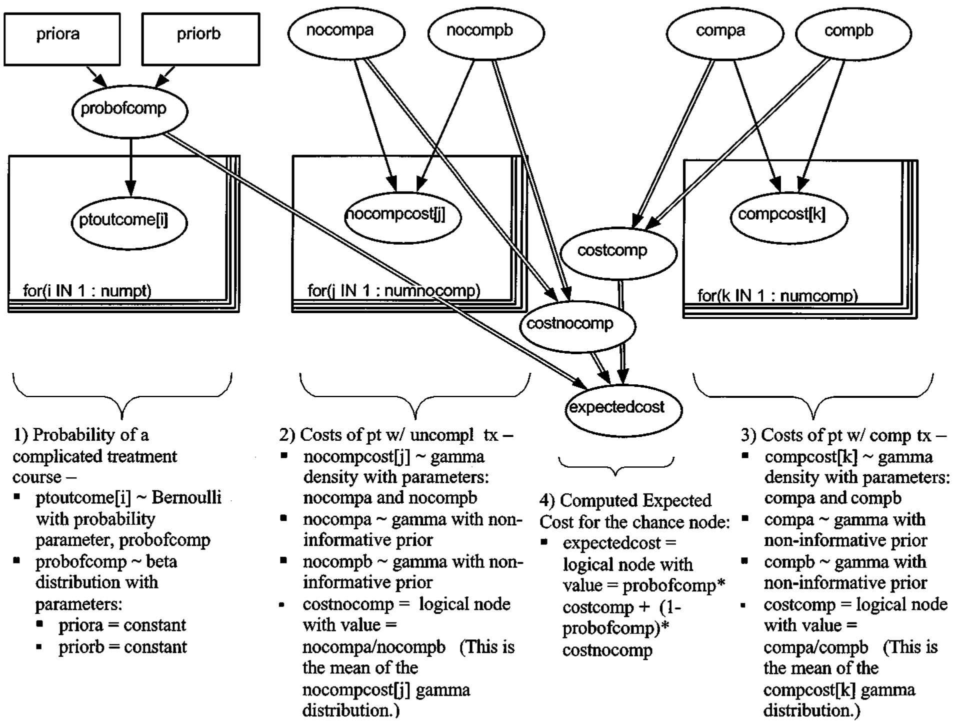

complicated tx course
probofcomp
Expected Cost
1-probofcomp
uncomplicated tx course
Expected costs for complicated patients (costcomp)
Expected costs for uncomplicated patients (costnocomp)

Figure 5. Chance node from a decision tree for the cost of medical treatment. The expected cost of a medical treatment is represented here as a chance node fragment from a decision tree. A patient receiving the treatment can experience either a complicated or uncomplicated treatment course.



1) Probability of a complicated treatment course –

- ptoutcome[i] ~ Bernoulli with probability parameter, probofcomp
- probofcomp ~ beta distribution with parameters:
  - priora = constant
  - priorb = constant

2) Costs of pt w/ uncompl tx –

- nocompcost[j] ~ gamma density with parameters: nocompa and nocompb
- nocompa ~ gamma with non-informative prior
- nocompb ~ gamma with non-informative prior
- costnocomp = logical node with value = nocompa/nocompb (This is the mean of the nocompcost[j] gamma distribution.)

4) Computed Expected Cost for the chance node:

- expectedcost = logical node with value = probofcomp* costcomp + (1-probofcomp)* costnocomp

3) Costs of pt w/ comp tx –

- compcost[k] ~ gamma density with parameters: compa and compb
- compa ~ gamma with non-informative prior
- compb ~ gamma with non-informative prior
- costcomp = logical node with value = compa/compb (This is the mean of the compcost[k] gamma distribution.)

Figure 6. This Doodle models the estimation of the expected cost of treatment. In essence, it is three separate models joined together with logical nodes. The left portion is the model for estimating a probability, also shown in Figure 1. The right portion estimates expected costs for complicated and uncomplicated treatment courses separately.

(Although potentially confusing, we use the conventional terminology for a branching point in a decision tree, a “node,” as well as the syntax adopted in WinBUGS for a parameter in a Doodle, also known as a “node.” Our meaning should be clear from context.)

To compute the expected cost of treatment, we need the probability of a complicated treatment course, *probofcomp*, the expected cost of a complicated treatment course, *cost-comp*, and the expected cost of an uncomplicated treatment course, *costnocomp*. As in the previous example, we use the treatment outcome data to estimate the probability of a complicated treatment course. In addition, we use the cost data to separately estimate the expected cost of treatment for both patient groups (i.e., those experiencing complicated and uncomplicated treatment courses). Figure 6 shows the completed Doodle for this problem and outlines the steps used to create it. The left portion of the Doodle in Figure 6 is the previous Doodle we constructed for estimating the probability of a complicated course of treatment. The right portion of the Doodle contains two plates for modeling the cost data for patients who experienced complicated and uncomplicated treatment courses and estimating the expected cost for each patient group.

We model the likelihood of the observed cost data for patients experiencing an uncomplicated treatment course with a gamma density. The gamma density is useful as a model of random quantities whose values are greater than zero and which have a skewed distribution with long tail to the right which is typical of cost data. WinBUGS has implemented the gamma density as a function of two parameters, $a > 0$ and $b > 0$ with a mean a/b, and variance a/b^2. Other software programs such as Microsoft Excel and Minitab (Minitab Inc., State College, PA) parameterize this distribution in such a way that the mean is *ab* and variance ab^2; users of software packages employing the gamma distribution should be cautions when determining how it is parameterized. Since the two parameters, a and b, each must be greater than zero, our prior uncertainty about these two parameters of the gamma density is modeled using gamma densities as well. WinBUGS provides values of 0.001 for both parameters to specify a noninformative gamma density. (Noninformative parameter values are also provided as defaults for several other densities, such as the normal density.) We will use these default values for our prior uncertainty about the two parameters; these will be updated in light of the observed cost data. If we had stronger opinions or other data about the likely costs, we would use prior values reflecting these opinions and data. We label the node for the observed cost data among patients without the complication as *nocompcost*, indexed by j, and the two parameters of the gamma likelihood function, *nocompa* and *nocompb*.

To estimate the expected cost for patients who experienced an uncomplicated treatment course, we use a logical node that we label *costnocomp*. The expected (i.e., mean) cost is the mean of the gamma density used to model the likelihood of the cost data. The mean of the gamma density is the ratio of its two parameters, so we set the value of the logical node equal to the ratio *nocompa/nocompb*. To indicate the dependency between the logical node, *costnocomp*, and the two parameters of the gamma density representing the likelihood function, we link the node *costnocomp* to the nodes *nocompa* and *nocompb*. WinBUGS represents links to logical nodes by double-lined arrows instead of the single-lined arrow that joins stochastic nodes. The expected cost for patients who experienced a complicated treatment course, *costcomp*, is modeled similarly.

Figure 7 shows the data and initial value statements for this model. In this model, we provide initial values for the nodes that WinBUGS is simulating. The logical nodes such as *costcomp* and *costnocomp* do not need to be initialized because the posterior distributions for these nodes are induced from other simulated posterior distributions.

```
Data;
list(ptoutcome=c(1,0,0,0,1,0,0,0,1,1,1,0,0,1,0,0,0,1,0,1,1,0,0,0,1,0,0,0,1,0,0,
0,0,0,1,1,1,0,0,1,1),
nocompcost=c(2508, 199, 624, 3828, 3758, 364, 4685, 2554, 548, 8153,
9, 425, 1029, 584, 2156, 2451, 462, 1896, 7613, 1300, 672,1092, 36,
2765, 4237),
compcost=c(9582, 6875, 12191, 8287, 17584, 6172, 8953, 11401, 9408,
14216, 12218, 5599, 9260, 9068, 7511, 15461),
numnocomp=25, numcomp=16,
numpt=41,
priora=.25, priorb=.25)


Inits;
list(probofcomp=.1, nocompa=.1, nocompb=.1, compa=.1, compb=.1)
```

Figure 7. The data and initial value statements for the model estimating the expected cost of treatment are shown. Initial values are provided for all stochastic nodes for which posterior distributions are simulated, except for the logical nodes.

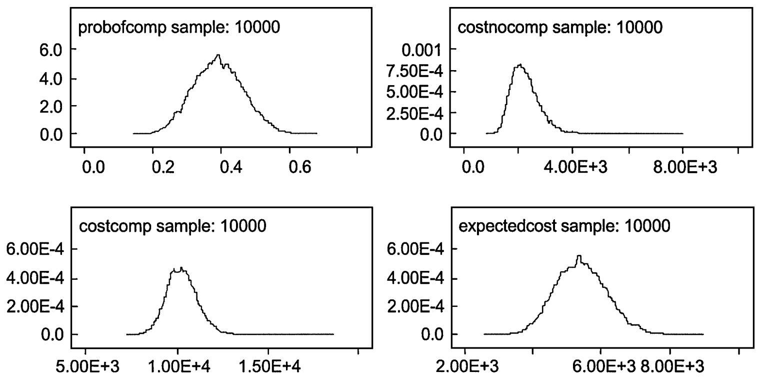
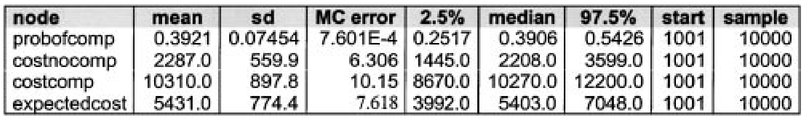


Figure 8. WinBUGS plots of the posterior densities for four of the stochastic nodes in Figure 6. The top left plot is the empirical density for the probability of a complicated treatment and is equivalent to the plot in Figure 4. The remaining three plots show the empirical posterior distributions for the expected cost associated with an uncomplicated treatment course, the expected cost associated with a complicated treatment course, and the expected cost averaged over treatment courses. Each plot is calculated from 11,000 iterations of the MCMC simulation and excludes the first 1,000 iterations as burn-in.



node	mean	sd	MC error	2.5%	median	97.5%	start	sample
probofcomp	0.3921	0.07454	7.601E-4	0.2517	0.3906	0.5426	1001	10000
costnocomp	2287.0	559.9	6.306	1445.0	2208.0	3599.0	1001	10000
costcomp	10310.0	897.8	10.15	8670.0	10270.0	12200.0	1001	10000
expectedcost	5431.0	774.4	7.618	3992.0	5403.0	7048.0	1001	10000

Figure 9. Sample statistics for the parameters estimated in the expected cost model. Facsimile of WinBUGS output window summarizing sample statistics for the problem specified in Figures 6 and 7.

Figures 8 and 9 show the plot of the simulated posterior distribution for the expected cost at the chance node in the decision tree, and the sample statistics for this random quantity, respectively. Similar to the previous example, we ran the simulation for 11,000 iterations and excluded the first 1,000 as “burn-in.” WinBUGS provides results for the other nodes in the Doodle of Figure 6, and these are shown as well in Figure 9.

CONCLUSION

The WinBUGS software is a useful and powerful tool for Bayesian analysis. It allows users to evaluate complex models without implementing the technical details of MCMC for each new problem. Then users are able to focus on model design and validation. The Doodle representation provides an intuitive method for graphically conceptualizing models and translating them to the software. The full sequence of simulated values for each node in a Doodle are easily portable to other software, such as Excel or Minitab, for further analysis by saving the WinBUGS “coda” output as a text file.

This tutorial was an introduction to the WinBUGS software. We presented two examples common to medical decision making. The first example, the beta-Bernoulli, estimated the probability of a complicated treatment course. The second example built on the beta-Bernoulli and evaluated the expected cost of a treatment as it might be represented in a decision tree or a cost-effectiveness analysis for technology assessment.

When one considers technology assessment problems more generally, these pieces, estimating proportions, costs, and expectations formed from them, are commonly encountered. Other common parameters in technology assessments are quality-of-life weights for health states and the various statistics representing survival or recurrence times. All of these may be addressed individually as well as combined to compute more involved outputs of technology assessment, such as cost per quality-adjusted life-year saved or net health benefits (13). Bayesian analysis of these outputs is a natural manner in which to represent the uncertainties inherent in technology assessment models. Software such as WinBUGS and its successors can be of great assistance in helping to diffuse the Bayesian approach as a common and useful tool for technology assessment. The collaborators behind WinBUGS at MRC Cambridge Biostatistics Unit and at Imperial College are providing a commendable service to the field of Bayesian analysis as well as the applied fields that need it, such as healthcare technology assessment.

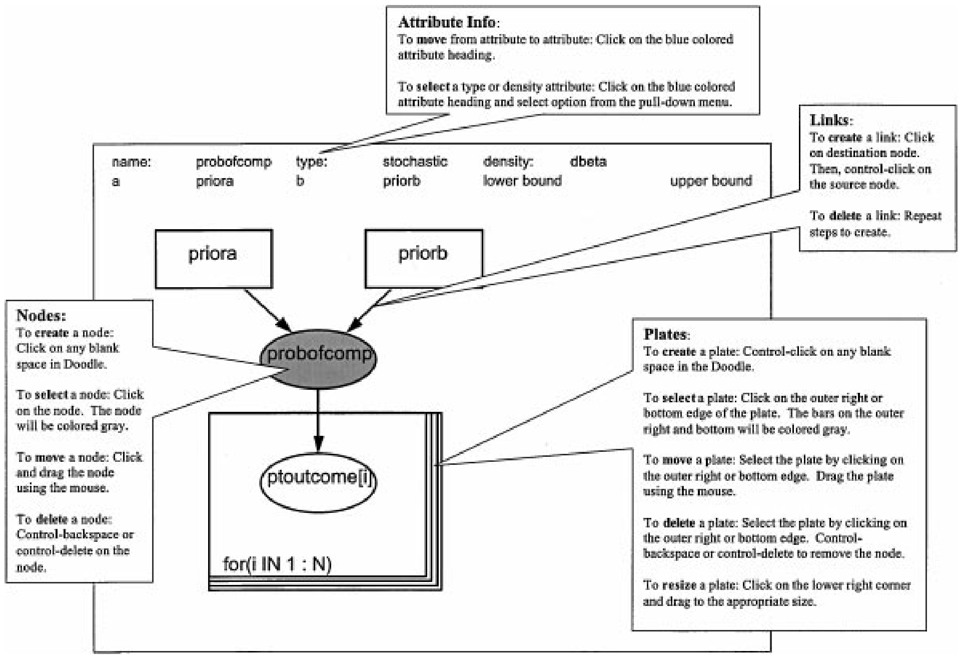
But, as the developers warn, the availability of the software does not absolve users of the need to learn about issues in Bayesian modeling and problems encountered in MCMC methods (particularly the problem of convergence of the MCMC simulation to the desired posterior distribution). Learning to use the software as it grows through its developmental stages and also learning or relearning Bayesian analysis can seem daunting. We hope that this introduction encourages others to take on these tasks to gain entry to powerful new tools for technology assessment.

REFERENCES

1. Berger JO. *Statistical decision theory*: *Foundations, Concepts, and Methods*. New York: Springer-Verlag; 1980.
2. Berry D. *Statistics: A Bayesian perspective*. Belmont, CA: Duxbury Press; 1996.
3. Chen M-H, Shao Q-M, Ibrahim JG. *Monte Carlo methods in Bayesian computation*. New York: Springer-Verlag; 2000.
4. Claxton K. The irrelevance of inference: A decision-making approach to the stochastic evaluation of health care technologies. *J Health Econ*. 1999;18:341-364.
5. Gamerman D. *Markov chain Monte Carlo: Stochastic simulation for Bayesian inference*. London: Chapman & Hall; 1997.
6. Gelman A, Carlin JB, Stern HS, et al. *Bayesian data analysis*. London: Chapman & Hall; 1995.
7. Gilks WR, Richardson S, Spiegelhalter DG. *Markov chain Monte Carlo in practice*. London: Chapman & Hall; 1996:486.
8. Gilks WR, Richardson S, Spiegelhalter DG. Introducing Markov chain Monte Carlo. In: *Markov chain Monte Carlo in Practice*, London: Chapman & Hall; 1996:1-20.
9. Goodman SN. Toward evidence-based medical statistics, I: The P value fallacy. *Ann Intern Med*. 1999;130:995-1004.
10. MRC Biostatistics Unit, Cambridge, UK. The BUGS Project. Available at: http://www.mrc-bsu.cam.ac.uk/bugs/. Accessed February 25, 2000.
11. Scollnik DPM. Actuarial modeling with MCMC and BUGS. Available at: http://www.math.ucalgary.ca/∼scollnik/abcd/. Accessed February 25, 2000.
12. Spiegelhalter D, Thomas A, Best N. WinBUGS version 1.2 user manual. Available at: http://www.mrc-bsu.cam.ac.uk/bugs/. Accessed February 25, 2000.
13. Stinnett AA, Mullahy J. Net health benefits: A new framework for the analysis of uncertainty in cost-effectiveness analysis. *Medical Decision Making*. 1998;18(suppl 2):S68-S80.

APPENDIX 1

Doodle Quick Reference



name: probofcomp type: stochastic density: dbeta
a priora b priorb lower bound upper bound

Attribute Info:
To **move** from attribute to attribute: Click on the blue colored attribute heading.

To **select** a type or density attribute: Click on the blue colored attribute heading and select option from the pull-down menu.

Links:
To **create** a link: Click on destination node. Then, control-click on the source node.

To **delete** a link: Repeat steps to create.

Nodes:
To **create** a node: Click on any blank space in Doodle.

To **select** a node: Click on the node. The node will be colored gray.

To **move** a node: Click and drag the node using the mouse.

To **delete** a node: Control-backspace or control-delete on the node.

Plates:
To **create** a plate: Control-click on any blank space in the Doodle.

To **select** a plate: Click on the outer right or bottom edge of the plate. The bars on the outer right and bottom will be colored gray.

To **move** a plate: Select the plate by clicking on the outer right or bottom edge. Drag the plate using the mouse.

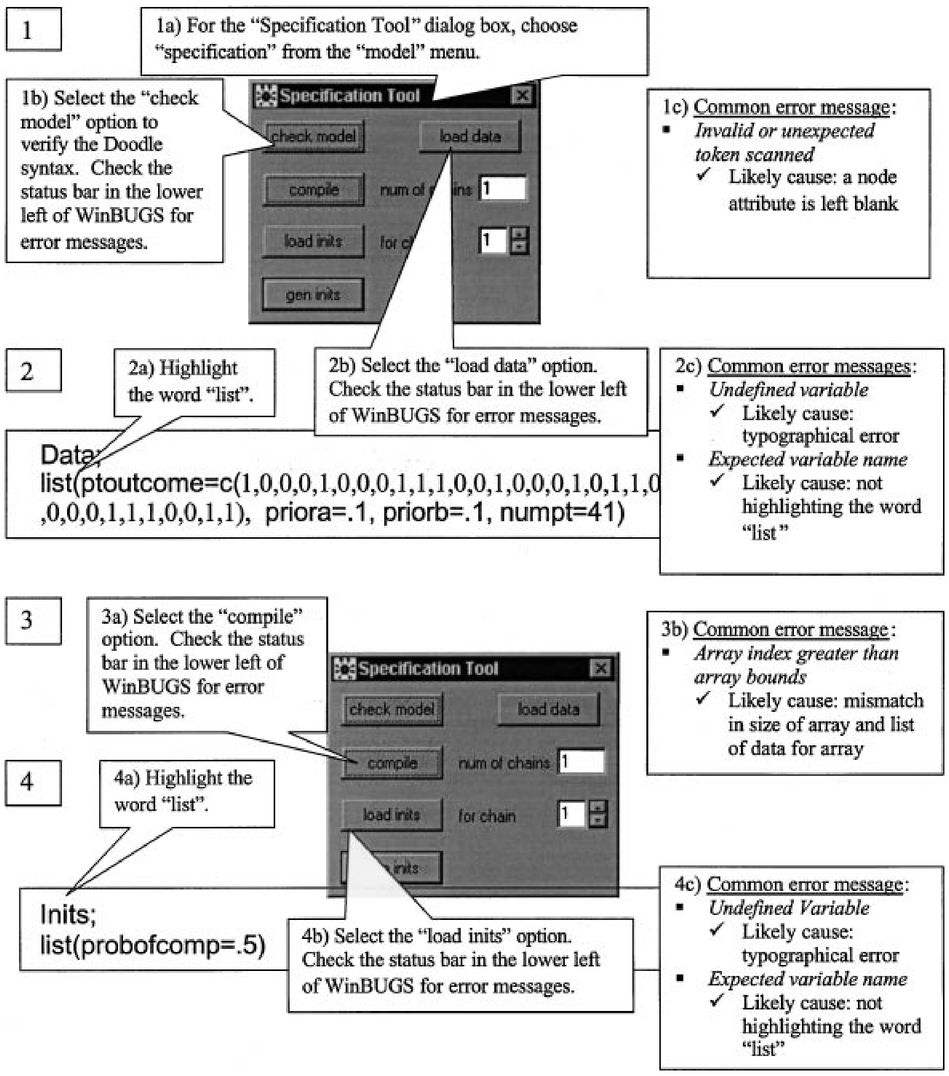
To **delete** a plate: Select the plate by clicking on the outer right or bottom edge. Control-backspace or control-delete to remove the node.

To **resize** a plate: Click on the lower right corner and drag to the appropriate size.

https://doi.org/10.1017/S0266462301104095 Published online by Cambridge University Press

APPENDIX 2

Quick Reference Sheet for Preparing the Model



1

1a) For the "Specification Tool" dialog box, choose "specification" from the "model" menu.

1b) Select the "check model" option to verify the Doodle syntax. Check the status bar in the lower left of WinBUGS for error messages.

1c) Common error message:
- *Invalid or unexpected token scanned*
  - ✓ Likely cause: a node attribute is left blank

2

2a) Highlight the word "list".

2b) Select the "load data" option. Check the status bar in the lower left of WinBUGS for error messages.

Data;
list(ptoutcome=c(1,0,0,0,1,0,0,0,1,1,1,0,0,1,0,0,0,1,0,1,1,0
,0,0,0,1,1,1,0,0,1,1), priora=.1, priorb=.1, numpt=41)

2c) Common error messages:
- *Undefined variable*
  - ✓ Likely cause: typographical error
- *Expected variable name*
  - ✓ Likely cause: not highlighting the word "list"

3

3a) Select the "compile" option. Check the status bar in the lower left of WinBUGS for error messages.

3b) Common error message:
- *Array index greater than array bounds*
  - ✓ Likely cause: mismatch in size of array and list of data for array

4

4a) Highlight the word "list".

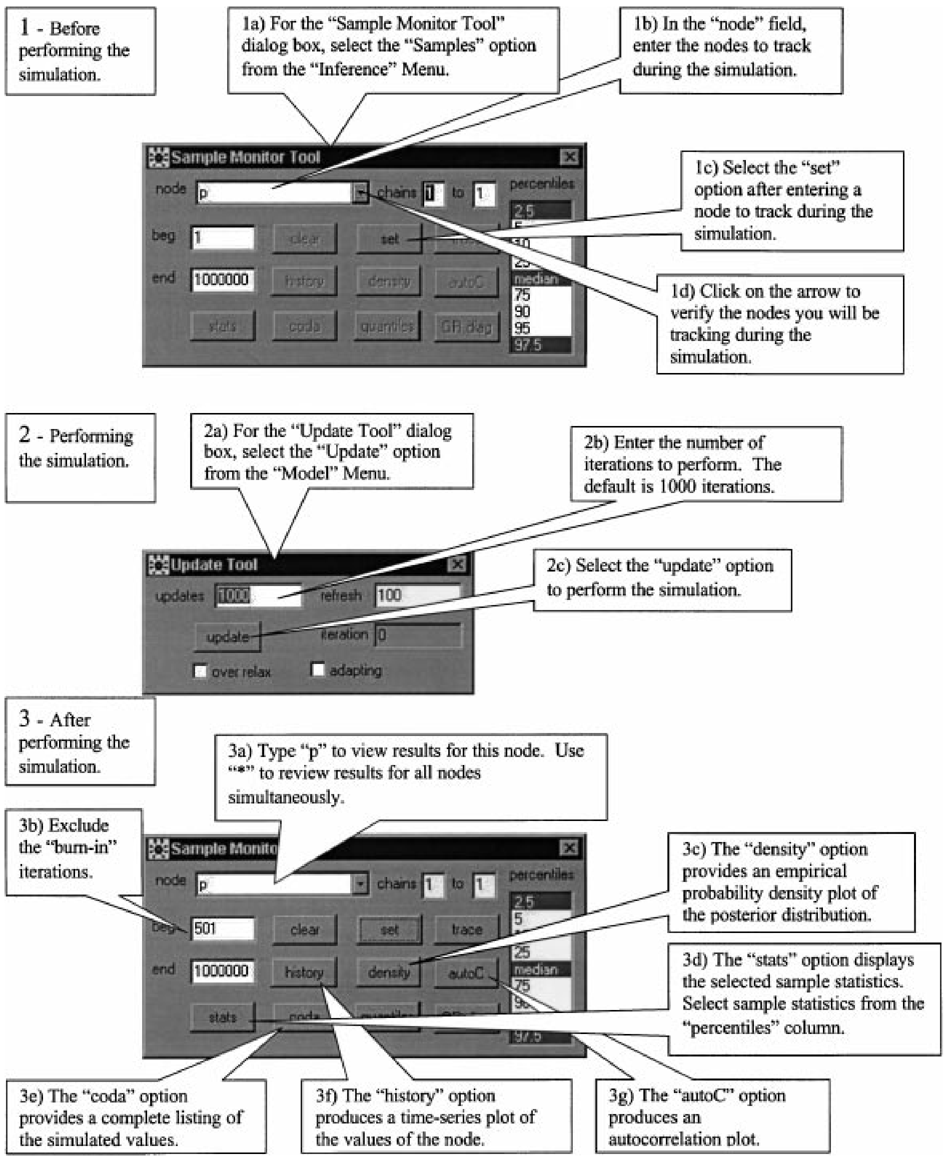
Inits;
list(probofcomp=.5)

4b) Select the "load inits" option. Check the status bar in the lower left of WinBUGS for error messages.

4c) Common error message:
- *Undefined Variable*
  - ✓ Likely cause: typographical error
- *Expected variable name*
  - ✓ Likely cause: not highlighting the word "list"

APPENDIX 3

Quick Reference Sheet for Performing the Analysis and Reviewing the Results



1 - Before performing the simulation.

1a) For the "Sample Monitor Tool" dialog box, select the "Samples" option from the "Inference" Menu.

1b) In the "node" field, enter the nodes to track during the simulation.

1c) Select the "set" option after entering a node to track during the simulation.

1d) Click on the arrow to verify the nodes you will be tracking during the simulation.

Sample Monitor Tool: node p; chains 1 to 1; beg 1; end 1000000

2 - Performing the simulation.

2a) For the "Update Tool" dialog box, select the "Update" option from the "Model" Menu.

2b) Enter the number of iterations to perform. The default is 1000 iterations.

2c) Select the "update" option to perform the simulation.

Update Tool: updates 1000; refresh 100; iteration 0

3 - After performing the simulation.

3a) Type "p" to view results for this node. Use "*" to review results for all nodes simultaneously.

3b) Exclude the "burn-in" iterations.

3c) The "density" option provides an empirical probability density plot of the posterior distribution.

3d) The "stats" option displays the selected sample statistics. Select sample statistics from the "percentiles" column.

3e) The "coda" option provides a complete listing of the simulated values.

3f) The "history" option produces a time-series plot of the values of the node.

3g) The "autoC" option produces an autocorrelation plot.

Sample Monitor Tool: node p; chains 1 to 1; beg 501; end 1000000