

ACTUARIAL MODELING WITH MCMC AND BUGS

David P. M. Scollnik*

ABSTRACT

In this paper, the author reviews some aspects of Bayesian data analysis and discusses how a variety of actuarial models can be implemented and analyzed in accordance with the Bayesian paradigm using Markov chain Monte Carlo techniques via the BUGS (Bayesian inference Using Gibbs Sampling) suite of software packages. The emphasis is placed on actuarial loss models, but other applications are referenced, and directions are given for obtaining documentation for additional worked examples on the World Wide Web.

1. INTRODUCTION

An important part of actuarial practice involves the construction of models and solutions for financial, business, and societal problems involving uncertain future events. This sentiment is expressed in the Mission and Vision Statement of the Society of Actuaries and is likewise echoed in the Syllabus of Examinations Goals and Objectives of the Casualty Actuarial Society. When modeling uncertain future events, it is often the case that uncertainty is modeled using statistical models, and that will be our *modus operandi* throughout this paper.

The Bayesian statistical method treats all unknown parameters appearing in a statistical model as random variables and derives their distribution conditional upon the known information. Makov, Smith, and Liu (1996, p. 503) noted that “statistical methods with a Bayesian flavour, in particular credibility theory, have long been used in the insurance industry as part of the process of estimating risks and setting premiums.” Klugman (1992, p. 1) made the case that many general actuarial problems involving model-based prediction are well suited for the Bayesian method. It may be argued that the Bayesian paradigm is the most natural and convenient one to adopt for the implementation and analysis of many models arising in actuarial science, insurance, and risk management.

Until recently, though, fully Bayesian analyses of these statistical models had been computationally infeasible and approximation methods were often utilized instead. This changed in the early 1990s following the rediscovery in the statistical literature of computer-intensive Markov chain Monte Carlo (MCMC) simulation methods like the Metropolis-Hastings algorithm and the Gibbs sampler (Gelfand and Smith 1990; Gelfand et al. 1990).

Soon after, a number of researchers began to utilize these MCMC methods in actuarial contexts. For example, they were utilized by Carlin (1992a, b) to conduct the Bayesian state space modeling of nonstandard actuarial time series; by Carlin (1992c) and Klugman and Carlin (1993) in the context of Bayesian graduation; by Scollnik (1993) to implement a Bayesian analysis of a simultaneous equations model for insurance rate making; by Rosenberg (1994) to implement the Bayesian analysis of a hierarchical model for the rate of nonacceptable in-patient hospital utilization; by Shephard and Pitt (1995) to estimate mortality rates and to estimate the parameters appearing in several stochastic volatility models for financial returns; by Scollnik (1995b) in order to conduct the Bayesian analysis of some generalized Poisson models for claim frequency data; by Haastrup and Arjas (1996) in the context of claims reserving; by Scollnik (1996) to implement the analysis of three hierarchical credibility models for classification rate making and the prediction of frequency counts in workers compensation insurance; by Makov, Smith, and Liu (1996) in the context of credibility models, loss

* David P. M. Scollnik, A.S.A., Ph.D., is Associate Professor, Department of Mathematics and Statistics, University of Calgary, Calgary, Alberta, Canada, T2N 1N4, e-mail, scollnik@ucalgary.ca.

reserving, and for actuarial graduation; by Pai (1997) to analyze a compound model for insurance claims; and by Rosenberg and Young (1999) to analyze time-dependent data with possible level and/or variance shifts in the process at all points in time. This list is not exhaustive, but it is sufficient to indicate that MCMC methods have been successfully employed in a variety of actuarial contexts.

Although many of the actuarial models appearing in these papers are easy enough to explain to other actuarial practitioners and researchers, a majority of these same individuals would be hard-pressed to reproduce quickly any of the corresponding MCMC-based Bayesian analyses, as this task would typically require significant scientific programming ability and facility with random number generators. Fortunately, specialized software for implementing MCMC analyses is now available. The foremost are the BUGS (Spiegelhalter et al. 1996) and WinBUGS software packages. BUGS is an acronym for Bayesian inference Using Gibbs Sampling. This software was originally developed by the Biostatistics Unit of the Medical Research Council, Cambridge, United Kingdom, and is now jointly developed with the Imperial College School of Medicine at St. Mary's, London.

BUGS is a specialized software package for implementing MCMC-based analyses of full probability models in which all unknowns are treated as random variables. Its programming language is easily understood and allows the user to make a straightforward specification of the full probability model under consideration. Because the sole purpose of the BUGS package is to implement MCMC-based analyses of full probability models, its associated programming language was very efficiently designed for this task. Consequently, a MCMC analysis written in Fortran and consisting of several hundreds of lines of code can often be reduced to literally one or two dozen lines of code in BUGS. Versions of BUGS exist for a variety of computer platforms. The WinBUGS package runs under Windows 95/98/NT and provides a graphical interface to the BUGS language that allows models to be constructed with the point and click of a mouse. Both packages are available on the World Wide Web. The Internet addresses for these and other related software packages are provided in Section 9.

From the perspective of an actuarial practitio-

ner, the BUGS/WinBUGS software is far from perfect: as yet, it does not explicitly support the full gamut of popular loss and frequency distributions found in Klugman, Panjer, and Willmot (1998), nor does it support arbitrarily truncated and/or grouped data to the extent one might wish. With luck, these deficiencies will be addressed in a future release of the software. Nevertheless, BUGS and WinBUGS are very convenient tools for implementing Bayesian analyses of a wide variety of actuarial models, thus making implementation of these analyses more routine and promising to make them significantly more accessible to the average actuarial practitioner.

This paper reviews the basics of MCMC, introduces Bayesian graphical models, and discusses and illustrates their implementation using the BUGS/WinBUGS programs. The implementation of several actuarial models will be illustrated, and directions given to access a number of additional examples of an actuarial nature worked by this author and available on the World Wide Web.

2. MARKOV CHAIN MONTE CARLO SIMULATIONS IN BRIEF

Consider a vector random variable $U = (U_1, \dots, U_k)$ with joint distribution $f(U_1, \dots, U_k)$. In a Bayesian context, some of these variables are model parameters while others may represent unobserved past or future data. Suppose $f(U)$ has a complicated and analytically intractable form, and the expected value of some integrable function $h(U)$ is sought. Even if this calculation cannot be performed analytically, it is still possible that the probabilistic model associated with $f(U)$ may be simple enough to permit independent random draws $U^{(t)}$, $t = 1, \dots, n$, from it. If this is the case, then the desired expectation can be approximated using

$$E[h(U)] \approx \frac{1}{n} \sum_{t=1}^n h(U^{(t)}).$$

For instance, the population mean can be estimated using the sample mean. This procedure is called Monte Carlo integration. Unfortunately, many complicated models will not readily permit independent random draws. In this case a MCMC simulation method can be used instead.

The main idea behind a MCMC method is to

simulate realizations from a Markov chain that has $f(U)$ as its stationary distribution. The random draws $U^{(1)}, U^{(2)}, \dots$, are no longer independent, but under mild regularity conditions (as described in the appendix of Smith and Roberts 1993, for example) they are such that

$$U^{(t)} \xrightarrow{d} U \sim f(U), \quad \text{as } t \rightarrow \infty, \quad (2.1)$$

and

$$\frac{1}{t} \sum_{i=1}^t h(U^{(i)}) \rightarrow E[h(U)],$$

as $t \rightarrow \infty$, almost surely. (2.2)

Equation (2.1) tells us that as t becomes moderately large, the value of $U^{(t)}$ tends in distribution to that of a random draw from $f(U)$. Equation (2.2) tells us that the ergodic average appearing on its left-hand side is a consistent estimator of the expected value of the integrable function $h(U)$ on its right-hand side.

How does one simulate realizations from a Markov chain with $f(U)$ as its stationary distribution? It turns out that a number of competing methods exist, dating back to Metropolis et al. (1953) and Hastings (1970). Recent reviews and applications appear in a number of papers and texts, including Gelman et al. (1995), Gilks, Richardson, and Spiegelhalter (1996), Smith and Roberts (1993), and Tierney (1994). We will restrict our attention to one of the simplest of MCMC methods, the Gibbs sampler, which was introduced by Geman and Geman (1984) in the context of image restoration and subsequently to the statistics literature by Gelfand and Smith (1990). Soon after, Carlin (1992a, b, c) utilized it in several actuarial contexts. As noted previously, the Gibbs sampler is also the basis of BUGS/WinBUGS; the algorithm defining it is described below. Note that Gibbs sampling is actually a special case of the Metropolis-Hastings algorithm (see Gelman et al. 1995, p. 328, for further details).

As before, consider a vector random variable $U = (U_1, \dots, U_k)$ with joint distribution $f(U_1, \dots, U_k)$. We assume that this distribution exists and is proper. Incidentally, we will use the words “distribution” and “density” interchange-

ably, trusting the context to make clear exactly which is meant. Each of the U_i terms might denote either a single random variable or, more generally, a block of several random variables grouped together. The j -th block of variables will have its marginal distribution identified as $f(U_j)$ and $f(U_j|U_1, \dots, U_{j-1}, U_{j+1}, \dots, U_k)$ will be used to identify the full conditional distribution of the j -th block given the remaining variables. Given an arbitrary vector of starting values $U^{(0)} = (U_1^{(0)}, \dots, U_k^{(0)})$, the first iteration of the Gibbs sampler proceeds by making random draws from the full conditional distributions as follows:

$$\begin{aligned} U_1^{(1)} &\sim f(U_1|U_2^{(0)}, \dots, U_k^{(0)}) \\ U_2^{(1)} &\sim f(U_2|U_1^{(1)}, U_3^{(0)}, \dots, U_k^{(0)}) \\ &\vdots \\ U_j^{(1)} &\sim f(U_j|U_1^{(1)}, \dots, U_{j-1}^{(1)}, U_{j+1}^{(0)}, \dots, U_k^{(0)}) \\ &\vdots \\ U_k^{(1)} &\sim f(U_k|U_1^{(1)}, U_2^{(1)}, \dots, U_{k-1}^{(1)}). \end{aligned}$$

This completes a single iteration of the algorithm and defines a transition from $U^{(0)}$ to $U^{(1)} = (U_1^{(1)}, \dots, U_k^{(1)})$. After t such iterations, we have $U^{(t)} = (U_1^{(t)}, \dots, U_k^{(t)})$. The resulting sequence of dependent draws $U^{(1)}, U^{(2)}, U^{(3)}, \dots$, will satisfy Equations (2.1) and (2.2), subject to the mild regularity conditions mentioned previously.

Exactly how long a MCMC simulation should be run is a function of the particular application. Usually several tens of thousands of iterations are more than sufficient for our purposes; often a few thousand will do. Several methods are available with which to monitor the convergence of a MCMC simulation, and a couple of these are described later (see Cowles and Carlin 1996 for a fairly recent review). In any case, the first portion of the simulated Markov chain is normally discarded in order to reduce the effect of the starting values. If we let m denote the number of discarded or “burn-in” iterations and let n denote the number of iterations in total, then $E[h(U)]$ can be approximated using the ergodic average

$$\frac{1}{n-m} \sum_{t=m+1}^n h(U^{(t)}). \quad (2.3)$$

An ad hoc but useful test of convergence is obtained by running several simulations in parallel, with different starting values, and then comparing the resulting estimates given by expression (2.3).

If these estimates are not in the same ballpark, the value of n must be increased.

An application of the Gibbs sampler is described in the next section. However, its implementation will be delayed until we have introduced the basics of graphical modeling and the BUGS/WinBUGS software.

3. A HIERARCHICAL POISSON MODEL FOR CLAIM FREQUENCY DATA

To provide a context for an example, suppose that an insurance company has three group workers compensation policies with associated payrolls and loss frequency counts for four policy years as shown in Table 1. The problem is to describe the loss frequencies anticipated in the fifth year.

If we accept the payroll figures as a proxy for the exposures, one approach is to make a straightforward application of the Bühlmann-Straub credibility model as described in Herzog (1996, chap. 7) and Klugman, Panjer, and Willmot (1998, sections 5.4.4 and 5.5.1). The non-parametric method described in these texts can be used to estimate the necessary population parameters and credibility factors; we omit the detailed calculations. In terms of the notation found in Herzog (1996) these calculations yield values $m = (1205, 1040, 370)$, $\hat{\alpha} = 0.00005529$, $\hat{\nu} = 0.023697$, $\hat{k} = 428.59$, $\hat{Z} = (0.7376, 0.7082, 0.4633)$, $\hat{\mu} = 0.0268$, and $\bar{X} = (0.0290, 0.0192, 0.0405)$. It follows that the Bühlmann-Straub estimates of the average loss frequency per unit of exposure for the three group policyholders in the fifth year are

$$\hat{Z}_1 \bar{X}_1 + (1 - \hat{Z}_1) \hat{\mu} = 0.0284,$$

$$\hat{Z}_2 \bar{X}_2 + (1 - \hat{Z}_2) \hat{\mu} = 0.0214,$$

$$\hat{Z}_3 \bar{X}_3 + (1 - \hat{Z}_3) \hat{\mu} = 0.0331.$$

The aggregate loss frequencies for the second and third group policyholders are simply $6.10 = 285(0.0214)$ and $3.81 = 115(0.0331)$. The aggregate loss frequency for the first group policyholder is a little more problematical to obtain, as the exposure for the fifth year is either unknown or at least unspecified in Table 1.

A number of authors, including Schnieper (1995), have noted that in many practical applications of credibility theory the structure parameters must be estimated from the data, as was done above. This leads to an estimator of the a posteriori mean that is often biased and where the credibility factor depends on the data. Another approach would be to treat the unknown parameters as random variables and then develop the posterior and predictive distributions of interest using the Bayesian method. In this light, we may be willing to adopt the following complete probability model for the data appearing in Table 1. Let X_{ij} and P_{ij} denote the number of claims and payroll, respectively, for the i -th group policyholder in the j -th policy year. We assume that all X_{ij} are conditionally independent and that $X_{ij} \sim \text{Poisson}(P_{ij}\theta_i)$, for all i and j . The parameters θ_i denote the expected number of losses per unit of exposure for group policyholder i . Given α and β , the θ_i are assumed to be conditionally independent with $\theta_i \sim \text{gamma}(\alpha, \beta)$, for all i . These gamma distributions are parameterized to have mean α/β . We complete this model by letting $\alpha \sim \text{gamma}(5, 5)$ and $\beta \sim \text{gamma}(25, 1)$. These last two distributions were selected rather arbitrarily for the sake of illustration. However, they imply that each θ_i has a prior mean and standard deviation approximately equal to 0.041 and 0.048,

Table 1

Claim Frequency Data for Three Group Policyholders

Year	Group 1		Group 2		Group 3	
	Payroll	Claims	Payroll	Claims	Payroll	Claims
1	280	9	260	6		
2	320	7	275	4	145	8
3	265	6	240	2	120	3
4	340	13	265	8	105	4
5	?	?	285	?	115	?

respectively, which is not unreasonable for the present context.

Let $X_1 = (X_{11}, X_{12}, X_{13}, X_{14})$, $X_2 = (X_{21}, X_{22}, X_{23}, X_{24})$, $X_3 = (X_{32}, X_{33}, X_{34})$, $X = (X_1, X_2, X_3)$, and $\theta = (\theta_1, \theta_2, \theta_3)$. According to the probability model under consideration, the variables X , θ , α , and β have their joint distribution $f(X, \theta, \alpha, \beta)$ given by

$$\prod_{j=1}^4 f(X_{1j}|P_{1j}\theta_1) \prod_{j=1}^4 f(X_{2j}|P_{2j}\theta_2) \times \prod_{j=2}^4 f(X_{3j}|P_{3j}\theta_3) \prod_{j=1}^3 f(\theta_j|\alpha, \beta) f(\alpha) f(\beta). \quad (3.1)$$

After the data are observed, our interest turns to the posterior distribution $f(\theta, \alpha, \beta|X)$. Bayes' Theorem tells us that $f(\theta, \alpha, \beta|X) \propto f(X, \theta, \alpha, \beta)$. As this posterior distribution has a complicated form, it is most conveniently summarized using simulation. This might be accomplished using the Gibbs sampler of the previous section. Implementing a Gibbs sampler coded from scratch would require us to identify and then construct an effective simulation method for each of the five related full conditional posterior distributions, namely, $f(\theta_1|X, \theta_2, \theta_3, \alpha, \beta)$, $f(\theta_2|X, \theta_1, \theta_3, \alpha, \beta)$, $f(\theta_3|X, \theta_1, \theta_2, \alpha, \beta)$, $f(\alpha|X, \theta_1, \theta_2, \theta_3, \beta)$, and $f(\beta|X, \theta_1, \theta_2, \theta_3, \alpha)$. These are exactly the same tasks the BUGS/WinBUGS software obviates, as it performs these steps internally and automatically. However, we will derive the form of the full conditional distributions one time for illustration's sake.

Before we do this, however, note that we are operating within a full Bayesian setting in the paragraphs above, as we are interpreting the parameters α and β as random variables in their own right. This is in accordance with the hierarchical Bayes approach to modeling introduced by Lindley and Smith (1972). This approach is discussed and illustrated several times in Klugman (1992); in fact, Klugman considered a full Bayesian analysis of a simpler hierarchical Poisson model than our own with equal exposures and a known value of α (pp. 67–71). Klugman observed that incorporating a prior density for α into the model would considerably complicate the analysis. However, treating α as a random parameter makes almost no difference to an analysis pro-

ceeding on the basis of MCMC. In fact, Makov, Smith, and Liu (1996, p. 512) demonstrated that it is a trivial matter to incorporate random exposures into the analysis as well.

Returning to the derivation of the full conditional posterior distributions required to implement the Gibbs sampler, Bayes' Theorem says that every one of these is proportional to expression (3.1). Terms in expression (3.1) not involving the variable of interest are simply absorbed into the proportionality. Thus, we have

$$f(\theta_1|X, \theta_2, \theta_3, \alpha, \beta) \propto \prod_{j=1}^4 f(X_{1j}|P_{1j}\theta_1) f(\theta_1|\alpha, \beta) \propto \theta_1^{\alpha + \sum_{j=1}^4 X_{1j} - 1} \exp\left(-\left[\beta + \sum_{j=1}^4 P_{1j}\right]\theta_1\right) \sim \text{gamma}\left(\alpha + \sum_{j=1}^4 X_{1j}, \beta + \sum_{j=1}^4 P_{1j}\right). \quad (3.2)$$

As this distribution does not depend on θ_2 and θ_3 , we can denote it more simply and concisely as $f(\theta_1|X, \alpha, \beta)$. Similarly,

$$f(\theta_2|X, \alpha, \beta) \sim \text{gamma}\left(\alpha + \sum_{j=1}^4 X_{2j}, \beta + \sum_{j=1}^4 P_{2j}\right), \quad (3.3)$$

and

$$f(\theta_3|X, \alpha, \beta) \sim \text{gamma}\left(\alpha + \sum_{j=2}^4 X_{3j}, \beta + \sum_{j=2}^4 P_{3j}\right). \quad (3.4)$$

In a like manner,

$$f(\alpha|X, \theta, \beta) \propto \prod_{i=1}^3 f(\theta_i|\alpha, \beta) f(\alpha) \propto \left\{\frac{\beta^\alpha}{\Gamma(\alpha)}\right\}^3 \left\{\prod_{i=1}^3 \theta_i\right\}^\alpha \alpha^4 \exp(-5\alpha). \quad (3.5)$$

This distribution is of a nonstandard form. However, the logarithm of its density function happens to be concave with respect to α , meaning

that the method of adaptive rejection sampling described in Gilks and Wild (1992) and Wild and Gilks (1993) could be used to generate random draws from it. The final full conditional posterior distribution required to implement the Gibbs sampler in this example is of the form

$$\begin{aligned}
 f(\beta|X, \theta, \alpha) &\propto \prod_{i=1}^3 f(\theta_i|\alpha, \beta)f(\beta) \\
 &\propto \beta^{3\alpha+24} \exp\left(-\left[\sum_{i=1}^3 \theta_i + 1\right]\beta\right) \\
 &\sim \text{gamma}\left(3\alpha + 25, \sum_{i=1}^3 \theta_i + 1\right).
 \end{aligned}
 \tag{3.6}$$

With the full conditional distributions given by Equations (3.2)–(3.6) now identified, we can proceed to iteratively sample from them in accordance with the algorithm described in the previous section with $f(\theta, \alpha, \beta|X)$ taking the place of $f(U)$, and beginning with an arbitrary vector of starting values $(\theta^{(0)}, \alpha^{(0)}, \beta^{(0)})$. The resulting sequence of draws $(\theta^{(1)}, \alpha^{(1)}, \beta^{(1)})$, $(\theta^{(2)}, \alpha^{(2)}, \beta^{(2)})$, \dots , can be used to implement posterior inference either by simply examining their empirical distribution or else in the manner of Equation (2.2) and Equation (2.3). To illustrate the latter, note that the posterior distribution $f(\theta_i|X)$ can be estimated using the observation that

$$\begin{aligned}
 f(\theta_i|X) &= \int_{\alpha} \int_{\beta} f(\theta_i|X, \alpha, \beta)f(\alpha, \beta|X) d\beta d\alpha \\
 &\approx \frac{1}{n - m} \sum_{t=m+1}^n f(\theta_i|X, \alpha^{(t)}, \beta^{(t)}),
 \end{aligned}
 \tag{3.7}$$

for $i = 1, 2, 3$. Similarly, the predictive distribution of X_{i5} for a fixed level of exposure P_{i5} can be estimated using the observation that

$$\begin{aligned}
 f(X_{i5}|X) &= \int_{\theta_i} f(X_{i5}|P_{i5}\theta_i)f(\theta_i|X) d\theta_i \\
 &\approx \frac{1}{n - m} \sum_{t=m+1}^n f(X_{i5}|P_{i5}\theta_i^{(t)}),
 \end{aligned}
 \tag{3.8}$$

for $i = 1, 2, 3$. In this expression we understand that X_{i5} is conditionally independent of X given θ_i , and that $f(X_{i5}|P_{i5}\theta_i)$ denotes a Poisson distribution with mean $P_{i5}\theta_i$.

4. BAYESIAN GRAPHICAL MODELING

In this section we give a brief introduction to directed graphical models and identify their relevance to complex Bayesian modeling. This discussion draws substantially from the papers by Best et al. (1996), Smith, Spiegelhalter, and Parmar (1996), and Spiegelhalter, Thomas, and Best (1996). Further details and additional references are available in these papers.

We start with a consideration of the following slight extension of the full probability model described in the previous section:

$$X_{ij} \sim \text{Poisson}(\lambda_{ij}), \tag{4.1}$$

$$\lambda_{ij} = P_{ij}\theta_i, \tag{4.2}$$

$$\theta_i \sim \text{gamma}(\alpha, \beta), \tag{4.3}$$

$$\alpha \sim \text{gamma}(5, 5), \tag{4.4}$$

$$\beta \sim \text{gamma}(25, 1), \tag{4.5}$$

$$P_{ij} \sim \text{gamma}(\alpha_i, b_i), \tag{4.6}$$

$$\alpha_i \sim \text{uniform}(0, 100), \tag{4.7}$$

$$b_i \sim \text{uniform}(0, 100). \tag{4.8}$$

For the data in Table 1, the values taken on by the indices are $i = 1, 2, 3$, and $j = 1, \dots, 5$. This is the form of the model we will implement using BUGS/WinBUGS. As before, we assume conditional independence for the variables appearing on the left-hand side of each line given the model parameters appearing on the right. So the variables X_{ij} are conditionally independent given the values of λ_{ij} , for example. The first five lines describe the original hierarchical Poisson model. The other three lines describe a simple probability model for the payrolls, thus permitting representative values of P_{15} and P_{31} to be simulated in the course of running the Gibbs sampler; recall that the values of these two payroll figures were not provided in Table 1. Observe that each group has its own payroll parameters α_i and b_i . Of course, a different model for the payrolls might be adopted, or the value of the missing payroll P_{15}

can simply be estimated and set equal to a fixed number. Notice that the unobserved variables P_{31} and X_{31} are also included in the model, but only to simplify the indexing in BUGS/WinBUGS. Otherwise, these variables play no role in the analysis, and their simulated values can be safely ignored.

The model has the particularly simple graphical representation given in Figure 1. This graph emphasizes the main qualitative features of the model without the use of cumbersome algebraic formulas. Nodes in the graph denote the data and parameters appearing in the model. Figure 1 is said to be a *directed* graph as each link between nodes is an arrow. The directed links are of two types, stochastic and logical. Stochastic links or dependencies, like Equation (4.1), are indicated with solid arrows. Logical links or dependencies, like Equation (4.2), are indicated with hollow ones. The arrows indicate the conditional independence assumptions of the model. They also suggest a parent-child analogy for the linked nodes. For example, we can say that α is a parent of θ_1 , and P_{15} is a child of α_1 . Given its parent nodes, denoted as $\text{parents}[\nu]$, each node ν is independent of all other nodes except for the descendants of ν . The graph in Figure 1 is also *acyclic*. This means that it is impossible to return

to a node after leaving it if one moves in the direction of the arrows. This also means we can speak unambiguously of a particular node's parents, ancestors, children, or descendants. A directed acyclic graph completely specifies the joint distribution of all the variables V it contains in terms of the parent-child distributions, that is,

$$f(V) = \prod_{\nu \in V} f(\nu | \text{parents}[\nu]). \quad (4.9)$$

Recall that expression (3.2) was of precisely this form.

The connection with Bayesian computation is that Equation (4.9) also defines the form of the full conditional distributions required for Gibbs sampling. Specifically, if ν is a node in the graph and $V \setminus \nu$ denotes all of the others, then its full conditional posterior distribution is given by

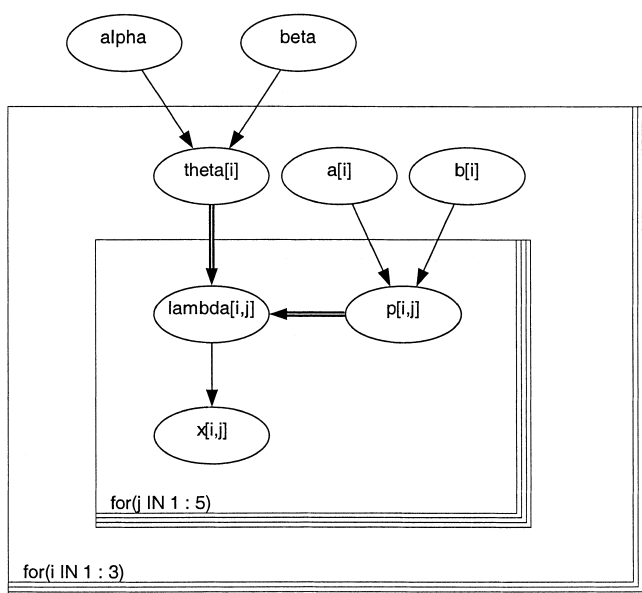
$$\begin{aligned} f(\nu | V \setminus \nu) &\propto f(\nu, V \setminus \nu) \\ &\propto \text{product of terms in } f(V) \text{ containing } \nu \\ &\propto f(\nu | \text{parents}[\nu]) \prod_{\tau \in \text{children}[\nu]} f(\tau | \text{parents}[\tau]). \end{aligned} \quad (4.10)$$

The same strategy was used in the derivation of expressions (3.2)–(3.6). This is a part of the “value added” to a Bayesian analysis by a graphical model with extensive conditional independence assumptions; that is, the form of the full conditional posterior distributions required by the Gibbs sampler can be derived automatically using Equation (4.10) to exploit the local conditional independence properties featured in the graphical model. The BUGS/WinBUGS packages essentially proceed on just this basis.

5. IMPLEMENTING THE POISSON MODEL USING BUGS/WINBUGS

We now discuss how the continuing Poisson model example, described by Equations (4.1)–(4.8), can be implemented using BUGS/WinBUGS. Recall that BUGS is a specialized software package for implementing the Bayesian analysis of complex statistical models using MCMC. Associated with the BUGS package is a specialized BUGS *language* in which a statistical model is expressed, a BUGS *compiler* to process the model and data, and a BUGS *sampler* to

Figure 1
Graphical Model for the Poisson Example



actually perform the MCMC simulation. The original BUGS package is run at the level of the command line prompt and is available for a variety of computer platforms. Sometimes it is referred to as “classic” BUGS in order to differentiate it from its interactive Windows counterpart, WinBUGS. This latter will process most models written in the classic BUGS language, but it also enables graphical models to be constructed and analyzed with the point and click of a mouse and the use of a menu-driven interface.

In summary, both packages provide a convenient means for the specification of graphical models and facilitate the Bayesian analysis of a graphical model’s unobserved nodes using MCMC by automatically deriving the form of their full conditional distributions and then selecting and implementing appropriate random number generation routines. Directions for obtaining the BUGS/WinBUGS software packages on the World Wide Web are provided in Section 9. Henceforth, our discussion will mostly concentrate on the WinBUGS package, as this appears to be the one now receiving the most attention from the BUGS development teams, contains some more sophisticated random number generation routines, and has the aforementioned graphical modeling capability and menu-driven interface. Further, many of the models constructed in WinBUGS can be ported over to classic BUGS with relatively little effort. The reader may wish to download and install the WinBUGS software in order to follow along the steps set forth below. At the time this paper was being prepared, the latest release of WinBUGS was (beta) version 1.2 (May 1999). The steps described below correspond to this version of the software.

Implementing a graphical model with WinBUGS normally involves the following steps. The graphical model has to be constructed, and a distribution must be assigned to each stochastic node. The model is next parsed by WinBUGS and checked to be syntactically correct. The observed data are read in, after which the model is compiled. Initial values for the unobserved nodes must now be specified or randomly generated, after which the MCMC simulation can be run. The values simulated for the unobserved nodes of interest can be monitored and saved, summarized, and/or checked for convergence as required. Most of these steps are illustrated below, but for complete details the reader should

refer to the WinBUGS user manual referenced in the next paragraph.

5.1. Getting Started with WinBUGS

When WinBUGS is started up, a window should appear with the following menu headings: *File, Tools, Edit, Attributes, Info, Model, Inference, Doodle, Text, Window, Help*. Contained within the main window may be a smaller one titled *Log* into which WinBUGS will send its responses to various session commands. If a *Log* window is not available, create one by selecting *Open Log* from the *Info* menu. At this time, the reader should review the first two help files (*User Manual* and *Doodle Help*) available under the last of the menu headings before continuing. The *User Manual* file is just that. The *Doodle Help* file is a quick help file listing the commands associated with DoodleBUGS. In WinBUGS parlance, a *doodle* is a pictorial representation of a graphical model with certain attributes (e.g., name, type, distribution, or logical function definition) assigned to each node. Doodles are created using DoodleBUGS, a doodle editor.

5.2. Constructing a Doodle with DoodleBUGS

Selecting the *New* command under the *Doodle* menu will open a dialog box titled *New Doodle*. Enter 200 and 150 in the *view width* and *view height* slots, respectively. These two options control the size of the editor window, and the values just input should be adequate for our purpose. Click OK. An untitled DoodleBUGS editor window should now appear. Our goal is to re-create the graphical model depicted in Figure 1 as a doodle in this editor window.

Clicking the mouse on a blank space in the editor window creates an ellipse-shaped node. Do this now, somewhere in the upper left-hand portion of the editor window. If you are unhappy with the placement of this node, it can be repositioned by clicking and dragging it with the mouse. When a node is created, blue headings will simultaneously appear at the top of the window, and a flashing cursor will be located beside the *name* heading. Enter *alpha* as the name of this node. To the right of the name is the *type* heading. Clicking on this heading will produce a list of three menu selections: *stochastic, logical, and constant*. The

first is the default and is the correct selection for this particular node as it represents a random parameter. To the right of the type is the *density* heading. Clicking on this heading will produce a list of 16 distribution types (see table I in the user manual for their definitions and parametrizations). Select the *dgamma* density for this node in accordance with expression (4.4). Next, click on the *shape* and *scale* headings and enter the numerical value 5 in each slot—again, in accordance with expression (4.4). We leave empty slots next to the *lower bound* and *upper bound* headings as we have no desire to restrict the range of values taken on by this particular node. This concludes the definition of the node corresponding to the model parameter α .

Click the mouse once again in the main part of the editor window to create a new node to be labeled *theta*[*i*]. The new node will now be highlighted instead of the node labeled *alpha*. The highlighted node is always the one being edited. When multiple nodes appear in the doodle editor, a node is highlighted when it is clicked upon with the mouse. Once highlighted, a node can be repositioned in the window by simply clicking and dragging it with the mouse. Position the nodes so that their placement roughly corresponds to that shown in Figure 2. Click on the *theta*[*i*] node to highlight it in order to continue editing its attributes. Select its density type as *dgamma* in accordance with expression (4.3). After first making sure that the *theta*[*i*] node is still the one highlighted, push and hold the control (Ctrl) key and then click on the *alpha* node. A solid arrow

(called an edge) will appear joining the nonhighlighted node labeled *alpha* to the highlighted node labeled *theta*[*i*]—that is, linking the parent node to its child—and the name of the former appears in the slot beside the *shape* heading associated with the latter. Figure 2 depicts our progress thus far. For future reference, note that the command process used to remove an edge linking two nodes is identical to the process used to create one (i.e., first highlight the child, then click the mouse on the parent node while pushing the Control key). Also, an errantly placed node is removed by first selecting it and then simultaneously pushing the Control and Delete or Control and Backspace keys. Hence, both nodes and edges are easily created and deleted.

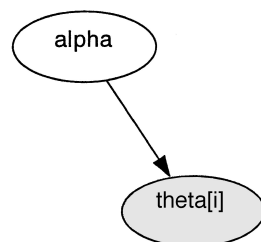
Continuing in this manner, create and define a new node labeled *beta* in accordance with expression (4.5). Select *dgamma* as its density and enter the values 25 and 1 in the *shape* and *scale* slots, respectively. Now highlight the node labeled *theta*[*i*] and create an edge from the node labeled *beta* to it. The name of the *beta* node will automatically be placed in the slot next to the *scale* heading associated with the node labeled *theta*[*i*] at the same time the edge is created. The density associated with the node labeled *theta*[*i*] should now be *dgamma* with *shape* parameter *alpha* and *scale* parameter *beta* in accordance with expression (4.3). If the need arose, the values in the *shape* and *scale* slots could be edited by clicking on the appropriate heading with the mouse and selecting the desired menu entry.

The next step will be to create and place the

Figure 2

Doodle Under Construction

name:	theta[i]	type:	stochastic	density:	dgamma	
shape	alpha	scale	1.0E-3	lowerbound		upperbound



two rectangular “plates” appearing in Figure 1, which represent the indexing over group ($i = 1, 2, 3$) and year ($j = 1, \dots, 5$). A plate is created in the doodle editor window by pushing the Control key as one clicks the mouse. A plate is selected and highlighted by clicking the mouse on one of its thick edges. A highlighted plate can be moved by clicking and holding the mouse on one of the thick edges and then moving the mouse. It is resized by dragging the mouse while clicking on the plate’s extreme lower right-hand corner, where the two thick edges meet, and deleted by simultaneously pushing the Control and Delete or Control and Backspace keys. When a plate is selected, the three blue headings *index*, *from*, and *up to* will appear at the top of the window. The index name is entered in the first of these slots. The index is understood to take on sequential integer values running from the integer value entered in the *from* slot to the integer value entered in the *up to* slot. If the reader has not already done so, two plates should be created at this time and then positioned, resized, and labeled as in Figure 3.

The reader should now create and place nodes

labeled $a[i]$, $b[i]$, and $p[i, j]$ as in Figure 4. The first two should have *dunif* selected for their density, with *lower bound* and *upper bound* set equal to 0 and 100 in each case in accordance with expressions (4.7) and (4.8). The node labeled $p[i, j]$ should be defined in accordance with expression (4.6) (i.e., using density *dgamma* with *shape* and *scale* parameters $a[i]$ and $b[i]$, respectively). Next, create and place the $\lambda[i, j]$ node. For this node, select its type to be *logical*. Click the *value* heading and enter the logical expression $p[i, j] * \theta[i]$ in the blank slot in accordance with expression (4.2). Logical expressions of this sort can be formed using parent nodes, the operators $+$, $-$, $*$, $/$, and a number of standard mathematical functions (e.g., *exp*, *log*, *abs*, and *sqrt*) listed in table II of the user manual. Next, create edges leading from the parent nodes labeled $\theta[i]$ and $p[i, j]$ to their child node $\lambda[i, j]$ using the same procedure as before. This time the edges will appear as hollow arrows to clearly differentiate these logical links from the stochastic ones we have created up until now.

Finally, create and place a node labeled $x[i, j]$ as depicted in Figure 5. For this node, select the

Figure 3
Doodle Under Construction

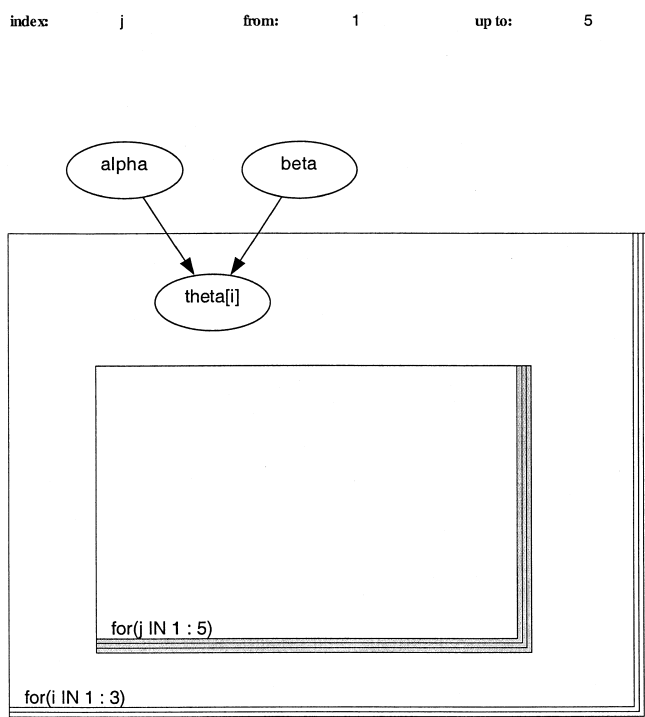


Figure 4
Doodle Under Construction

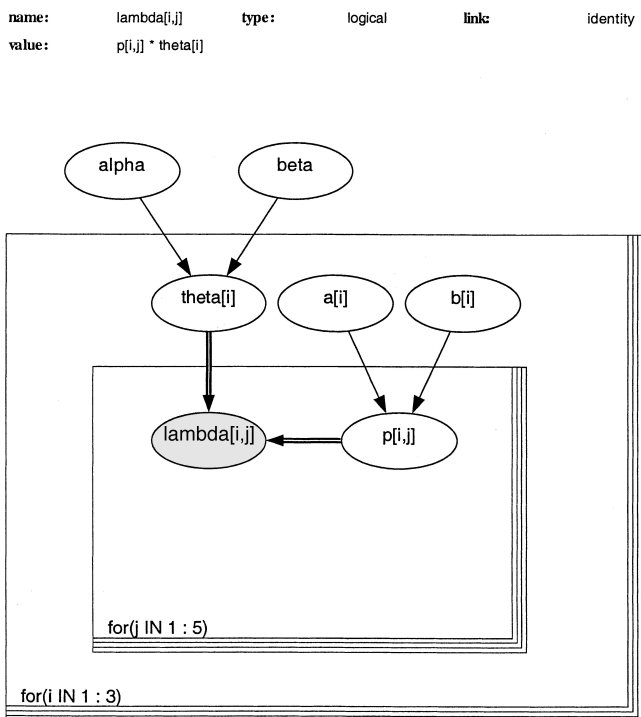


Table 2
Estimated Posterior Summary Statistics

Parameter	Mean	SD	2.5%	Median	97.5%
x[1, 5]	8.918	3.654	3.0	9.0	17.0
x[2, 5]	5.611	2.677	1.0	5.0	11.0
x[3, 1]	4.889	2.766	1.0	5.0	11.0
x[3, 5]	4.638	2.436	1.0	4.0	10.0
theta[1]	0.02935	0.004946	0.02039	0.0291	0.03956
theta[2]	0.01975	0.004352	0.01218	0.01938	0.02905
theta[3]	0.04047	0.01019	0.02269	0.0397	0.06247
alpha	1.067	0.346	0.4836	1.033	1.82
beta	25.98	5.014	17.27	25.63	36.68
ρ [1, 5]	303.3	47.16	216.8	300.5	403.3
ρ [3, 1]	122.8	23.82	80.44	121.1	175.0
a[1]	63.61	22.72	18.37	65.37	98.18
a[2]	71.4	19.55	28.4	74.01	98.87
a[3]	49.06	24.73	12.06	45.84	96.11
b[1]	0.2112	0.07659	0.06042	0.2166	0.3321
b[2]	0.2693	0.07486	0.1059	0.2777	0.3825
b[3]	0.4039	0.2057	0.0968	0.3771	0.7988
lambda[1, 1]	8.217	1.385	5.708	8.149	11.08
lambda[1, 2]	9.391	1.583	6.524	9.313	12.66
lambda[1, 3]	7.777	1.311	5.402	7.712	10.48
lambda[1, 4]	9.978	1.682	6.931	9.895	13.45
lambda[1, 5]	8.904	2.073	5.446	8.709	13.62
lambda[2, 1]	5.135	1.132	3.167	5.039	7.552
lambda[2, 2]	5.431	1.197	3.35	5.33	7.988
lambda[2, 3]	4.74	1.044	2.923	4.652	6.971
lambda[2, 4]	5.234	1.153	3.228	5.136	7.697
lambda[2, 5]	5.629	1.24	3.471	5.524	8.278
lambda[3, 1]	4.964	1.581	2.474	4.742	8.677
lambda[3, 2]	5.868	1.477	3.289	5.756	9.059
lambda[3, 3]	4.856	1.222	2.722	4.764	7.497
lambda[3, 4]	4.249	1.07	2.382	4.168	6.56
lambda[3, 5]	4.654	1.171	2.609	4.565	7.185

Note: Estimates from WinBUGS.

density *dpois* in accordance with expression (4.1) and create an edge from the node labeled *lambda*_[i] to it. The name of the parent node will automatically appear in the slot next to the *mean* heading.

This concludes the construction of the doodle for the Poisson example. At this time it may be wise to save the doodle by selecting the *Save As* command in the *File* menu and responding to the dialog box in an appropriate fashion (e.g., select a suitable directory and enter *claims-gm* as the file name). Saved files can later be accessed through the *Open* command in the *File* menu.

5.3. Checking and Compiling the Model

After constructing a doodle, it is necessary to verify whether it is recognized by WinBUGS as

being syntactically correct. Should multiple files and windows be open in your WinBUGS session, bring the window containing the doodle of interest to the forefront. Then select the *Specification* command in the *Model* menu. A dialog box titled *Specification Tool* should appear. Click its *check model* button. If the doodle was properly constructed, the message “model is syntactically correct” will appear in the status line located in the bottom left-hand corner of the main WinBUGS window and also in the *Log* window. Otherwise, you will need to inspect the doodle node by node and plate by plate to verify that each is properly defined. Remember, highlight a node or plate by clicking on it to inspect its defined attributes. Make changes or corrections where necessary. When

satisfied, try clicking the *check model* button once again, and now WinBUGS will—we hope—declare the model as being syntactically correct. If that is still not the case, try downloading a correct version of the doodle from the website referenced in Section 8 before giving up in frustration. Comparing the doodle in this file to your own should reveal any remaining errors. Incidentally, a doodle can be selected at any time by pushing the Control and Spacebar and then copied and pasted or dragged with the mouse to other applications (e.g., MS Word).

With the model checked, it is now time to define the data and the starting values. Begin by selecting the *New* command in the *File* menu. The new window is an editor for a hyperdocument, into which regular text can be input and formatted using the commands in the *Attributes* menu. Doodles can also be copied and pasted or dragged into this hyperdocument. The data appearing in Table 1 can be defined with these lines:

```
list(x = structure( .Data = c( 9, 7, 6, 13, NA
                          6, 4, 2, 8, NA,
                          NA, 8, 3, 4, NA,
                          .Dim = c( 3, 5 ) ),
      p = structure( .Data = c( 280, 320, 265, 340, NA,
                          260, 275, 240, 265, 285,
                          NA, 145, 120, 105, 115 ),
                          .Dim = c( 3, 5 ) ) )
```

This form of data representation is said to be in the S-PLUS format. Observe that x and p are both defined as 3×5 dimensional arrays. Missing values are entered as NA. These arrays could also be coded in a rectangular format like this:

$x[,1]$	$x[,2]$	$x[,3]$	$x[,4]$	$x[,5]$	$p[,1]$	$p[,2]$	$p[,3]$	$p[,4]$	$p[,5]$
9	7	6	13	NA	280	320	265	340	NA
6	4	2	8	NA	260	275	240	265	285
NA	8	3	4	NA	NA	145	120	105	115

Both modes of data representation are fairly self-evident and are described in detail in sec-

tion 5.1 of the WinBUGS user manual. Once the data are typed into the file (using either representation), highlight them using the mouse and then click the *load data* button on the *Specification Tool* dialog box. The message “data loaded” will appear in the status line and in the *Log* window. Note that to highlight a text passage in a WinBUGS window, simply drag the mouse over the appropriate passage while simultaneously pushing the left mouse button. Section 5.1 of the WinBUGS user manual describes the process of data entry in more detail.

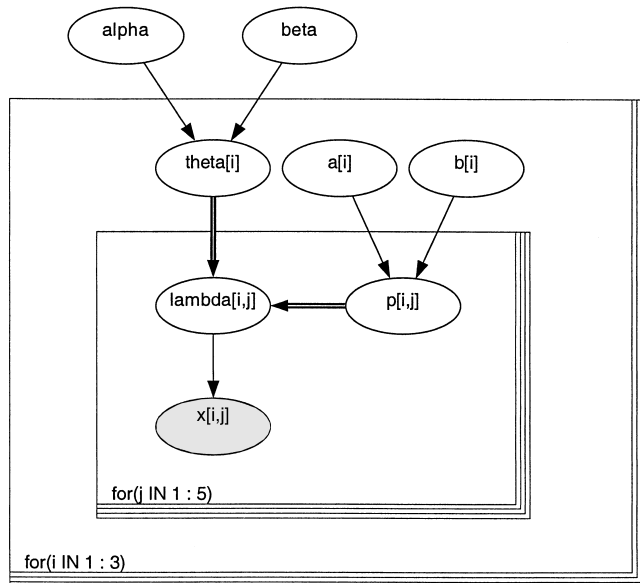
The next step is to compile the model, which is accomplished by clicking the *compile* button. Prior to so doing, note that a positive integer greater than the default value of 1 can be input into the text entry box next to the *compile* button if the user wishes to simulate a number of independent chains in parallel. For now, we will be satisfied with generating a single chain. As noted in the user manual, compiling the model builds the data structures required to carry out the

Gibbs sampling. Also, the model is checked for completeness and consistency with the data. When the model has been successfully compiled, the message “model compiled” should appear in the status line and in the *Log* window.

Before the compiled model can be run, it remains to initialize the values of any unobserved nodes.

Figure 5
Doodle Under Construction

name:	x[i,j]	type:	stochastic	density:	dpois
mean:	lambda[i,j]	lowerbound:		upperbound:	



The following line will make a start on this task:

```
list( alpha = 1, theta = c( 2, 2, 2 ) )
```

Remember, these are only starting values for a simulation and need not necessarily bear much resemblance to the values we actually expect these parameters to take on. Enter this line into the file, highlight it with the mouse, then click the *load inits* button. The message “initial values loaded; model contains uninitialized nodes (try running *gen inits*)” will appear in the usual places. (If you were compiling the model so as to generate a number of parallel chains, then initial values would need to be loaded as above once for each chain. In this case it usually makes sense to use different starting values for each chain.) The remaining uninitialized nodes (i.e., *beta*, *a*[1:3], *b*[1:3], and the missing counts and payrolls) can be taken care of by clicking the *gen inits* button as WinBUGS suggested above.

The message “initial values generated: model initialized” should now appear. At this point the model is ready to be run. First, though, it is probably wise to save the contents of the window you have just been editing to a file called, say, *claims*. This file will automatically be given the WinBUGS default file extension *.odc* as it is saved. A *claims*.

odc file is available from the website referenced in Section 8, along with files *claims.bug*, *claims.dat*, and *claims.in* for use with classic BUGS.

Incidentally, after a doodle is checked it is possible to select the *Write Code* command in the *Doodle* menu. This will open a hyperdocument window with the model expressed in the classic BUGS language. This facility allows models to be ported from WinBUGS to BUGS with relative ease. In fact, it is also the case that a model coded in this fashion can be read into WinBUGS by simply highlighting the word “model” at the start of the file and then clicking the *check model* button on the *Specification Tool* dialog box. Sometimes, especially after a little practice, it is easier to code a complicated model for WinBUGS directly in this manner than it is to create the corresponding doodle in DoodleBUGS. Illustrative BUGS code for the Poisson example is given below. Observe that each node appears once on the left-hand side of an expression and is defined as either stochastic (\sim) or logical ($<-$). The order in which the definitions appear is not particularly important, although the integrity of the “for” loops must be maintained. The code itself is fairly self-explanatory and is clearly describing the model defined by Equations (4.1)–(4.8).

CODE FOR THE POISSON MODEL

```
model;
{
  for( j in 1 : 5 ) {
    for( i in 1 : 3 ) {
      x[i,j] ~ dpois( lambda[i,j] )
      lambda[i,j] <- p[i,j] * theta[i]
      p[i,j] ~ dgamma( a[i], b[i] )
    }
  }
  for( i in 1 : 3 ) {
    theta[i] ~ dgamma( alpha, beta )
  }
  alpha ~ dgamma( 5, 5 )
  beta ~ dgamma( 25, 1 )
  for( i in 1 : 3 ) {
    a[i] ~ dunif( 0, 100 )
    b[i] ~ dunif( 0, 100 )
  }
}
```

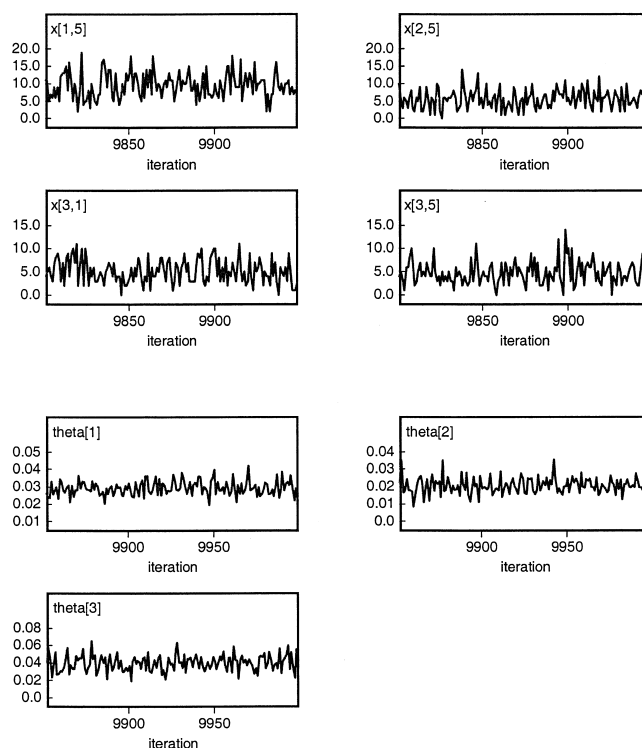
5.4. Running the Model and Monitoring the Output

Once the model is compiled and initialized, it is run by selecting the *Update* command in the *Model* menu. An *Update Tool* dialog box will appear. Enter the number of iterations, or updates, to be performed and click the *update* button. Try updating the model for 5,000 iterations. These updates took five seconds on a 150 MHz Pentium laptop PC, and two seconds on a dual 200 MHz Pentium Pro.

The output of the simulation is largely monitored using the commands under the *Inference* menu described in section 6 of the WinBUGS user manual. In particular, selecting the *Samples* command on the *Inference* menu opens a dialog box titled *Sample Monitor Tool*. The name of the variable to be monitored is entered into the open slot and either the *set* or *trace* button clicked. More than one variable can be monitored, but each must be entered and “set” individually. Now when the model is updated, the values taken on by the set variables will be stored by WinBUGS. If the *trace* button is clicked, the sample path of the selected variable will also be displayed in a dynamic trace plot. Set the nodes x and θ and open trace plots for each. Use the *Update Tool* to run another 5,000 updates. The trace plots generated should look similar to those depicted in Figure 6. The *stats* button produces summary statistics for a selected variable. Alternatively, we can enter the shortcut “*” to represent “all monitored nodes” and then click the *stats* button to produce summary statistics for the complete set of monitored variables all at once. Table 2 summarizes the results for the variables appearing in the Poisson example. The *density* button plots a smoothed kernel density estimate for the selected variable if it is continuous or a histogram if it is discrete. The density plots generated for x and θ are given in Figure 7. Estimated density plots could also be generated on the basis of Equations (3.7) and (3.8).

One other button available on the *Sample Monitor Tool* bears mention at this time (see section 6 of the user manual for an explanation of the remainder). Clicking the *coda* button opens two output windows. One, which we will refer to as the *ind* window, contains a list of names of the monitored variables and gives a

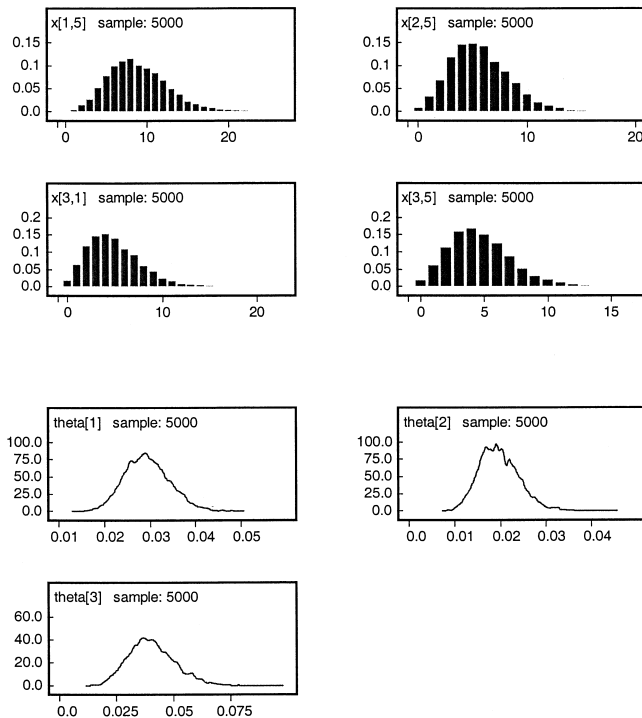
Figure 6
Trace Plots for Monitored Nodes



range of row numbers beside each. These row numbers correspond to lines of entries in the second window (the *out* window), which contains the set of simulated values taken on by the monitored variables. Hence, these two windows give an ASCII representation of the simulated data, which may be ported over (or clipped and pasted) to other applications, like Excel or S-PLUS. Two other application software packages worth mentioning are CODA and BOA, both of which are briefly described in Section 9. For use with either, the contents of the first and second windows should be saved to separate files as Plain Text documents with extensions *.ind* and *.out*, respectively (e.g., files *claims.ind* and *claims.out*). If your computer gives you trouble, try putting double quotation marks around the file name when saving these files, in order to preserve these non-standard file extensions.

In passing, we note that Makov, Smith, and Liu (1996) applied a Poisson model very similar to our own to a data set appearing in Klugman (1992, pp. 185–96). The data consist of payrolls and loss counts for 133 occupation classes over each of

Figure 7
Density Plots for Monitored Nodes



seven years (cf. Scollnik 1996). Our model is easily applied to the same data by simply modifying the sizes of the vectors and arrays so that $i = 1, \dots, 133$, and $j = 1, \dots, 7$. We must also replace zero payrolls in the data with the value NA in order to simplify the indexing in WinBUGS as before. It took WinBUGS 346 seconds to run the revised model for 5,000 updates on a 150 MHz Pentium laptop PC (169 seconds on a dual 200 MHz Pentium Pro), and our results are in substantial agreement with the predictions in Makov, Smith, and Liu (1996, p. 513) and Scollnik (1996, p. 155). The reader can verify this analysis using WinBUGS along with the file *klugdata.odc*, which is available from the website referenced in Section 8.

5.5. Monitoring and Assessing Convergence

Monitoring and assessing the convergence of the MCMC simulation is an important part of any MCMC-based analysis of a full probability model. A MCMC simulation is said to have converged when its output from that point on is

coming from the true stationary or target distribution of the Markov chain. On rare occasions it may be possible to prove convergence theoretically. More often than not, we are satisfied with a diagnosis of effective convergence. Brooks and Gelman (1998, p. 435) wrote that “Operationally, effective convergence of Markov chain simulation has been reached when inferences for quantities of interest do not depend on the starting point of the simulations.” Effective convergence is usually assessed on the basis of some statistical analysis, using one or more convergence diagnostics. Our discussion will focus on the ability of WinBUGS to simulate multiple chains in parallel and to calculate and plot modified Gelman-Rubin convergence diagnostic statistics on the basis of these parallel chains.

The Poisson model—or claims example—described in the *claims.odc* file will remain the context for our discussion. Recall that after checking this model and loading its data using the *Specification Tool* in WinBUGS, the next step was to compile the model. In WinBUGS a model can be compiled to simulate either a single chain of simulated values (the default) or a number of independent chains generated in parallel. For the latter, it is necessary to input a positive integer greater than 1 in the open slot labeled “num of chains” beside the *compile* button on the *Specification Tool* dialog box prior to compiling the model. In order to follow along below, the reader should now recheck the Poisson model in the *claims.odc* file and reload its data. Enter the integer 4 in the open slot beside the *compile* button, and then compile the model. Generally, we would recommend compiling the model to run between three and six multiple chains.

If all went well, the *load inits* button will now become active. Initial values are now loaded much as before, except that this time one set of values will need to be entered for each chain; that is, each chain needs to be initialized separately. Different initial values should be supplied for each chain. The different initial values assigned to a particular parameter should also be fairly widely dispersed compared to the values it is likely to take on in its stationary or target distribution. For our illustration, we used these four sets of values:

Enter these four lines in a WinBUGS editor window,

```
list( alpha = 1, theta = c( 2, 2, 2 ) )      # for chain 1
list( alpha = 0.2, theta = c( 0.2, 0.6, 1 ) )  # for chain 2
list( alpha = 2, theta = c( 0.2, 0.2, 0.2 ) )  # for chain 3
list( alpha = 4, theta = c( 6, 6, 6 ) )        # for chain 4
```

possibly at the end of the *claims.odc* file you created previously. With the mouse, highlight each line, one at a time, and then click the *load inits* button immediately after highlighting each line in turn. After loading the fourth and last of these lines, click the *gen inits* button to initialize the remaining model parameters. The model is now ready to be updated using the *Update Tool* as before.

Use the *Sample Monitor Tool* to set and also trace the nodes alpha, beta, and theta (remember, you can use the shortcut “*” to represent “all monitored nodes” after the desired nodes have been set). Note that the entries in the open slots to the right of the one in which the name of the node is entered indicate over which chains the values of the named node will be monitored (e.g., chains 1–4). These numerical entries can be edited if the practitioner wishes to monitor only a subset of the chains (e.g., chains 2–3). Keep the default setting for now in order to monitor all four chains and use the *Update Tool* to update the model for 200 iterations. The trace plots generated for the monitored nodes across the four chains should look similar to those in Figure 8.

From these simple plots, we observe that the chains do appear to mix rapidly: that is, there is no discernible behavior differentiating the path taken by a particular node in one chain from the path it takes in another, the different paths taken by a particular node frequently cross over one another, and the values taken on by a particular node in a given chain move quickly throughout that node’s parameter space. Update the model for another 4,800 iterations, watching the dynamic trace plots all the while in order to verify that this behavior continues. This visual inspection of multiple chain trace paths is a simple way to monitor convergence, but it is somewhat informal, qualitative, and hardly foolproof.

A number of more formal quantitative convergence diagnostics are available, many of which are discussed in the review article by Cowles and Carlin

(1996); see also section 5.4.5 of Carlin and Louis (1996). Many of these diagnostics are implemented by the CODA and BOA packages mentioned in Section 5.4 above. One diagnostic, based on a generalization due to Brooks and Gelman (1998) of a method proposed by Gelman and Rubin (1992), is available in WinBUGS. You can try implementing it for yourself by entering “*” to represent “all monitored nodes” in the open slot on the *Sample Monitor Tool* dialog box, and then clicking the *GR diag* button. Doing so should generate a sequence of plots, one for each monitored node, as in Figure 9. These plots along with the modified Gelman-Rubin convergence statistic itself as implemented in WinBUGS—at least, to the best of our understanding based on the discussion in the WinBUGS user manual—require some explanation.

Suppose we have run m chains in parallel and the parameter ψ was one of the monitored nodes. Any summary estimate (either point or interval) of the parameter ψ can be constructed using either the total pooled output across all m chains or on the

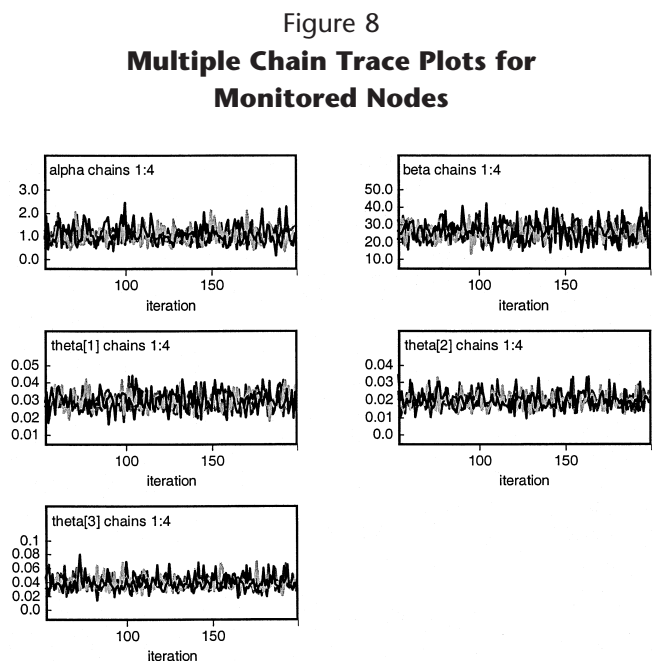
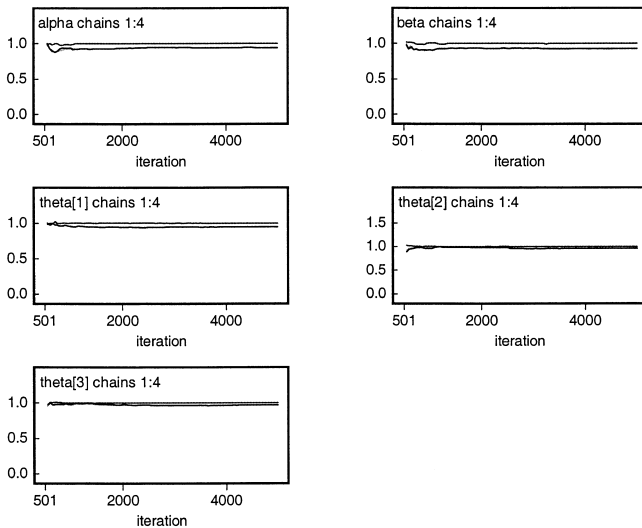


Figure 9
**Gelman-Rubin Convergence Statistic
 Diagnostic Plots**



basis of the output from each chain individually. The former would lead to a single pooled or total-sequence estimate, while the latter would lead to m within-sequence estimates. Consider the determination of an empirical central $100(1 - \alpha)\%$ (posterior) interval estimate for ψ using the pooled output. The endpoints of this interval are given by the empirical $100(\alpha/2)\%$ and $100(1 - \alpha/2)\%$ points for the complete set of values of ψ , that is, pooled across all m chains. Denote this pooled or total-sequence interval width estimate as R_p . Considering the output from each chain individually, we can also obtain m within-chain empirical central $100(1 - \alpha)\%$ interval estimates for ψ and then determine their mean width. Denote this mean within-sequence interval width estimate as R_w . Finally, calculate the ratio R of the total-sequence interval width estimate R_p to the mean within-sequence interval width estimate R_w , that is, $R = R_p/R_w$. All of these values can be calculated on the fly and recorded as the MCMC simulation progresses. As the MCMC simulation converges, the values of the total-sequence and mean within-sequence interval width estimates should each converge to a stable value, and their ratio R (the modified Gelman-Rubin convergence diagnostic) should converge to one. If this is not the case and R is still much greater than one, this suggests that the practitioner should either continue the simulation for a longer while or assume that there is some feature of the

model that is causing slow convergence. The process just described for ψ should generally be applied to each of the model parameters of main interest.

When the *GR diag* button is clicked, WinBUGS calculates these interval width quantities, after discarding a portion of the initial iterations as burn-in, for each selected node in bins of length 50 using $\alpha = 0.20$. WinBUGS also displays a plot of the three quantities R_p , R_w , and R for each selected node as in Figure 9. Note that the actual plots in WinBUGS are color-coded with the sequence of R_p , R_w , and R values in green, blue, and red, respectively. For plotting purposes, the total-sequence and within-sequence interval width estimates are also normalized to have an overall maximum of one. From Figure 9, we can observe that the convergence diagnostic R does appear to converge to one nearly immediately, and certainly by iteration 2,500 at the very latest, in the case of each monitored node, and that R_p and R_w have also stabilized in each plot by this time. Note that the actual values underlying any one of these diagnostic plots can be listed to a window by double-clicking on that plot followed by a Ctrl-left mouse click on the window. The table of values so reported by WinBUGS for the monitored node *beta* is presented as Figure 10.

Although we have not proved convergence of our MCMC simulation, we have gone some way toward diagnosing effective convergence by iteration 2,500. We could now discard the first 2,500 iterations as burn-in and base our inferences on the remaining 2,500. Conveniently, the *Sample Monitor Tool* has two open slots labeled “beg” and “end” that can be used to specify which iterations should be included in the calculation of any summary statistics and plots. Recall that the summary statistics in Table 2 were based on 5,000 iterations following a burn-in of 5,000. In light of the investigation above, this amount of burn-in was probably unnecessarily conservative.

Refer to the *bivreg* example in Scollnik (2000) for a much more interesting application of convergence diagnostics, in the context of a slowly converging regression model for bivariate claims data.

6. MODELING ARBITRARY DISCRETE DISTRIBUTIONS IN WINBUGS

The WinBUGS package explicitly supports very few discrete distributions, namely, the Bernoulli, bino-

Figure 10
**Gelman-Rubin Convergence Statistic Values
 for Beta**

End iteration of bin	-----80% interval-----		Normalized as plotted		BGR ratio
	Unnormalized of pooled chains	mean within chain	of pooled chains	mean within chain	
550	0.9177	0.9308	0.9859	1.0	0.9859
600	0.8552	0.8589	0.9188	0.9227	0.9957
650	0.8229	0.8343	0.8841	0.8963	0.9864
700	0.8126	0.8158	0.873	0.8764	0.9961
750	0.8293	0.8292	0.8909	0.8909	1.0
800	0.842	0.86	0.9045	0.9239	0.979
850	0.8499	0.873	0.9131	0.9379	0.9735
900	0.8615	0.8686	0.9256	0.9332	0.9918
950	0.8604	0.8657	0.9243	0.93	0.9939
1000	0.8602	0.8732	0.9241	0.9381	0.985
.
.
.
4500	0.878	0.879	0.9432	0.9443	0.9989
4550	0.8776	0.8778	0.9428	0.9431	0.9998
4600	0.8776	0.8778	0.9428	0.943	0.9998
4650	0.8778	0.8781	0.943	0.9433	0.9997
4700	0.8767	0.8777	0.9418	0.9429	0.9989
4750	0.876	0.8763	0.9411	0.9414	0.9997
4800	0.8767	0.8769	0.9418	0.9421	0.9998
4850	0.8765	0.8771	0.9417	0.9423	0.9994
4900	0.8764	0.8771	0.9415	0.9423	0.9992
4950	0.8759	0.8766	0.941	0.9418	0.9991
5000	0.8755	0.8767	0.9405	0.9418	0.9986

mial, Poisson, and multinomial. However, in theory any arbitrary discrete distribution can be implemented by coding the observed frequencies as a multinomial random variable. We will illustrate this procedure by applying four different models (viz., the two-component Poisson mixture, zero-inflated Poisson, generalized Poisson, and negative binomial) to the claim frequency data for 25,000 policies given in the first two columns of Table 3. The same approach should work for any of the frequency distributions found in chapter 3 of Klugman, Panjer, and Willmot (1998).

The general procedure is as follows. Suppose we have an independent sample of size N from a

discrete distribution with scalar or vector parameter θ and probability mass function $\Pr(X = x|\theta)$. Further suppose that the observations all fall into one of k mutually exclusive and exhaustive cells or classes, with n_i observations taking on the values associated with the i -th cell, for $i = 1, \dots, k$. Then, $N = n_1 + \dots + n_k$ and the likelihood function for the sample is given by

$$\prod_{i=1}^k p_i^{n_i}, \tag{6.1}$$

with p_i equal to the probability assigned to the i -th cell by the discrete distribution with probability mass function $\Pr(X = x|\theta)$. This likelihood is proportional to

$$\frac{N!}{\prod_{i=1}^k n_i!} \prod_{i=1}^k p_i^{n_i},$$

which is the probability attached to a multinomial random variable with n_i counts taking on the values associated with cell i , for $i = 1, \dots, k$, and with the cell probabilities as defined above. A stochastic node with multinomial density is defined with the declaration

$$n[] \sim \text{dmulti}(p[], N)$$

in WinBUGS. To implement a Bayesian analysis using WinBUGS, the only basic remaining is to code the cell probabilities under the discrete model being considered and assign a prior distribution to θ .

Four models are under consideration for the data in Table 3. These are the two-component Poisson mixture (TCPM) distribution with probability function

Table 3
Illustrative Claims Frequency Data

Number of Claims	Number of Policies	Estimated Predictive Mean Frequencies (SD)		
		TCPM	LP	NB
0	23,148	23,147.28 (41.2)	23,150.65 (41.4)	23,149.31 (41.4)
1	1,639	1,637.66 (39.1)	1,625.31 (39.0)	1,625.48 (39.0)
2	173	177.26 (13.3)	191.25 (13.8)	194.20 (13.9)
3	35	31.84 (5.64)	27.46 (5.24)	26.48 (5.14)
4	3	5.18 (2.27)	4.40 (2.10)	3.85 (1.96)
5	2	0.70 (0.84)	0.76 (0.87)	0.58 (0.76)
6+	0	0.09 (0.30)	0.17 (0.41)	0.11 (0.33)

$$\begin{aligned} \Pr(X = x|\lambda, \mu_1, \mu_2) \\ = \lambda \frac{e^{-\mu_1}\mu_1^x}{x!} + (1 - \lambda) \frac{e^{-\mu_2}\mu_2^x}{x!}, \\ x = 0, 1, 2, \dots, \quad (6.2) \end{aligned}$$

with $0 < \lambda < 1$, $\mu_1 > 0$, $\mu_2 > 0$, and with $\mu_1 < \mu_2$ for purposes of identifiability; the zero-inflated Poisson (ZIP) distribution with probability function

$$\begin{aligned} \Pr(X = x|\lambda, \mu) \\ = \lambda + (1 - \lambda)e^{-\mu}, \quad x = 0, \\ = (1 - \lambda) \frac{e^{-\mu}\mu^x}{x!}, \quad x = 1, 2, 3, \dots, \quad (6.3) \end{aligned}$$

with $0 < \lambda < 1$ and $\mu > 0$; the generalized or Lagrangian Poisson (LP) distribution (Consul 1989) with probability function

$$\begin{aligned} \Pr(X = x|\lambda, \mu) = \frac{\mu(\mu + \lambda x)^{x-1}}{x!} e^{-(\mu + \lambda x)}, \\ x = 0, 1, 2, \dots, \quad (6.4) \end{aligned}$$

with $0 < \lambda < 1$ and $\mu > 0$; and the negative binomial (NB) distribution with probability function

$$\begin{aligned} \Pr(X = x|r, \beta) \\ = \binom{x+r-1}{x} \left(\frac{1}{\beta+1}\right)^r \left(\frac{\beta}{\beta+1}\right)^x, \\ x = 0, 1, 2, \dots, \quad (6.5) \end{aligned}$$

with $r > 0$ and $\beta > 0$. The Bayesian analysis of the generalized Poisson distribution was recently considered by Scollnik (1995b, c, 1998)), and that of the negative binomial by Morgan and Hickman (1993) and Duvall (1999). We believe the Bayesian analyses of these models using WinBUGS will be considerably more accessible to the majority of actuarial practitioners.

In order to complete each model description, we will assume simple and relatively noninformative prior distributions for each model parameter (e.g., gamma(0.001, 0.001) densities for parameters on the semi-infinite interval $(0, \infty)$ and uniform densities for parameters on the unit interval). Of course, these can be modified by the practitioner as needed. Illustrative BUGS code for the first model (TCPM) is given below, along with the corresponding data vectors. BUGS code for all four models is contained in the file *counts.odc* available from the Web site referenced in Section 8.

CODE FOR THE TCPM MODEL

```

model;
{
  # Compute the NLL for model selection purposes.
  NLL ← - ( sum( NLLterms[] ) )
  for ( i in 1:k ) {
    NLLterms[i] ← n[i] * log( p[i] )
  }

  # Define a multinomial variable based on the cell counts.
  n[1:k] ~ dmulti( p[], N )
  N ← sum( n[] )

  # Compute the cell probabilities under the mixture model.
  for( i in 1 : k ) {
    p[i] ← lambda * p1[i] + ( 1 - lambda ) * p2[i]
  }

  # Each component's cell probabilities must sum to one.
  # This relation defines the probability in the last cell.
  p1[k] ← 1 - sum( p1[1:(k-1)] )
  p2[k] ← 1 - sum( p2[1:(k-1)] )

  # Each component is described by Poisson probabilities.
  for( i in 1 : ( k - 1 ) ) {
    log( p1[i] ) ← - mu[1] + x[i] * log( mu[1] ) - loggam( x[i] + 1 )
    log( p2[i] ) ← - mu[2] + x[i] * log( mu[2] ) - loggam( x[i] + 1 )
  }
}

```

```

CODE FOR THE TCPM MODEL (continued)

# Define prior densities for the founder nodes.
mu[1] ~ dgamma( 0.001, 0.001 ) I( , mu[2] )
mu[2] ~ dgamma( 0.001, 0.001 ) I( mu[1] , )
lambda ~ dunif( 0, 1 )
}

DATA

list( k = 7,
      x = c( 0, 1, 2, 3, 4, 5, 6 ),
      n = c( 23148, 1639, 173, 35, 3, 2, 0 ) )
    
```

The interesting parts of this BUGS code are already commented, and it is otherwise, by and large, self-explanatory. However, note that the `loggam` function calculates the log gamma of its argument and thus can be used to calculate factorials. The restriction $\mu_1 < \mu_2$, required in order to differentiate between these two parameters, is imposed with the constructs `I(, mu[2])` and `I(mu[1] ,)`. Restrictions of this type are discussed in section 3.2 of the WinBUGS user manual.

One additional feature of the BUGS code worth noting is the definition of the node NLL. The value of this node is defined equal to the negative log of the likelihood given by expression (6.1). NLL is a stochastic node as its value depends on the unobserved model parameters, and so the values it takes on can be monitored like those of any other node. Dempster (1997) suggested that one might examine the posterior distribution of the loglikelihood to assist with model selection, and we will adopt this simple strategy by monitoring the value of NLL and favoring the model

with the smallest posterior expected value for this variable. See Spiegelhalter, Best, and Carlin (1998) for modifications of this approach, which are particularly useful when the models under consideration differ in complexity (the number of free parameters).

It took WinBUGS about 19 seconds to run the TCPM model for 10,000 updates on a 150 MHz Pentium laptop PC (12 seconds on a dual 200 MHz Pentium Pro), after which we began monitoring various model parameters of interest for an additional 20,000 updates. Summary statistics are presented in Table 4. This process was repeated for the remaining three models. From an examination of the posterior summary statistics for the NLL node under the four models, we can rank the fit of the four models to the data from best to worst as TCPM, LP, NB, and ZIP. This is reassuring, as the illustrative data in Table 3 were actually simulated using a TCPM model with $\lambda = 0.925$, $\mu_1 = 0.6$, and $\mu_2 = 0.05$.

In Table 3 we list the estimated predictive

Table 4
Estimated Posterior Summary Statistics

Model	Parameter	Mean	SD	2.5%	Median	97.5%
TCPM	NLL	7387.0	1.334	7386.0	7387.0	7391.0
	lambda	0.9423	0.0209	0.892	0.9465	0.9709
	mu[1]	0.0520	0.0058	0.0393	0.0526	0.0615
	mu[2]	0.6505	0.115	0.448	0.6438	0.8921
ZIP	NLL	7400.0	1.015	7400.0	7400.0	7403.0
	lambda	0.6836	0.0185	0.6441	0.6845	0.7173
	mu	0.2679	0.0164	0.2358	0.2677	0.3008
LP	NLL	7388.0	0.9786	7387.0	7388.0	7391.0
	lambda	0.0905	0.0076	0.0763	0.0903	0.106
	mu	0.0769	0.0018	0.0735	0.0768	0.0804
NB	NLL	7389.0	1.003	7388.0	7388.0	7391.0
	r	0.419	0.0398	0.3495	0.4166	0.5063
	beta	0.2034	0.0197	0.1658	0.2028	0.243

Note: Estimates from WinBUGS.

mean frequencies for the top three models, along with the predictive standard deviation associated with each predictive cell count. These are for an independent future sample of size 25,000. The idea is as follows: If N is the size of the future sample and p_i is the predictive probability associated with the i -th cell, then the number of the future draws falling into cell i is binomial with mean Np_i and variance $Np_i(1 - p_i)$. We can estimate p_i using the posterior mean of the node $p[i]$, which is available from the output of the WinBUGS run. (Another estimate of p_i could be obtained by first clicking the *coda* button on the *Sample Monitor Tool* dialog box to dump an ASCII representation of the simulated model parameters, read them into a spreadsheet or mathematical/statistical package, and then average Equations (6.2), (6.3), (6.4), or (6.5), as the case may be, over these values in the manner of expression (2.3) (cf. Eq. 3.8). These averages would correspond to estimates of the desired predictive cell probabilities.)

A reviewer of this paper asked whether posterior probability contours or regions could be formed for a pair of monitored parameters using the many pairs of values simulated for these parameters in WinBUGS. In the case of the NB model, for example, the practitioner may want to examine posterior probability contours for the parameters r and β . This can be accomplished using Excel by first condensing the output from WinBUGS for the pair of monitored parameters of interest into a modest-sized square array (say, 20 to 30 rows and columns). Suppose that the columns group the observed values of r into adjacent and exhaustive ranges, and the rows likewise group the observed values of β (the ranges for the columns and rows will not be the same). Then the entries in this array will correspond to observed counts. The entry in the i -th column and j -th row, for instance, represents the number of simulated pairs of r and β for which r falls into the range of values associated with column i and β falls into the range of values associated with column j . Using the Chart Wizard in Excel, you will now want to select a "Surface" chart type and then one of subtypes, probably either "3-D Surface" or "Contour" (depending on your need or preference). The result will be an estimated posterior probability plot. The general procedure described above can be used with other software packages,

such as R or S-PLUS. Several other solutions are described in a short note available on the website referenced in Section 8.

7. MODELING GROUPED SIZE OF LOSS DATA IN WINBUGS

The grouped size of loss data can be modeled in WinBUGS using the procedure set forth in this section. This procedure is widely applicable, although it does require that the distribution function associated with the ground-up (Klugman, Panjer, and Willmot 1998, p. 133) loss random variable can be expressed in WinBUGS using the elementary mathematical functions available therein. This rules out the generalized (i.e., three-parameter) Pareto model, for instance, as the distribution function for this model involves the incomplete beta function. (Incidentally, the incomplete gamma function is also not supported in WinBUGS.) However, we have successfully tested the procedure with the Burr, one- and two-parameter Pareto, and Weibull models, and it should be applicable to a number of the other continuous distributions cataloged in Appendix A of Klugman, Panjer, and Willmot (1998) as well.

Suppose we have an independent sample of size N from a loss distribution with scalar or vector parameter λ and distribution function $F(x|\lambda) = \Pr(X \leq x|\lambda)$. The observations are grouped into k cells, with n_i observations falling into the i -th cell, for $i = 1, \dots, k$, and $N = n_1 + \dots + n_k$. The likelihood function for the grouped data has the form

$$\prod_{i=1}^k p_i^{n_i}, \quad (7.1)$$

where p_i denotes the likelihood contribution made by one of the losses in the i -th group. This value is a function of the distribution function and is equal to

$$p_i = \frac{F(\text{ubound}[i]|\lambda) - F(\text{lbound}[i]|\lambda)}{F(\text{utrunc}[i]|\lambda) - F(\text{ltrunc}[i]|\lambda)}, \quad (7.2)$$

where $\text{ubound}[i]$ and $\text{lbound}[i]$ are the upper and lower class limits for the i -th group, respectively,

and $utrunc[i]$ and $ltrunc[i]$ are the upper and lower truncation limits associated with the distribution for the ground-up losses in the i -th group. Often the truncation limits will be the same from group to group, but this need not be the case—say, when losses from different portfolios are combined. It is also possible that some groups will overlap. Note that the values of p_i , for $i = 1, \dots, k$, need not sum to one.

In WinBUGS the likelihood (7.1) can be generated by defining each group count n_i as a stochastic node with a binomial density having n_i trials and probability of success p_i . In other words, conditioning on n_i we proceed as if we observed n_i successes in n_i independent trials. The lines

```
for( i in 1 : k ) {
  n[i] ~ dbin( p[i], n[i] )
}
```

comprise the appropriate declaration in WinBUGS. To implement a Bayesian analysis using WinBUGS, it only remains to code the probabilities p_i under the loss model being considered and assign a prior distribution to the elements of λ .

For a context, consider the illustrative general liability payment data in Table 5. These payments are understood to be truncated from below at 5,000. The average payment in each group is given, and the empirical layer average severities were computed using the standard method (e.g., as in Klugman, Panjer, and Willmot 1998, p. 114). We will apply a Bayesian analysis to these data assuming a three-parameter Burr (α, θ, τ) loss model with distribution function

$$F(x|\alpha, \theta, \tau) = 1 - u^\alpha, \quad u^{-1} = 1 + \left(\frac{x}{\theta}\right)^\tau \quad (7.3)$$

(Klugman, Panjer, and Willmot 1998, p. 574). The values of p_i for this example can now be defined on the basis of Equation (7.2) and the illustrative data, using the distribution function (7.3). For example,

$$p_3 = \frac{F(20,000|\alpha, \theta, \tau) - F(15,000|\alpha, \theta, \tau)}{1 - F(5,000|\alpha, \theta, \tau)}.$$

For the parameters α and τ , we initially assumed independent $\text{gamma}(2, 2)$ prior densities. These priors have mean 1, standard deviation 0.707, and median 0.839. These priors are relatively informative, but not at odds with common sense as the parameters α and τ are often near 1 for size of loss data. The scale parameter θ , on the other hand, can take on quite large values. We assume a $\text{gamma}(10, 10/30,000)$ density for this parameter, with corresponding mean 30,000, standard deviation 9,486.83, and median near about 29,006. The value of 30,000 selected can thus be interpreted as a prior estimate of the central tendency of the parameter θ . Of course, these priors are all for illustration and can be modified by the practitioner when and if more a priori information is available. Note that our experience suggests that WinBUGS may have difficulty updating the complete probabilistic model if too weak a prior is assigned to the scale parameter θ .

The BUGS code for this model is given at the end of this section and is also contained in the file *grouped.odc* available from the website referenced in Section 8. In this file three deterministic nodes named NLL, eofx.exist, and eofx2.exist are defined. The first is the negative log of the likelihood (7.1). The last two are indicator variables monitoring whether or not the posterior draws of the model parameters satisfy the conditions $\alpha\tau > 1$ and $\alpha\tau > 2$. These are the conditions required for the existence of the size of loss random variable's first and second moments, respectively. The *grouped.odc* file also includes code for the (two-parameter) Pareto and Weibull models. As the Burr model provided the best fit to the data under consideration, it is the only one discussed below.

It took WinBUGS about 81 seconds to run the Burr model for 10,000 updates on a 150 MHz Pentium laptop PC (54 seconds on a dual 200

Table 5
General Liability Payments

Payment Interval	Number of Payments	Empirical Average	Empirical LAS
5,000–10,000	27	7,347	4,284
10,000–15,000	18	12,217	3,149
15,000–20,000	9	16,848	2,466
20,000–25,000	6	23,474	2,208
25,000–50,000	19	36,048	7,349
50,000–75,000	7	62,833	4,398
75,000–100,000	4	83,240	2,830
100,000–150,000	4	127,097	4,084
150,000–200,000	0	—	3,000
200,000–∞	6	490,745	17,445

MHz Pentium Pro), after which we began monitoring various model parameters of interest for an additional 20,000 updates. Summary statistics are presented in Table 6. We observe that the posterior probability associated with the event $\alpha\tau > 1$ exceeds 99%. Running the analysis over again in WinBUGS, this time incorporating the restriction $\alpha\tau > 1$, yielded the summary results in the bottom half of Table 6. For comparison's sake, we calculated the maximum likelihood estimates of the Burr model parameters as $\hat{\alpha} = 0.83196$, $\hat{\theta} = 8349.817$, $\hat{\tau} = 1.22615$. The value of the negative log likelihood in this case is 199.9. The data in Table 5 were actually generated from a Burr model with $\alpha = 1.4$, $\theta = 20,000$, and $\tau = 0.9$, and truncated from below at 5,000.

After the second WinBUGS run was completed, we clicked the *coda* button on the *Sample Monitor Tool* dialog box to dump an ascii representation of the final 20,000 triplets of simulated model parameters α , θ , and τ , after first clearing the other monitored nodes from memory. These values of α , θ , and τ can be read into a spreadsheet or mathematical/statistical package and then used for implementing additional posterior/predictive inferences. In particular, one might be interested in examining the predictive distribution of a future claim from the truncated distribution, the posterior distribution of one or more class average severities, or the posterior distribution of one or more layer average severities. These inferences can be accomplished as follows.

The predictive loss distribution, say, $f(y|Obs. Data)$, is the theoretical average of the density associated with the truncated Burr model taken

with respect to the posterior distribution of the model parameters. This can be estimated on the basis of a sample average as in expression (2.3) (cf. Eq. 3.8), that is,

$$f(y|Obs. Data) \approx \frac{1}{20,000} \sum_{t=10,001}^{30,000} \frac{f(y|\alpha^{(t)}, \theta^{(t)}, \tau^{(t)})}{1 - F(5,000|\alpha^{(t)}, \theta^{(t)}, \tau^{(t)})}$$

for $y > 5,000$. Inflation and/or policy limit modifications can be incorporated by adjusting the truncated Burr density appearing in the summation in the appropriate manner (see, e.g., Klugman, Panjer, and Willmot 1998, pp. 131–43).

For the group with payments between l and u , the theoretical average loss, conditional on the model parameters α , θ , and τ , is given by the expression

$$\frac{E(X \wedge u|\alpha, \theta, \tau) - E(X \wedge l|\alpha, \theta, \tau) + l[1 - F(l|\alpha, \theta, \tau)] - u[1 - F(u|\alpha, \theta, \tau)]}{F(u|\alpha, \theta, \tau) - F(l|\alpha, \theta, \tau)} \tag{7.4}$$

The notation $E(X \wedge u|\alpha, \theta, \tau)$ denotes the usual limited expected value function for the Burr distribution (Klugman, Panjer, and Willmot 1998, p. 574). As our simulation proceeded under the constraint $\alpha\tau > 1$, this limited expected value exists even when $u = \infty$. Expression (7.4) can be evaluated for each of the 20,000 simulated triplets of model parameters drawn from their posterior distribution. The 20,000 values of expression (7.4) represent dependent draws from its posterior distribution, and these can be plotted in the form of

Table 6
Estimated Posterior Summary Statistics

Model	Parameter	Mean	SD	2.5%	Median	97.5%
Burr	NLL	201.0	0.9348	200.1	200.7	203.5
	alpha	2.078	0.6303	0.9858	2.043	3.468
	theta	24,310	8,148	11,280	23,430	42,540
	tau	0.7773	0.2198	0.4293	0.7502	1.277
	eofx.exist	0.9908	0.09573	1.0	1.0	1.0
	eofx2.exist	0.0295	0.1692	0.0	0.0	1.0
Burr ($\alpha\tau > 1$)	NLL	201.0	0.9362	200.1	200.7	203.5
	alpha	2.079	0.6249	1.037	2.016	3.476
	theta	24,420	7,976	11,860	23,380	42,810
	tau	0.7787	0.2145	0.4246	0.7569	1.258
	eofx.exist	1.0	0.0	1.0	1.0	1.0
	eofx2.exist	0.0301	0.1709	0.0	0.0	1.0

Note: Estimates from WinBUGS.

Table 7
Class Averages (Observed, Exact, Predicted)

Payment Interval	Empirical Average	Exact Value	Using MLEs	Predictive Class Average Estimates			
				Mean	2.5%	Median	97.5%
5,000–10,000	7,347	7,295	7,273	7,248	7,142	7,243	7,383
10,000–15,000	12,217	12,340	12,305	12,316	12,268	12,314	12,373
15,000–20,000	16,848	17,367	17,336	17,351	17,324	17,350	17,381
20,000–25,000	23,474	22,385	22,360	22,374	22,355	22,373	22,393
25,000–50,000	36,048	35,426	35,090	35,270	34,970	35,279	35,525
50,000–75,000	62,833	61,086	60,954	61,002	60,765	61,011	61,194
75,000–100,000	83,240	86,421	86,365	86,363	86,163	86,369	86,527
100,000–150,000	127,097	121,797	121,741	121,629	121,015	121,648	122,144
150,000–200,000	—	172,623	172,649	172,495	172,018	172,511	172,890
200,000–∞	490,745	956,281	1,669,639	1,008,744	441,863	730,687	2,431,378

Table 8
Layer Average Severities (Observed, Exact, Predicted)

Payment Interval	Empirical LAS	Exact Value	Using MLEs	Predictive LAS Estimates			
				Mean	2.5%	Median	97.5%
5,000–10,000	4,284	3,096	4,257	4,286	4,084	4,287	4,484
10,000–15,000	3,149	2,478	3,197	3,295	2,950	3,294	3,651
15,000–20,000	2,466	2,061	2,538	2,669	2,304	2,665	3,051
20,000–25,000	2,208	1,759	2,096	2,234	1,869	2,230	2,616
25,000–50,000	7,349	6,171	7,019	7,561	5,897	7,543	9,369
50,000–75,000	4,398	3,915	4,328	4,603	3,144	4,565	6,289
75,000–100,000	2,830	2,820	3,123	3,230	1,990	3,177	4,762
100,000–150,000	4,084	3,938	4,439	4,399	2,371	4,281	6,992
150,000–200,000	3,000	2,721	3,161	2,964	1,392	2,842	5,118
200,000–∞	17,445	35,210	80,771	52,379	5,730	25,548	177,235

a histogram, or their empirical summary statistics computed and tabulated as in Table 7. The values in Table 7 under the “Exact Value” and “Using MLEs” headings correspond to expression (7.4) evaluated using the real parameter values and the maximum likelihood estimates, respectively. The layer average severity between l and u is given by

$$\frac{E(X \wedge u | \alpha, \theta, \tau) - E(X \wedge l | \alpha, \theta, \tau)}{1 - F(5,000 | \alpha, \theta, \tau)}$$

This can also be evaluated for each of the 20,000 posterior draws of model parameters. Empirical summary statistics for the different layer average severities of interest are presented in Table 8.

This example is like one in Klugman, Panjer, and Willmot (1998, section 2.8.3), where a Bayesian analysis is done using an example from a paper by Meyers (1994). These authors conduct an approximate Bayesian analysis of a single-parameter Pareto model by discretizing the prior on a small number of points. (Meyers also used a normal approximation for the distribution of the maximum likelihood estimator in place of the exact likelihood function.) Then the layer average severities were estimated. It is significant that WinBUGS allowed for a full and exact Bayesian analysis of a more complicated three-parameter Burr model with less work.

```
Code for the Burr Model
model:
{
  # Compute the NLL for model selection purposes.
  NLL <- - ( sum( NLLterms[ ] ) )
  for( i in 1:k ){
    NLLterms[i] <- n[i] * log( p[i] )
  }
}
```

```

Code for the Burr Model (continued)
# Define the likelihood terms using binomial random variables.
for( i in 1 : k)
  n[i] ~ dbin( p[i], n[i] )
}
# Determine the cell probabilities, using the definition of the model
# and also being sure to adjust the probabilities for truncation.
#
# These probabilities have the form :
#
# ( F(ubounds[i]) - F(lbounds[i]) ) / ( F(utruncs[i] - F(ltruncs[i]) ) ).
#
# The value of F(ubounds[i]) is coded as Fub[i], F(lbounds[i]) as Flb[i],
# F(utruncs[i]) as Fut[i], F(ltruncs[i]) as Flt[i].
for( i in 1 : k){
  p[i] ← ( Fub[i] - Flb[i] ) / ( Fut[i] - Flt[i] )
  Fut[i] ← utai[i] + ( 1 - utai[i] ) * ( 1 - pow( ut[i], alpha ) )
  ut[i] ← 1 / ( 1 + pow( utruncs[i] / theta, tau ) )
  Flt[i] ← 1 - pow( lt[i], alpha )
  lt[i] ← 1 / ( 1 + pow( ltruncs[i] / theta, tau ) )
  Fub[i] ← ubai[i] + ( 1 - ubai[i] ) * ( 1 - pow( ub[i], alpha ) )
  ub[i] ← 1 / ( 1 + pow( ubounds[i] / theta, tau ) )
  ubai[i] ← step( lbounds[i] - ubounds[i] )
  Flb[i] ← 1 - pow( lb[i], alpha )
  lb[i] ← 1 / ( 1 + pow( lbounds[i] / theta, tau ) )
}
# The k-th moment of the Burr distribution exists if alpha * tau > k.
# These nodes monitor associated posterior probabilities.
eofx.exist ← step( alpha * tau - 1 )
eofx2.exist ← step( alpha * tau - 2 )
# Define prior densities for the founder nodes.
# Imposing the order restrictions will ensure that the k-th exists.
alpha ~ dgamma( 2, 2 ) # I( tauinv, )
tau ~ dgamma( 2, 2 ) # I( alphainv, )
alphainv ← k / alpha
tauinv ← k / tau
k ← 1
# mx is a prior guess as to the 'middle' or 'center' of the severity
# distribution, such as its mean or median. The parameter theta has
# a gamma distribution with mean = mx and sd = mx / sqrt( 10 ).
theta ← mx * thetax
thetax ~ dgamma( 10, 10 )
}
DATA
# utai[i] is an indicator variable, indicating whether cell i is upper
# truncated at infinity. When this event occurs, we know that the value
# of the distribution function at utrunc[i] should be set equal to 1.
list( k = 9, mx = 30000 )

```

n[]	lbounds[]	ubounds[]	ltruncs[]	utruncs[]	utai[]
27	5000	10000	5000	99	1
18	10000	15000	5000	99	1
9	15000	20000	5000	99	1
6	20000	25000	5000	99	1
19	25000	50000	5000	99	1
7	50000	75000	5000	99	1
4	75000	100000	5000	99	1
4	100000	150000	5000	99	1
6	200000	99	5000	99	1

8. ADDITIONAL WORKED EXAMPLES

Illustrative BUGS/WinBUGS program files for the examples in this paper are available from this author's Web site at www.math.ucalgary.ca/~scollnik/abcd/. Files and documentation for the following additional worked examples are located there also.

Exact: Size of Loss Distributions for Exact Data. This example illustrates the Bayesian analysis of several severity models applied to exact size of loss data. These models include the gamma, inverse gamma, loggamma, lognormal, Pareto, inverse Pareto, Weibull, and inverse Weibull distributions.

Bivreg: Regression Models for Bivariate Loss Data. This example illustrates the Bayesian analysis of two possible regression models for bivariate claims data (actual losses and allocated loss adjustment expenses). The context and data are borrowed from section 2.11.3 ("Bivariate Models") in Klugman, Panjer, and Willmot (1998).

Motor: A Predictive Aggregate Claims Distribution. This example illustrates the Bayesian predictive analysis of a compound Poisson-Pareto model for aggregate claims. The model is applied to five years' worth of reinsurance data appearing in Rytgaard (1990).

Norweg: Heterogeneity in Group Life Insurance. Haastrup (1997) described an approach for modeling heterogeneity between policyholders and applied it to data arising from 1,125 groups insured through the whole or parts of the period 1982–85 by a major Norwegian insurance company. This example reproduces Haastrup's parametric fully Bayesian and parametric empirical Bayesian analyses of these data.

Credit: An Unbalanced Two-Way Crossed Classification Model. Dannenburg, Kaas, and Goovaerts (1996, section 6.5) analyze data arranged in a two-way table and corresponding to payments of a credit insurer to several banks to cover losses caused by clients defaulting on private loans. In this example, the credibility estimates developed by Dannenburg et al. are compared to the estimates arising from a fully Bayesian parametric analysis of an unbalanced two-way crossed classification model applied to the data.

Health: A Normal Model for Order-Restricted Graduation. Following the lead of Broffitt's papers on the subject (1984, 1986, 1988), Carlin (1992c)

described the Bayesian analysis of two models for graduation incorporating order restrictions on the parameters of interest. Carlin applied one of these models, one with a normal-normal distributional structure, to a set of observed health claim cost aging factors. The health example describes how to reproduce Carlin's analysis using BUGS/WinBUGS.

Broff: Increasing Ordered Graduation of Mortality Rates. Broffitt (1988) described how to implement the Bayesian analysis of a model for mortality rates subject to the requirement that the force of mortality increase with age. The broff example implements several variations on Broffitt's analysis via MCMC using BUGS/WinBUGS, for data representing male ultimate experience for a particular class of premium-paying policies.

Kimjo: Kimeldorf-Jones-Styled Bayesian Graduations. This example illustrates how the Kimeldorf-Jones (Kimeldorf and Jones 1967) graduation model can be implemented via MCMC using BUGS/WinBUGS. The main advantage here over the traditional analysis is that percentiles associated with the posterior distribution can easily be determined, and not just means. Also, order restrictions between neighboring rates can be easily introduced.

Monty: The Monty Hall, or Let's Make a Deal, Problem. This example is a consideration of simple conditional probabilities in a confusing context, with no explicit actuarial content. Coincidentally, though, this puzzle was posted (most assuredly not by this author) to discussion forums on both the Society of Actuaries and Casualty Actuarial Society websites, and to the e-mail discussion list of the Canadian Institute of Actuaries, during the spring/summer of 1999. Hundreds of responses were generated—many of which were incorrect. This example highlights the correct solution.

9. SOFTWARE AVAILABLE ON THE WWW AND OTHER RESOURCES

Virtually all of the software described in this paper is freely available on the World Wide Web. The BUGS project home page is located at www.mrc-bsu.cam.ac.uk/bugs; it links to others making available the classic BUGS and WinBUGS software. After downloading the WinBUGS software, be sure to complete and submit the request form for the key for unrestricted use of WinBUGS. Without it, you

will be unable to run models containing 100 nodes or more.

Whereas the WinBUGS software program comes with online documentation accessible under its help menu, the documentation and examples manuals for classic BUGS are distributed in the form of Postscript files. Much of the material in these manuals also relates to WinBUGS, and so these files for classic BUGS should be downloaded even if WinBUGS is of main interest. Under Windows, one way to print and view these files is with the Ghostscript and GView utilities. These utilities are freely available at www.cs.wisc.edu/~ghost/ and www.cs.wisc.edu/~ghost/aladdin/get601.html.

The *Convergence Diagnostic and Output Analysis* (CODA) software is available from the BUGS project home page. CODA was originally designed to run under the statistical computing and graphics environment S-PLUS. S-PLUS is a commercial product available from Mathsoft at www.mathsoft.com/splus/. However, Martyn Plummer has since translated CODA into R, the freeware equivalent of S-PLUS, and this is available as R-CODA at www.fis.iarc.fr/coda/. Thus, it is no longer necessary to have S-PLUS to run the diagnostics in CODA, since R can be used instead. Comprehensive information about R is available at www.ci.tuwien.ac.at/R/. Readers interested in learning more about implementing modern applied statistical methodologies with S-PLUS are encouraged to examine the text by Venables and Ripley (1999). A home page for this text maintained at www.stats.ox.ac.uk/pub/MASS3/ page links to a collection of on-line complements to the book itself. One of these, 'R' Complements, describes how to use the book with the R system.

An alternative to CODA is the Bayesian Output Analysis Program (BOA) written by Brian Smith and available at www.public-health.uiowa.edu/boa. BOA runs under both S-PLUS and R and is advertised as being faster, less memory intensive, and easier to modify than CODA. More information about BOA can be obtained at its Web site.

A large number of articles concerned with MCMC and its applications are available on the MCMC Preprint Service at www.mcs.surrey.ac.uk/Personal/S.Brooks/MCMC/. Although many of the papers at this site are quite technical in nature, a number of accessible survey papers are available there as well. Another good reference is the book *Bayesian Data Analysis* by Gelman et al. (1995),

which provides an excellent overview of current approaches to Bayesian modeling and computation in applied statistics. This readable and accessible text is highly recommended.

10. ACKNOWLEDGMENTS

This paper was supported by a grant from the Actuarial Education and Research Fund. I thank the AERF for its financial support and guidance and the members of my Project Oversight Group for their helpful comments. Special thanks are due to Stuart Klugman for suggesting numerous improvements to this paper.

REFERENCES

- BEST, N.G., SPIEGELHALTER, D.J., THOMAS, A., AND BRAYNE, C.E.G. 1996. "Bayesian Analysis of Realistically Complex Models." *Journal of the Royal Statistical Society, Series A* 159 (2): 323–42.
- BROFFIT, J.D. 1984. "A Bayes Estimator for Ordered Parameters and Isotonic Bayesian Graduation." *Scandinavian Actuarial Journal*, 231–47.
- BROFFIT, J.D. 1986. "Isotonic Bayesian Graduation with an Additive Prior," in *Advances in the Statistical Sciences: Vol. 6, Actuarial Science*, edited by MacNeill, I.B., and Umphrey, G.J.D. Boston: Reidel, pp. 19–40.
- BROFFIT, J.D. 1988. "Increasing and Increasing Convex Bayesian Graduation." *Transactions of the Society of Actuaries* 40 (1):115–48.
- BROOKS, S.P., AND GELMAN, A. 1998. "General Methods for Monitoring Convergence of Iterative Simulations." *Journal of Computational and Graphical Statistics* 7 (4):434–55.
- CARLIN, B.P. 1992a. "Analyzing Nonlinear and Non-Gaussian Actuarial Time Series." *Actuarial Research Clearing House* 1992 (1):27–60.
- CARLIN, B.P. 1992b. "State Space Modeling of Non-Standard Actuarial Time Series." *Insurance: Mathematics and Economics* 11:209–22.
- CARLIN, B.P. 1992c. "A Simple Monte Carlo Approach to Bayesian Graduation." *Transactions of the Society of Actuaries* 44:55–76.
- CARLIN, B.P., AND LOUIS, T.A. 1996. *Bayes and Empirical Bayes Methods for Data Analysis*. London: Chapman and Hall.
- CONSUL, P.C. 1989. *Generalized Poisson Distributions: Properties and Applications*. New York: Marcel Dekker.
- 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.
- DANNENBURG, D.R., KAAS, R., AND GOOVAERTS, M.J. 1996. *Practical Actuarial Credibility Models*. Belgium: Ceuterick.
- DEMPSTER, A.P. 1997. "The Direct Use of Likelihood for Significance Testing." *Statistics and Computing* 7:247–52.
- DUVALL, R.M. 1999. "A Bayesian Approach to Negative Binomial

- Parameter Estimation." *Casualty Actuarial Society Forum* (Winter):377–85.
- GELFAND, A.E., HILLS, S.E., RACINE-POON, A., AND SMITH, A.F.M. 1990. "Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling." *Journal of the American Statistical Association* 85:972–85.
- GELFAND, A.E., AND SMITH, A.F.M. 1990. "Sampling-Based Approaches to Calculating Marginal Densities." *Journal of the American Statistical Association* 85:398–409.
- GELMAN, A., CARLIN, J.B., STERN, H.S., AND RUBIN, D.B. 1995. *Bayesian Data Analysis*. New York: Chapman and Hall.
- GELMAN, A., AND RUBIN, D. 1992. "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science* 7:457–511.
- GEMAN, S., AND GEMAN, D. 1984. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721–41.
- GILKS, W.R., RICHARDSON, S., AND SPIEGELHALTER, D.J. 1996. "Introducing Markov Chain Monte Carlo," in *Markov Chain Monte Carlo in Practice*, ed. Gilks, W.R., Richardson, S., and Spiegelhalter, D.J. New York: Chapman and Hall.
- GILKS, W.R., AND WILD, P. 1992. "Adaptive Rejection Sampling for Gibbs Sampling." *Applied Statistics* 41 (2):337–48.
- HAASTRUP, S. 1997. "Comparison of Some Bayesian Analyses of Heterogeneity in Group Life Insurance." Working Paper 150, Laboratory of Actuarial Mathematics, University of Copenhagen.
- HAASTRUP, S., AND ARJAS, E. 1996. "Claims Reserving in Continuous Time: A Nonparametric Bayesian Approach." *ASTIN Bulletin* 26 (2):139–64.
- HASTINGS, W.K. 1970. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications." *Biometrika* 57: 97–109.
- HERZOG, T.N. 1996. *Introduction to Credibility Theory*. Winsted, Conn.: ACTEX Publications.
- KIMELDORF, G.S., AND JONES, D.A. 1967. "Bayesian Graduation." *Transactions of the Society of Actuaries* 19.
- KLUGMAN, S.A. 1992. *Bayesian Statistics in Actuarial Science, with Emphasis on Credibility*. Dordrecht: Kluwer Academic.
- KLUGMAN, S.A., AND CARLIN, B.P. 1993. "Hierarchical Bayesian Whittaker Graduation." *Scandinavian Actuarial Journal*, pages 183–96.
- KLUGMAN, S.A., PANJER, H.H., AND WILLMOT, G.E. 1998. *Loss Models: From Data to Decisions*. New York: John Wiley and Sons.
- LINDLEY, D.V., AND SMITH, A.F.M. 1972. "Bayes Estimates for the Linear Model." *Journal of the Royal Statistical Society, Series B*, 34:1–41.
- MAKOV, U.E., SMITH, A.F.M., AND LIU, Y.-H. 1996. "Bayesian Methods in Actuarial Science." *The Statistician* 45 (4):503–15.
- METROPOLIS, N., ROSENBLUTH, A.W., ROSENBLUTH, M.N., TELLER, A.H., AND TELLER, E. 1953. "Equations of State Calculations by Fast Computing Machines." *J. Chem. Phys.* 21:1087–92.
- MEYERS, G. 1994. "Quantifying the Uncertainty in Claim Severity Estimates for an Excess Layer When Using the Single Parameter Pareto." *Proceedings of the Casualty Actuarial Society* 81:91–122 (including discussion).
- MORGAN, I.M., AND HICKMAN, J.C. 1993. "Conjugate Bayesian Analysis of the Negative Binomial Distribution." *Actuarial Research Clearing House* 1993 (1):97–113.
- PAI, J. 1997. "Bayesian Analysis of Compound Loss Distributions." *Journal of Econometrics* 79:129–46.
- ROSENBERG, M. 1994. *A Hierarchical Bayesian Model of the Rate of Non-Acceptable In-Patient Hospital Utilization*. Ph.D. thesis, University of Michigan.
- ROSENBERG, M., AND YOUNG, V.R. 1999. "A Bayesian Approach to Understanding Time Series Data." *North American Actuarial Journal* 3 (2):130–43.
- RYTGAARD, M. 1990. "Estimation in the Pareto Distribution." *ASTIN Bulletin* 20 (2):201–16.
- SCHNIEPER, RENE. 1995. "On the Estimation of the Credibility Factor: A Bayesian Approach." *ASTIN Bulletin* 25 (2):137–51.
- SCOLLNIK, D.P.M. 1993. "A Bayesian Analysis of a Simultaneous Equations Model for Insurance Rate-Making." *Insurance: Mathematics and Economics* 12:265–86.
- SCOLLNIK, D.P.M. 1995a. "Simulating Random Variates from Makeham's Distribution and from Others with Exact or Nearly Log-Concave Densities." *Transactions of the Society of Actuaries*. 47:409–37.
- SCOLLNIK, D.P.M. 1995b. "Bayesian Analysis of Two Overdispersed Poisson Models." *Biometrics* 51:1117–26.
- SCOLLNIK, D.P.M. 1995c. "The Bayesian Analysis of Generalized Poisson Models for Claim Frequency Data Utilising Markov Chain Monte Carlo Methods." *Actuarial Research Clearing House* 1995 (1):339–56.
- SCOLLNIK, D.P.M. 1996. "An Introduction to Markov Chain Monte Carlo Methods and Their Actuarial Applications." *Proceedings of the Casualty Actuarial Society* 83:114–65.
- SCOLLNIK, D.P.M. 1998. "On the Analysis of the Truncated Generalized Poisson Distribution Using a Bayesian Method." *ASTIN Bulletin* 28 (1):135–52.
- SCOLLNIK, D.P.M. 2000. "Actuarial Modeling with MCMC and BUGS: Additional Worked Examples." Working Paper. Available at www.math.ualgary.ca/~scollnik/abcd/.
- SHEPARD, N., AND PITT, M.K. 1995. "Parameter-Driven Exponential Family Models." Technical Report, Nuffield College, Oxford.
- SMITH, A.F.M., AND ROBERTS, G.O. 1993. "Bayesian Computation via the Gibbs Sampler and Related Markov Chain Monte Carlo Methods" (with discussion). *Journal of the Royal Statistical Society, Series B* 55:3–23.
- SMITH, T.C., SPIEGELHALTER, D.J., AND PARMAR, M.K.B. 1996. "Bayesian Meta-Analysis of Randomized Trials Using Graphical Models and BUGS," in *Bayesian Biostatistics*, edited by Berry D.B., and Stangl, D.K. New York: Marcel Dekker, pp. 411–27.
- SPIEGELHALTER, D.J., BEST, N.G., AND CARLIN, B.P. 1998. "Bayesian Deviance, the Effective Number of Parameters, and the Comparison of Arbitrarily Complex Models." Research Report 98-009, Division of Biostatistics, University of Minnesota. Preprint available at www.biostat.umn.edu/~brad/.

Submitted to *Journal of the Royal Statistical Society, Series B*.

- SPIEGELHALTER, D.J., THOMAS, A., AND BEST, N.G. 1996. "Computation on Bayesian Graphical Models," in *Bayesian Statistics 5*, edited by Bernardo, J.M., Berger, J.O., Dawid, A.P., and Smith, A.F.M. Oxford: Oxford University Press, pp. 407–25.
- SPIEGELHALTER, D.J., THOMAS, A., BEST, N.G., AND GILKS, W.R. 1996. *BUGS 0.5: Bayesian Inference Using Gibbs Sampling Manual (Version ii)*. Cambridge, U.K.: MRC Biostatistics Unit.
- SPIEGELHALTER, D.J., THOMAS, A., BEST, N.G., AND GILKS, W.R. 1997. *BUGS 0.6: Bayesian Inference Using Gibbs Sampling (Addendum to Manual)*. Cambridge, U.K.: MRC Biostatistics Unit.
- TIERNEY, L. 1994. "Markov Chains for Exploring Posterior Distributions" (with discussion). *Annals of Statistics* 22:1701–62.
- VENABLES, W.N., AND RIPLEY, B.D. 1999. *Modern Applied Statistics with S-PLUS*, 3rd ed. New York: Springer.
- WILD, P., AND GILKS, W.R. 1993. "Adaptive Rejection Sampling from Log-concave Density Functions." *Applied Statistics* 42 (4): 701–9.

DISCUSSION

DAVID SPIEGELHALTER*

The team responsible for the BUGS and WinBUGS software are delighted with this paper. The BUGS project has been going for over 10 years now and originally started with the aim of disseminating MCMC methods into the biostatistics world. It is perhaps an indication of the flexibility of the methodology that WinBUGS is now being used in areas undreamt of by the developers (and funders!) of the software. Apart from the actuarial context, we have been surprised at the interest shown by, for example, researchers in ecology and wildlife management, political science, and marketing.

There is little we can add to Professor Scollnik's excellent discussion of the principles and practice of the software. Frankly, his tutorial material is better than much of what we provide, and we already recommend new users to download his papers and examples. It is worth emphasizing that BUGS and WinBUGS follow a different philosophy to most statistical packages in which pro-

cedures to apply to data are selected either from menus or with command lines. Instead, BUGS uses a *declarative* description of the probability model from which all inferences naturally flow, using probability theory in the guise of Bayes' Theorem. As Scollnik shows, this description can be either based on a graph or on the BUGS language: the latter can quickly become preferable for complex models.

There are a few revisions that I might have made to his paper. First, WinBUGS 1.3 is now available from our Web site www.mrc-bsu.cam.ac.uk/bugs. Second, we are becoming increasingly uncomfortable with using inverse-gamma (0.001, 0.001) for variance components; although these priors have usually come up with reasonable answers, they can sometimes be more influential than desired. Our preference for "minimally informative" priors is to be uniform on an interpretable quantity over a reasonable range: it is then straightforward to monitor that quantity and be clear what information the data are providing. Third, I feel confident that the analysis of the predictive loss distribution described in Section 7 could be carried out within WinBUGS without dumping out results to another package.

The issues of model criticism and comparison are hot topics in Bayesian analysis, now that fitting complex models is no longer a bottleneck. Professor Scollnik summarizes the posterior distribution of the log-likelihood arising from different models: for nonhierarchical models this has approximately a shifted and scaled chi-square distribution, which gives it some interesting properties. First, this posterior should have a variance approximately half the number of parameters in the model; this can be seen to be reasonably true in Table 4. Second, Akaike's criterion (AIC) is approximately equivalent to adding twice the mean log-likelihood to the number of parameters, which for Table 4 is minimized by TCPM—the correct model. We have developed a generalization (DIC) that can work with hierarchical models, and we hope that future versions of WinBUGS will give measures of fit and complexity for each model, which can be used as a basis for model criticism and comparison.

The software is certainly not perfect, and any new user has to come to grips with graphical modeling, declarative model descriptions, Bayesian statistics, and MCMC, as well as an interface of limited user-friendliness. In the future we are hoping to

* David Spiegelhalter, Ph.D., is Senior Scientist at the MRC Biostatistics Unit, Institute of Public Health, University Forvie Site, Robinson Way, Cambridge CB2 2SR, United Kingdom, e-mail, david.spiegelhalter@mrc-bsu.cam.ac.uk.

provide more tutorial material in order to give a gentler introduction to this somewhat daunting novelty, and Professor Scollnik's careful and generous contribution has set a very high standard for us to reach.

Additional discussions on this paper can be submitted until April 1, 2001. The author reserves the right to reply to any discussion. Please see the Submission Guidelines for Authors on the inside back cover for instructions on the submission of discussions.