2002

Modeling Size-of-Loss Distributions for Exact Data in WinBUGS

David P.M. Scollnik

University of Calgary, scollnik@math.ucalgary.ca

Follow this and additional works at: http://digitalcommons.unl.edu/joap

Part of the Accounting Commons, Business Administration, Management, and Operations Commons, Corporate Finance Commons, Finance and Financial Management Commons, Insurance Commons, and the Management Sciences and Quantitative Methods Commons

http://digitalcommons.unl.edu/joap/50

This Article is brought to you for free and open access by the Finance Department at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in Journal of Actuarial Practice 1993-2006 by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
Modeling Size-of-Loss Distributions for Exact Data in WinBUGS

David P.M. Scollnik*

Abstract†

This paper discusses how the statistical software WinBUGS can be used to implement a Bayesian analysis of several popular severity models applied to exact size-of-loss data. The particular models targeted are the gamma, inverse gamma, loggamma, lognormal, (two-parameter) Pareto, inverse (two-parameter) Pareto, Weibull, and inverse Weibull distributions. It is possible to implement additional size-of-loss models (including those for truncated data) using methods analogous to those described herein.

Key words and phrases: Bayesian, severity, Markov chain Monte Carlo, simulation

1 Introduction

1.1 Why WinBUGS?

BUGS (Bayesian inference using Gibbs sampling) is a specialized suite of statistical software packages for implementing Markov chain Monte Carlo (MCMC)-based analysis of full probability models in which all unknowns are treated as random variables. The BUGS programming language allows the user to make a straightforward specification of the full probability model under consideration. The Windows version of

*David P.M. Scollnik, A.S.A., Ph.D., is an associate professor of actuarial science and statistics at the University of Calgary. He holds a Ph.D. in statistics from the University of Toronto.

Dr. Scollnik's address is: Department of Mathematics and Statistics, University of Calgary, 2500 University Drive N.W., Calgary, Alberta, Canada, T2N 1N4. Internet address: scollnik@math.ucalgary.ca and <http://www.math.ucalgary.ca/~scollnik>

†An earlier unpublished version of this paper was supported by a grant from the Actuarial Education and Research Fund (AERF). This version was supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).
BUGS is known as WinBUGS, and is available from the BUGS Project website at: <http://www.mrc-bsu.cam.ac.uk/bugs>.

Scollnik (2001) describes how a number of different actuarial models can be implemented and analyzed in accordance with the Bayesian paradigm using the MCMC simulation method via BUGS. The MCMC method can be used to generate a dependent sequence of random draws from a Markov chain with a stationary distribution equal to the distribution associated with some probabilistic model of interest, even if the distribution is multi-dimensional with a very complicated form. A wide variety of simulation-based inferences for the model then can be developed on the basis of these dependent simulated values.

Due to its astonishing flexibility and to its ability to simplify the analysis of even extremely complicated multi-dimensional random models, the MCMC method has become increasingly popular over the last dozen or so years, as is evident in the statistical and related literature. See Scollnik (2001) for a detailed description of the MCMC method, list of references, and summary of recent actuarial applications.

1.2 Objectives

The main purpose of this paper is to show actuaries how WinBUGS can be used to implement a Bayesian analysis of several popular severity models when the data consist of the exact size of losses, i.e., before items such as deductibles and policy limits are applied. Scollnik (2001) considers only the case of grouped size-of-loss data, i.e., where losses are grouped according to size.

The particular models (distributions) studied in this paper are the gamma, inverse gamma, loggamma, lognormal, (two-parameter) Pareto, inverse (two-parameter) Pareto, Weibull, and inverse Weibull distributions. Each of these models is applied to the size-of-loss data in Table 1, after which we discuss how Bayesian posterior prediction and model checking and selection can be performed. Several authors have demonstrated that Bayesian predictions are an improvement over traditional classical statistical predictions based on conditioned maximum likelihood estimates. Bayesian predictions incorporate parameter uncertainty and prior information, which are, in effect, ignored by classical statistical predictions. See, for example, Dickson, Tedesco, and Zehnwirth (1998), Cairns (2000), and Scollnik (2002) for a discussion of this point along with some numerical comparisons.

While this paper introduces relevant WinBUGS programming tips and implementation details, Scollnik (2001) should be referenced for more detailed information about MCMC-based simulations in general,
and for specific details regarding the actual operation of the WinBUGS software in particular. The reader is assumed to have a basic working knowledge of WinBUGS. In addition, we assume the reader is familiar with the general nature of Bayesian inference. A quick overview of the Bayesian approach is as follows: Suppose the data consist of \( n \) independent observations \( x_i, i = 1, 2, \ldots, n \), from a common density function \( g(x | \alpha, \beta) \) where \( \alpha \) and \( \beta \) are random parameters (possibly vector valued) with joint prior density \( \pi(\alpha, \beta) \). From Bayes theorem and the conditional independence of losses, the posterior distribution is

\[
\pi(\alpha, \beta | x_1, \ldots, x_n) = \frac{\prod_{i=1}^{n} g(x_i | \alpha, \beta) \pi(\alpha, \beta)}{\int \prod_{i=1}^{n} g(x_i | \alpha, \beta) \pi(\alpha, \beta) \, d\alpha \, d\beta} 
\]

\[
\propto \pi(\alpha, \beta) \prod_{i=1}^{n} g(x_i | \alpha, \beta). 
\]

In the Bayesian context, inferences concerning the unknown model parameters are constructed from the posterior distribution. The posterior distribution describes all that is known about the unknown model parameters in light of the observed data and prior information. The posterior knowledge can be summarized using summary statistics such as posterior means, quantiles, and variances or summarized graphically by posterior density plots and the like. Instead of deriving the form of the posterior distribution and the value of its desired summary statistics analytically, it is common and often easier to simulate random draws from the posterior distribution and then use this posterior sample to fashion the posterior inferences (e.g., via empirical posterior summary statistics and density plots).

MCMC is one method of simulating random draws from a Markov chain with a stationary distribution equal to the posterior distribution. WinBUGS is a useful and easy-to-use software package that can be used to implement these simulations. WinBUGS does not require that the user analytically derive the posterior distribution first. Rather, the user need only specify the conditional model generating the data and the prior distribution for any unknown parameters. WinBUGS uses this information to construct, and simulate random draws from, a Markov chain with a stationary distribution equal to the correct posterior distribution for the model and data under consideration.

One advantage of a simulation-based Bayesian analysis is that the simulation results can be reused. For instance, random draws from the posterior distribution of \( (\alpha, \beta) \) can be used to estimate the posterior mean of \( \alpha \) or of \( \beta \). The same random draws, however, also can be
used to make inferences about any function of $\alpha$ and $\beta$, say $h(\alpha, \beta)$, by simply applying $h(\cdot, \cdot)$ to each random draw of $(\alpha, \beta)$ and then summarizing the results. So if $m$ draws of $(\alpha_j, \beta_j)$, $j = 1, 2, \ldots, m$, are made from the distribution $\pi(\alpha, \beta|x_1, \ldots, x_n)$, then one can use $h(\alpha_j, \beta_j)$, $j = 1, 2, \ldots, m$ to fashion inferences about the posterior distribution of $h(\alpha, \beta)$. The function of interest may even be the likelihood function, i.e.,

$$h(\alpha, \beta) = l(\alpha, \beta|x_1, \ldots, x_n) = \prod_{i=1}^{n} g(x_i|\alpha, \beta)$$

as in the example above, or the log-likelihood function.

Good discussions of Bayesian inference are provided in Klugman (1992) and Klugman et al., (1998, Section 2.8). Makov (2001) gives an overview of principal applications of Bayesian methods in actuarial science, while Scollnik (2001) includes many additional references to recent papers in actuarial science with a Bayesian perspective. Gelman, Carlin, Stern, and Rubin (1995) is an excellent non-actuarial text on Bayesian data analysis that also discusses many simulation methods, including MCMC, for use in Bayesian analyses.

Table 1

<table>
<thead>
<tr>
<th>Twenty Exact Size of Losses</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
</tr>
<tr>
<td>223</td>
</tr>
<tr>
<td>1,127</td>
</tr>
<tr>
<td>1,681</td>
</tr>
<tr>
<td>5,386</td>
</tr>
</tbody>
</table>

2 Size-of-Loss Model Specification in WinBUGS

Though WinBUGS can be used to analyze complex stochastic models, it explicitly supports only a few continuous distributions (size-of-loss models). These include the beta, chi-squared, double exponential, exponential, gamma, normal, t, (single-parameter) Pareto, uniform, and Weibull distributions. Before using any one of these distributions, however, the practitioner should note the parameterization of distributions given in the WinBUGS User Manual in order to avoid any possibility of confusion.
Though several of our models are not explicitly supported by WinBUGS, some of them can be constructed from those available in WinBUGS using mixtures of distributions and/or by applying simple transformations, such as the inverse or logarithmic transform, to the data. The remainder can be implemented using the general purpose 'ones' or 'zeroes' tricks described below in our discussion of the Pareto models.

Dempster (1974; reprinted in 1997) suggested that one might examine the posterior distribution of the loglikelihood to assist with model selection. To this end the node \( NLL \) in our WinBUGS program, which represents the negative log of the likelihood function for the observed exact size of loss values, will be used. The value of \( NLL \) depends on the unobserved model parameters, and the different values it takes as the model is updated in WinBUGS can be monitored like those of any other node. Our strategy is simple: monitor the value of \( NLL \) and choose the model with the smallest posterior mean for \( NLL \). See Spiegelhalter, Best, Carlin, and van der Linde (2001) for modifications of this approach that are particularly useful when the models under consideration differ in complexity (the number of free parameters).

In this section we will review the definitions for our targeted models, and discuss how they may be coded in WinBUGS. It should be understood that the code can be ported over to 'classic' BUGS with little effort.

### 2.1 The Gamma, Inverse Gamma, and Loggamma Models

Let \( x \) denote an observed exact size-of-loss value. In WinBUGS, the declaration

\[
    x \sim \text{dgamma}(\alpha, \beta)
\]

corresponds to a definition of the gamma model with probability density function

\[
    f(x|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \quad x > 0,
\]

with \( \alpha > 0 \) and \( \beta > 0 \).

Consider what happens if we assign a gamma distribution, as in equation (1), to the transformed variable \( y = \log(x) \) instead. The resulting pdf of \( x \) is

\[
    g(x|\alpha, \beta) = \frac{\beta^\alpha(\log(x))^{\alpha-1}}{\Gamma(\alpha)x^{\beta+1}}, \quad x > 1,
\]

with \( \alpha, \beta > 0 \), which is the definition of the density function for the loggamma model. In WinBUGS, the lines of code
describe the loggamma model defined in equation (2). Specifically, the first line states the relationship between \( x \) and \( y \) and the second line assigns the relevant density type to \( y \). The order of the two lines is actually immaterial to WinBUGS.

As before, let \( x \) denote the exact size of loss. This time assign a gamma distribution, as in equation (1), to the inverse transformed variable \( y = 1/x \). The resulting pdf of \( x \) is

\[
h(x | \alpha, \beta) = \frac{\beta^\alpha \exp(-\beta/x)}{\Gamma(\alpha)x^{\alpha+1}}, \quad x > 0, \tag{3}\]

with \( \alpha, \beta > 0 \), which is the definition of the density function for the inverse gamma model. In WinBUGS, the lines of code

\[
\begin{align*}
y &\leftarrow \frac{1}{x} \\
y &\sim \text{dgamma}(\alpha, \beta)
\end{align*}
\]

describe the inverse gamma model.

All of the models described above will need to be completed by adding prior density specifications for the model parameters. Suppose that we are interested in modeling an inverse gamma model to the twenty exact size of loss observations appearing in Table 1. Then our specification of a complete inverse gamma model in WinBUGS might proceed as shown below.

**CODE FOR THE INVERSE GAMMA MODEL**

```r
model;
{
    # Compute negative loglikelihood (NLL) in terms of x.
    NLL <- - sum( loglik[] )
    for( i in 1 : N ) {
        loglik[i] <- alpha * log( beta ) - loggam( alpha ) -
                    beta / x[i] - ( alpha + 1 ) * log( x[i] )
    }

    # Define exact size-of-loss random variables.
    for( i in 1 : N ) {
        y[i] <- 1 / x[i]
        y[i] ~ dgamma( alpha, beta )
    }
}
```
# Define 'naive' prior densities for founder nodes.
alpha ~ dgamma( 0.001, 0.001 )
beta ~ dgamma( 0.001, 0.001 )

# More informative priors, each with mean = mle and
# sd = 5 x mle. See discussion below for more details.
#
# alpha ~ dgamma( aparm1, aparm2 )
# beta ~ dgamma( bparm1, bparm2 )
#
# amle <- 0.5661338 ; aparm1 <- 0.04
# aparm2 <- aparm1 / amle
# bmle <- 193.6986 ; bparm1 <- 0.04
# bparm2 <- bparm1 / bmle

DATA
list( N = 20,
      x = c( 59, 71, 127, 217, 223, 524, 537, 1089, 1127,
             1181, 1189, 1516, 1681, 1708, 1784, 3639,
             5386, 6100, 9945, 15295 ) )

INITS
list( alpha = 2, beta = 2 )

The prior density specifications assigned to the random parameters
\( \alpha \) and \( \beta \) in the sample code above are independently gamma random
variables with common mean and variance of 1 and 1000, respectively.
This is a naive assumption. While the selection of gamma distributions
is reasonable enough for parameters that are non-negative valued (like
\( \alpha \) and \( \beta \)), it is difficult to believe that an experienced actuary cannot give
a more informed specification of the prior mean and variance. When
all else fails, it may be reasonable—or at least, be not too objection­
able from a pragmatic point of view—to assign each variable a gamma
distribution a priori with mean and standard deviation equal to its max­
imum likelihood estimate (mle) and, say, five times its mle, respectively.
Such a distribution is approximately centered in the appropriate region...
yet is still widely spread. Code implementing this mle strategy is pro-
vided above for illustration's sake. The mle values themselves were
determined outside of WinBUGS using standard techniques. When the
data are used to estimate prior parameters in this way, the analysis is
sometimes called empirical Bayes (Gelman, et al., 1995, page 123).

Illustrative WinBUGS code for the various models described above
appear in the file exact.odc available on this author's website at:
<http://www.math.ucalgary.ca/~scollnik/abcd/>. The same is
true for each of the models described in the following sections.

2.2 The Lognormal Model

In WinBUGS, the declaration

\[
x \sim \text{dnorm}( \mu, \tau )
\]

corresponds to a definition of the normal model with density function

\[
f(x|\mu, \tau) = \frac{\sqrt{\tau}}{\sqrt{2\pi}} \exp \left[ -\frac{\tau}{2} (x-\mu)^2 \right], -\infty < x < \infty,
\]

with \(-\infty < \mu < \infty\) and \(\tau > 0\). In this parameterization, \(\tau\) is called the
precision or inverse variance parameter.

Consider what happens if we assign a normal distribution as in equa-
tion (4) to the transformed variable \(y = \log(x)\). The resulting pdf of
\(x\) is

\[
g(x|\mu, \tau) = \frac{\sqrt{\tau}}{x\sqrt{2\pi}} \exp \left( -\frac{\tau}{2} [\log(x) - \mu]^2 \right),
\]

for \(x > 0\), with \(-\infty < \mu < \infty\) and \(\tau > 0\). This is the definition of
the density function for the lognormal model. In WinBUGS, the lines of
code

\[
y <- \log( x )
y \sim \text{dnorm}( \mu, \tau )
\]

describe the lognormal model defined above. Another way in which to
define the same lognormal model is with the declaration

\[
x \sim \text{dlnorm}( \mu, \tau )
\]

This appears to work for all recent versions of WinBUGS, even though
the \text{dlnorm} density is undocumented in the \text{User Manual} for some re-
cent versions of WinBUGS.
As usual, we still need to complete the model with a prior density specification and also define the data and initial values. Our complete model specification might proceed as shown below:

**CODE FOR THE LOGNORMAL MODEL**

```r
model;
{
    # Compute negative loglikelihood (NLL) in terms of x.
    NLL <- - sum( loglik[] ]

    for( i in 1 : N ) {
        loglik[i] <- - log( sqrt( 2 * Pi / tau ) ) -
                      log( x[i] ) - pow( log( x[i] ) -
                                           mu, 2 ) * tau / 2
    }

    Pi <- 3.14159265

    # Define the exact size of loss random variables.
    for( i in 1 : N ) {
        y[i] <- log( x[i] )
        y[i] ~ dnorm( mu, tau )
    }

    # Define 'naive' prior densities for the founder nodes.
    mu ~ dnorm( 0, 0.001 )
    tau ~ dgamma( 0.001, 0.001 )

    # More informative priors, each with mean = mle and
    # sd = 5 x mle. See discussion below for more details.
    #
    # mu ~ dnorm( mparm1, mparm2 )
    # tau ~ dgamma( tparm1, tparm2 )
    #
    # mmle <- 6.936106 ; mparm1 <- mmle
    # mparm2 <- 1 / pow( 5 * mmle, 2 )
    # tmle <- 0.432222 ; tparm1 <- 0.04
    # tparm2 <- tparm1 / tmle
}
```
DATA

\texttt{list( N = 20,}
\texttt{ x = c( 59, 71, 127, 217, 223, 524, 537, 1089, 1127,}
\texttt{ 1181, 1189, 1516, 1708, 1784, 3639,}
\texttt{ 5386, 6100, 9945, 15295 ) )}

INITS

\texttt{list( mu = 2, tau = 2 )}

Again, a definition of the NLL is included in the code and its values can be monitored and used to assist with model selection. Note that we adopted a prior normal distribution for the parameter \( \mu \) (i.e., instead of a gamma distribution), as the support for this parameter is the entire real number line. Included for illustration's sake, is a more informative prior density specification for each model parameter, as before, centered at that parameter's mle and with standard deviation equal to five times the mle.

2.3 The Weibull and Inverse Weibull Models

In WinBUGS, the declaration

\texttt{x \sim \text{dweib( tau, lambda )}}

corresponds to a definition of the Weibull model with density function

\[ f(x|\tau, \lambda) = \tau \lambda x^{\tau-1} \exp(-\lambda x^\tau), \quad x > 0, \]  

with \( \tau > 0 \) and \( \lambda > 0 \).

Consider what happens if we assign a Weibull distribution as in (6) to the transformed variable \( y = 1/x \). The density of \( x \) is

\[ h(x|\alpha, \beta) = \frac{\tau \lambda \exp(-\lambda/x^\tau)}{x^{\tau+1}}, \quad x > 0, \]

with \( \tau > 0 \) and \( \lambda > 0 \). This is the definition of the density function for the inverse Weibull model. In WinBUGS, the lines of code

\texttt{y <- 1 / x}
\texttt{y \sim \text{dweib( tau, lambda )}}
describe the inverse Weibull model.

As usual, we still need to complete either model with a prior density specification and also define the data and initial values. We omit a presentation of either complete model specification as they are both similar to those presented earlier in this section. As mentioned earlier, the code is available on this author’s website.

2.4 The Pareto and Inverse Pareto Models

The discussion of the Pareto and inverse Pareto models has been left for last, as the tricks used to implement these models have more general application and deserve to be emphasized. In recent versions of WinBUGS, specifically (beta) Version 1.2 (May, 1999) or later, a version of the Pareto model is available with the declaration

\[ x \sim \text{dpar}(\alpha, \theta) \]

This declaration, however, corresponds to the single-parameter Pareto model with density function

\[ f(x|\alpha, \theta) = \frac{\alpha \theta^\alpha}{x^{\alpha+1}}, \ x > \theta, \]

with \( \alpha > 0 \) and \( \theta > 0 \). This form of the Pareto distribution may be appropriate in certain instances, for example when modeling losses above a given deductible. This distribution is used in the analysis of the motor example in Section 4 of Scollnik (2000).

As the data in Table 1 have no deductible associated with them, a more sensible version of the Pareto distribution for this context would be the two-parameter model with density function

\[ f(x|\alpha, \theta) = \frac{\alpha \theta^\alpha}{(x + \theta)^{\alpha+1}}, \ x > 0, \] (8)

with \( \alpha > 0 \) and \( \theta > 0 \). A related distribution is the inverse Pareto model which arises in the expected manner by assigning a Pareto distribution as in equation (8) to the transformed variable \( y = 1/x \). The density of \( x \) is

\[ h(x|\alpha, \theta) = \frac{\alpha \theta^\alpha x^{-1}}{(1 + x \theta)^{\alpha+1}}, \ x > 0, \] (9)

with \( \alpha > 0 \) and \( \theta > 0 \). Although neither of these distributions is explicitly supported in WinBUGS, we are aware of two ways in which to implement them.
The first is based on a trick that was originally found on the FAQ (frequently asked questions) page of the BUGS website at <http://www.mrc-bsu.cam.ac.uk/bugs>. This 'ones' trick now appears in Section 3.2 of the WinBUGS User Manual and its discussion there reads as follows:

Suppose your data is y (of length n) and you want to fit the model \( p(y) = f(y, t) \) where t are the unknown parameters and f is the formula of the density that is not currently handled by BUGS.

The trick is to create a new vector 'ones', that comprises just 1's and is of length n (note the use of the data transformation ability described in Section 3.7). Then use the BUGS code:

```plaintext
for(i in 1 : n) {
    ones[i] <- 1
    ones[i] ~ dbern(p[i])
    p[i] <- f(y[i], t) / K
}
```

where K is a sufficiently large constant to ensure that all sampled values of p[i] are less than one. This should provide a likelihood term proportional to f(y, t).

To illustrate, in the case of a random sample from the two-parameter Pareto model, with density function equation (8), we would assign

\[
p[i] \leftarrow \alpha \cdot \frac{\theta, \alpha}{x[i] + \theta, \alpha + 1}
\]

When using the inverse Pareto model, with density function equation (9), we would use the lines of code

\[
y[i] \leftarrow 1 / x[i]
p[i] \leftarrow \alpha \cdot \frac{\theta, \alpha}{y[i] + \theta, \alpha + 1} / \frac{1}{x[i]^2}
\]

It should be apparent to the reader that this 'ones' trick can be used to construct the likelihood function for a sample drawn from any continuous distribution, including truncated models, provided that the relevant density function may be expressed using the operators +, -, *, /, and the standard mathematical functions (e.g., exp, log, abs, and sqrt).
Scollnik: Modeling Size-of-Loss Distributions

listed in Table I of the WinBUGS User Manual. Incidentally, as the likelihood function for the observed data is the product of the p[i] terms, it will be an easy matter to calculate the node NLL, as it is simply equal to the negative logarithm of this product.

A variation of the 'ones' trick was first suggested to us through a public communication by Serguei N. Smirnov on an email discussion list devoted to BUGS at: <http://www.jiscmail.ac.uk/lists/bugs.html>. Smirnov's idea was to modify the 'ones' trick by using an exponential distribution in place of the Bernoulli as follows

for(i in 1 : n) {
    zeroes[i] <- 0
    zeroes[i] ~ dexp( p[i] )
    p[i] <- f(y[i], t)
}

The advantage to Smirnov's method is that a large constant K need no longer be specified.

The second method with which to implement the two-parameter Pareto and inverse Pareto models relies on the observation that a two-parameter Pareto random variable can be defined as a mixture of two gamma random variables. (See, for example, Hogg and Klugman, 1984, page 54.) Specifically, if the distribution of x given T is $\Gamma(1, \tau)$ and the distribution of T given (X and e) is $\Gamma(\alpha, \theta)$, then the distribution of x given (X and e) has the density function equation (8). To code this relationship in WinBUGS, we would use the lines of code

$$x[i] \sim \text{dgamma}(1, \tau[i])$$
$$\tau[i] \sim \text{dgamma}(\alpha, \theta)$$

It is important to note that each observation requires its own mixing parameter tau[i]; see Section 2.7.3.4 of Klugman et al., (1998), for a further discussion of this point and of mixture modeling in general. The lines

$$y[i] \leftarrow 1 / x[i]$$
$$y[i] \sim \text{dgamma}(1, \tau[i])$$
$$\tau[i] \sim \text{dgamma}(\alpha, \theta)$$

serve to define the inverse Pareto model.

Other distributions with interpretations as mixture models may be implemented in an analogous manner. Although hard and fast advice
is difficult to give, our experience suggests that the 'ones' and 'zeroes' tricks lead to complete model specifications which update more quickly and also take fewer updates to converge in WinBUGS. The mixture modeling approach is still valuable, though, as it may be used to generate posterior predictive draws from the two-parameter Pareto models described above. This is discussed below.

No matter which of the the methods we adopt, we still need to complete the model with a prior density specification and also define the data and initial values in the usual way. Illustrative code for a complete model specification appears in the aforementioned exact.odc file at the author's website.

3 Posterior Predictive Draws and Model Checks

The preceding discussion described how a variety of size of loss models can be implemented in WinBUGS. By examining the values of the NLL node associated with each model, selection between competing models is facilitated. Models with low values of the NLL are generally preferred, ceteris paribus. Although examination of the values taken by the NLL node will provide some guidance as to how well a particular model fits a given data set, it does not tell the complete story. Model checking is also important and is discussed in Gelman et al., (1995, especially Chapters 6 and 18). One method presented by these authors, that of posterior predictive checks, involves drawing simulated values from the posterior predictive distribution of replicated data and comparing these samples to the observed data (Gelman, et al., 1995, pages 162-174). Systematic differences between the simulations and observed data indicate potential failings of the model.

The method of posterior predictive checks is fairly simple to implement using WinBUGS. The first step is to generate a replicated sample from the same model (i.e., from the same distribution and with the same model parameter values) that is assumed to have generated the observations at hand. The replicated sample is of the same size as the original and would use the identical covariate values if the model happened to contain explanatory variables. Often, this replicated sample is easily obtained by essentially duplicating the code used to model the original observations. For example, suppose we were assuming a loggamma model as in equation (2) for the data in Table 1 and so had specified

\[ y[i] \leftarrow \log(x[i]) \]
y[i] ~ dgamma( alpha, beta)

Then the replicated data would be defined analogously with the lines

\[ x_{\text{rep}[i]} \leftarrow \exp( y_{\text{rep}[i]} ) \]
\[ y_{\text{rep}[i]} \sim \text{dgamma}(\alpha, \beta) \]

Note the transformations are now coded from \( y_{\text{rep}[i]} \) to \( x_{\text{rep}[i]} \) as the former is logically defined in advance of the latter. The same idea works for all of the distributions previously discussed except the Pareto and inverse Pareto. As these two are not explicitly supported in WinBUGS, we utilize the mixture model interpretation of the Pareto distribution in order to generate the predictive draws. In the case of the Pareto model with density function equation (8), this is accomplished with the lines of code

\[ x_{\text{rep}[i]} \sim \text{dgamma}(1, \tau_{\text{rep}[i]}) \]
\[ \tau_{\text{rep}[i]} \sim \text{dgamma}(\alpha, \theta) \]

whereas the code segment

\[ x_{\text{rep}[i]} \leftarrow 1 / y_{\text{rep}[i]} \]
\[ y_{\text{rep}[i]} \sim \text{dgamma}(1, \tau_{\text{rep}[i]}) \]
\[ \tau_{\text{rep}[i]} \sim \text{dgamma}(\alpha, \theta) \]

would be appropriate if we were assuming the inverse Pareto model with density function equation (9).

The next step is to compare the simulated values from the posterior predictive distribution to the observed data. This may be accomplished using graphical summaries or through the use of test quantities. Here, we will briefly describe the latter approach and direct the reader to Figures 6.3-6.5, 13.2, and 16.2-16.3 in Gelman et al., (1995) for examples of the former. In any case, the reader is once again referred to Gelman et al., (1995, pages 162-174) for a more extensive discussion of posterior predictive model checking.

Let \( x = (x_1, x_2, \ldots, x_n) \) be the observed data, let \( \theta \) be the vector of unknown model parameters, and let \( x_{\text{rep}} \) be the replicated data as defined above that might have been observed if a new sample of observations were sampled from the same distribution and with the same model parameter values used to generate \( x \). A test quantity, also called a discrepancy measure, \( T(x, \theta) \) is a scalar summary of the parameters and data that is used as a standard when comparing the observed data to the replicate simulations. The possibilities include, but are certainly not limited to,
\[ T(x, \theta) = \min(x_i), \]
\[ T(x, \theta) = \sum x_i, \quad \text{and} \]
\[ T(x, \theta) = \bar{x} - E(X_i|\theta). \]

Test quantities are suggested by the problem context, and some examples are considered below. Any given discrepancy measure can also be calculated using the posterior simulations of \((x^{\text{rep}}, \theta)\) in order to obtain values we denote \(T(x^{\text{rep}}, \theta)\).

The Bayesian posterior predictive \(p\)-value is defined as the probability that the replicated data could be more extreme than the observed data, as measured by the test quantity and given the assumed model. Mathematically, we write

\[ \text{Bayes } p\text{-value} = P[T(x^{\text{rep}}, \theta) \geq T(x, \theta)|x], \quad (10) \]

with the probability understood to be taken over the joint posterior distribution of \((x^{\text{rep}}, \theta)\); that is, over the joint conditional distribution of \((x^{\text{rep}}, \theta)\) given the observed data. WinBUGS will automatically generate random draws from this posterior distribution, provided that the replicated data were defined in WinBUGS as described earlier in this section.

When the tail-area probability equation (10) is close to 0 or 1 for some meaningful test quantity, the assumed model is suspect. In this case, the definition of the discrepancy measure might suggest how the model can be improved. For example, suppose \(T(x, \theta) = \max(x_i)\) and the Bayes \(p\)-value is approximately 0.84. This says that nearly 17 times out of 20 the assumed model will generate a predictive sample containing a maximum value greater than that observed in the original sample. The practitioner will have to decide whether or not this is a crucial model failing, given the problem context. It needn’t be, say, if the practitioner’s real interest is in developing inferences with respect to the distribution of total future claims and the test quantity \(T(x, \theta) = \sum x_i\) happens to yield a Bayes \(p\)-value close to 0.50. But if it is judged to be a crucial failing, the practitioner might try a model with a thinner tail. As an alternative course of action, the practitioner may keep the assumed model for the original sample but impose a reasonable a priori upper bound on each predictive draw. See Gelman et al., (1995, pages 463–468) for an example of this sort.

When inference is proceeding on the basis of a MCMC simulation, as with WinBUGS, it is easy to estimate the Bayes \(p\)-value by monitoring
the values taken on by an indicator variable assigned equal to 1 when \( T(x^\text{rep}, \theta) \geq T(x, \theta) \), and 0 otherwise. The average of these values is an estimate of equation (10). In the particular case of the exact size of losses in Table 1, it may make sense to monitor the minimum, maximum, and total losses in each of the replicated data sets. The WinBUGS code following below could be used to implement the appropriate posterior predictive checks. The approximate Bayes \( p \)-values are equal to the estimated posterior means of the nodes \( p.\text{repmin} \), \( p.\text{repmax} \), and \( p.\text{repsum} \).

**ILLUSTRATIVE CODE FOR POSTERIOR PREDICTIVE CHECKS**

```
# Use the step function to define indicator variables
# with which to estimate the Bayes p-values. The
# step function is equal to 0 (1) when its argument
# is less than (greater than or equal to) zero.
# So, for example, p.repmin <- step( x.repmin - x.min
# is assigned the value of 1 if x.repmin >= x.min.

p.repmin <- step( x.repmin - x.min )
p.repmax <- step( x.repmax - x.max )
p.repsum <- step( x.repsum - x.sum )

# Calculate min, max, and total of observed data.

x.min <- ranked( x[, 1 ] )
x.max <- ranked( x[, N ] )
x.sum <- sum( x[] )

# Calculate min, max, and total of replicated data.

x.repmin <- ranked( x.rep[, 1 ] )
x.repmax <- ranked( x.rep[, N ] )
x.repsum <- sum( x.rep[] )
```

4 Fitting the Models to the Data in Table 1

Finally, we are ready to apply the models and methods discussed in this section to the exact size of loss data in Table 1. In each case we have assumed independent prior distributions for all model parameters. Positive model parameters were assigned prior gamma distribu-
tions and the lognormal model's real parameter $\mu$ was assigned a normal prior distribution. Each model parameter had its prior distribution assigned a mean and standard deviation equal to its mle and five times its mle, respectively. These distributions are clearly informative, but we would argue only very weakly so. In practice, the actuarial practitioner often will be able to ascertain more informative prior distributions than these from past experience.

Each model compiled readily in WinBUGS and updated fairly quickly. The loggamma model was typical of the majority and took three seconds to burn-in for 5000 updates and 25 seconds to run for an additional 20,000 iterations on a dual 200 MHz Pentium Pro PC. The Pareto and inverse Pareto models were slowest, and each took about twice as long to run as the others. Summary statistics for the eight models appear in Table 2. The estimates from WinBUGS are based on the final 20,000 of the 25,000 iterations performed for each model. On the basis of the summary statistics for the NLL node, the lognormal and Pareto models rank as our first and second choices. The posterior predictive checks we monitored give us no reason to suspect either model.

Note that WinBUGS will always output estimated posterior means and SDs for the nodes $x_{.\text{repmin}}$, $x_{.\text{repmax}}$, and $x_{.\text{repsum}}$ using sample moment calculations applied to the 20,000 simulated values of each, even though the corresponding theoretical posterior predictive moments may not exist under the assumed model. In these cases, the posterior mean and SD estimates should be ignored. If it is believed a priori that certain predictive moments do exist, then the model parameters should be constrained appropriately.
### Table 2
Estimated Posterior Summary Statistics

<table>
<thead>
<tr>
<th>Model Parameters</th>
<th>Mean</th>
<th>SD</th>
<th>2.5%</th>
<th>Median</th>
<th>97.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamma</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NLL</td>
<td>177.3</td>
<td>1.03</td>
<td>176.3</td>
<td>177.0</td>
<td>180.1</td>
</tr>
<tr>
<td>alpha</td>
<td>0.6241</td>
<td>0.1699</td>
<td>0.341</td>
<td>0.6075</td>
<td>0.9989</td>
</tr>
<tr>
<td>beta</td>
<td>2.35E-4</td>
<td>9.23E-5</td>
<td>8.72E-5</td>
<td>2.23E-4</td>
<td>4.45E-4</td>
</tr>
<tr>
<td>p.repmin</td>
<td>0.3686</td>
<td>0.4824</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>p.repmax</td>
<td>0.1905</td>
<td>0.3927</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>p.repsum</td>
<td>0.4953</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>x.repmin</td>
<td>82.57</td>
<td>131.9</td>
<td>0.02948</td>
<td>31.4</td>
<td>466.0</td>
</tr>
<tr>
<td>x.repmax</td>
<td>11160.0</td>
<td>7002.0</td>
<td>3519.0</td>
<td>9392.0</td>
<td>2.9E+4</td>
</tr>
<tr>
<td>x.repsum</td>
<td>58390.0</td>
<td>26510.0</td>
<td>22660.0</td>
<td>53140.0</td>
<td>123200.0</td>
</tr>
</tbody>
</table>

| Inverse Gamma†  |        |        |        |        |       |
| NLL              | 179.0  | 1.045  | 178.0  | 178.7  | 181.8 |
| alpha            | 0.5504 | 0.1476 | 0.3059 | 0.536  | 0.8788|
| beta             | 188.0  | 75.82  | 67.81  | 178.5  | 359.8 |
| p.repmin         | 0.7805 | 0.4139 | 0.0    | 1.0    | 1.0   |
| p.repmax         | 0.7222 | 0.4479 | 0.0    | 1.0    | 1.0   |
| p.repsum         | 0.7859 | 0.4102 | 0.0    | 1.0    | 1.0   |
| x.repmin         | 106.0  | 63.92  | 27.7   | 92.03  | 263.1 |
| x.repmax         | 9.8E+11| 9.9E+13| 2482.0 | 50780.0| 1.117E+8|
| x.repsum         | 6.4E+13| 6.9E+15| 16160.0| 214500.0| 6.082E+8|

† Note the discussion in the main text concerning the existence of the theoretical posterior predictive moments for the nodes x.repmin, x.repmax, and x.repsum.
### Table 2 (Continued)

**Estimated Posterior Summary Statistics**

<table>
<thead>
<tr>
<th>Model Parameters</th>
<th>Mean</th>
<th>SD</th>
<th>2.5%</th>
<th>Median</th>
<th>97.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loggamma</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NLL</td>
<td>177.0</td>
<td>0.9604</td>
<td>176.0</td>
<td>176.7</td>
<td>179.6</td>
</tr>
<tr>
<td>alpha</td>
<td>18.52</td>
<td>5.826</td>
<td>9.211</td>
<td>17.72</td>
<td>31.67</td>
</tr>
<tr>
<td>beta</td>
<td>2.669</td>
<td>0.8512</td>
<td>1.311</td>
<td>2.553</td>
<td>4.575</td>
</tr>
<tr>
<td>p.repmin</td>
<td>0.7228</td>
<td>0.4476</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>p.repmax</td>
<td>0.5332</td>
<td>0.4989</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>p.repsum</td>
<td>0.6401</td>
<td>0.48</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>x.repmin</td>
<td>117.2</td>
<td>91.01</td>
<td>18.08</td>
<td>93.77</td>
<td>352.6</td>
</tr>
<tr>
<td>x.repmax</td>
<td>175700.0</td>
<td>4.432E+6</td>
<td>2672.0</td>
<td>17010.0</td>
<td>590800.0</td>
</tr>
<tr>
<td>x.repsum</td>
<td>608500.0</td>
<td>2.988E+7</td>
<td>17600.0</td>
<td>73250.0</td>
<td>1.432E+6</td>
</tr>
<tr>
<td><strong>Lognormal</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NLL</td>
<td>176.5</td>
<td>1.017</td>
<td>175.5</td>
<td>176.2</td>
<td>179.2</td>
</tr>
<tr>
<td>mu</td>
<td>6.933</td>
<td>0.365</td>
<td>6.22</td>
<td>6.934</td>
<td>7.66</td>
</tr>
<tr>
<td>tau</td>
<td>0.4105</td>
<td>0.1335</td>
<td>0.1969</td>
<td>0.3956</td>
<td>0.712</td>
</tr>
<tr>
<td>p.repmin</td>
<td>0.6247</td>
<td>0.4842</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>p.repmax</td>
<td>0.4294</td>
<td>0.495</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>p.repsum</td>
<td>0.5647</td>
<td>0.4958</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>x.repmin</td>
<td>109.6</td>
<td>102.0</td>
<td>6.599</td>
<td>81.54</td>
<td>380.4</td>
</tr>
<tr>
<td>x.repmax</td>
<td>32870.0</td>
<td>161600.0</td>
<td>2764.0</td>
<td>12900.0</td>
<td>159100.0</td>
</tr>
<tr>
<td>x.repsum</td>
<td>1.02E+5</td>
<td>310700.0</td>
<td>18770.0</td>
<td>59560.0</td>
<td>397900.0</td>
</tr>
</tbody>
</table>

† Note the discussion in the main text concerning the existence of the theoretical posterior predictive moments for the nodes x.repmin, x.repmax, and x.repsum.
### Table 2 (Continued)
Estimated Posterior Summary Statistics

<table>
<thead>
<tr>
<th>Model Parameters</th>
<th>Estimates from WinBUGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td><strong>Pareto</strong>†</td>
<td></td>
</tr>
<tr>
<td>NLL</td>
<td>176.7</td>
</tr>
<tr>
<td>alpha</td>
<td>3.484</td>
</tr>
<tr>
<td>theta</td>
<td>6827.0</td>
</tr>
<tr>
<td>p.repmin</td>
<td>0.6236</td>
</tr>
<tr>
<td>p.repmax</td>
<td>0.3059</td>
</tr>
<tr>
<td>p.repsum</td>
<td>0.5065</td>
</tr>
<tr>
<td>x.repmin</td>
<td>126.8</td>
</tr>
<tr>
<td>x.repmax</td>
<td>3.585E+7</td>
</tr>
<tr>
<td>x.repsum</td>
<td>4.725E+7</td>
</tr>
<tr>
<td><strong>Inverse Pareto</strong>†</td>
<td></td>
</tr>
<tr>
<td>NLL</td>
<td>177.1</td>
</tr>
<tr>
<td>alpha</td>
<td>1.536</td>
</tr>
<tr>
<td>theta</td>
<td>0.002069</td>
</tr>
<tr>
<td>p.repmin</td>
<td>0.593</td>
</tr>
<tr>
<td>p.repmax</td>
<td>0.5238</td>
</tr>
<tr>
<td>p.repsum</td>
<td>0.6594</td>
</tr>
<tr>
<td>x.repmin</td>
<td>106.7</td>
</tr>
<tr>
<td>x.repmax</td>
<td>96650.0</td>
</tr>
<tr>
<td>x.repsum</td>
<td>219100.0</td>
</tr>
</tbody>
</table>

† Note the discussion in the main text concerning the existence of the theoretical posterior predictive moments for the nodes x.repmin, x.repmax, and x.repsum.
Table 2 (Continued)
Estimated Posterior Summary Statistics

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Estimates from WinBUGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td><strong>Weibull</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLL</td>
<td>176.8</td>
</tr>
<tr>
<td></td>
<td>alpha</td>
<td>0.7236</td>
</tr>
<tr>
<td></td>
<td>beta</td>
<td>0.006146</td>
</tr>
<tr>
<td></td>
<td>p.repmin</td>
<td>0.4091</td>
</tr>
<tr>
<td></td>
<td>p.repmax</td>
<td>0.2338</td>
</tr>
<tr>
<td></td>
<td>p.repsum</td>
<td>0.4978</td>
</tr>
<tr>
<td></td>
<td>x.repmin</td>
<td>84.74</td>
</tr>
<tr>
<td></td>
<td>x.repmax</td>
<td>12590.0</td>
</tr>
<tr>
<td></td>
<td>x.repsum</td>
<td>60370.0</td>
</tr>
<tr>
<td><strong>Inverse</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Weibull</strong> †</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLL</td>
<td>178.1</td>
</tr>
<tr>
<td></td>
<td>tau</td>
<td>0.6671</td>
</tr>
<tr>
<td></td>
<td>lambda</td>
<td>71.95</td>
</tr>
<tr>
<td></td>
<td>p.repmin</td>
<td>0.745</td>
</tr>
<tr>
<td></td>
<td>p.repmax</td>
<td>0.6937</td>
</tr>
<tr>
<td></td>
<td>p.repsum</td>
<td>0.7718</td>
</tr>
<tr>
<td></td>
<td>x.repmin</td>
<td>109.5</td>
</tr>
<tr>
<td></td>
<td>x.repmax</td>
<td>1.796E+8</td>
</tr>
<tr>
<td></td>
<td>x.repsum</td>
<td>8.665E+9</td>
</tr>
</tbody>
</table>

† Note the discussion in the main text concerning the existence of the theoretical posterior predictive moments for the nodes x.repmin, x.repmax, and x.repsum.
Scollnik: Modeling Size-of-Loss Distributions

In the case of the Pareto model, for example, the restrictions \( \alpha > 1 \) and \( \alpha > 2 \) would need to be imposed in order to ensure the existence of a finite posterior predictive mean and variance, respectively (Klugman et al., 1998, page 575). The posterior probability attached to these restrictions can be checked by monitoring the frequency with which they arise in the MCMC simulation-based analysis of the unconstrained model. This procedure is illustrated in the analysis of the motor example in Section 4 of Scollnik (2000), and in the analysis of the grouped example ("Modeling Grouped Size of Loss Data in WinBUGS") in Section 7 of Scollnik (2001).

5 Implementing Predictive Inference

Suppose that \( f(x|\psi) \) is the loss model responsible for generating the original observed losses, and that \( g(y|\psi) \) is the loss model responsible for generating the losses that will be observed in the next period. Given the model parameters, \( \psi \), we assume that the past and future losses are mutually independent. The predictive density \( h(y|x) \) associated with a future loss is defined as the theoretical average of \( g(y|\psi) \) taken with respect to the posterior distribution of the model parameters. That is,

\[
h(y|x) = \int g(y|\psi)p(\psi|x)d\psi.
\]

In Section 1.2, we discussed how to simulate a dependent sequence of random draws from a posterior distribution of model parameters, like \( p(\psi|x) \), using WinBUGS. Let us assume that WinBUGS has been used in this manner to generate a sequence of such draws, which we will denote as \( \psi^{(t)} \), for \( t = m, \ldots, n \) (\( m = 5,001 \) and \( n = 25,000 \), in the example above). Provided that the model parameter vector \( \psi \) was monitored in WinBUGS over these \( n - m + 1 \) iterations, we can click the Coda button on the Sample Monitor Tool dialog box to dump an ASCII (text) representation of its simulated values. These can be read into a spreadsheet or mathematical/statistical package and then used to estimate equation (11) on the basis of the ergodic sample average

\[
h(y|x) \approx \frac{1}{n - m + 1} \sum_{t=m}^{n} g(y|\psi^{(t)}).
\]

This is easily evaluated for any value(s) of \( y \) desired. Note that the conditional model \( g(y|\psi) \) needn't be identical to the model \( f(x|\psi) \) responsible for generating the original observed losses. In particular,
it may be modified in accordance with the effect(s) of inflation and/or policy limit modifications. For instance, if the original model was

\[ f(x|\alpha, \theta) \sim \text{Pareto}(\alpha, \theta) \]

and inflation through the next period was 100r percent, then the conditional loss model at the end of this period would be

\[ g(y|\alpha, \theta) \sim \text{Pareto}(\alpha, [1 + r]\theta) , \]

as noted in Table 5.1 of Hogg and Klugman (1984, page 180). To simulate a variable representing a predictive draw from a loss model, use lines of code patterned after Section 3. For the Pareto loss model with inflation, for example, we would code

\[
\begin{align*}
y & \sim \text{dgamma}(1, \tau.y) \\
\tau.y & \sim \text{dgamma}(\alpha, \theta.y) \\
\theta.y & \leftarrow (1 + r) \times \theta
\end{align*}
\]

The value of r would be set as a constant, loaded as part of the data list.

Scollnik (2002) provides a detailed illustration of predictive inference constructed via WinBUGS in the context of two possible regression models for a set of bivariate claims data (of the actual loss and allocated loss adjustment expense variety) and develops predictive forecasts of the total loss distributions under these two models for two different coverages. The reader is directed to this example for further insight into the predictive modeling process.

6 Concluding Remarks

This paper discusses how a number of different actuarial models for exact size-of-loss data can be implemented and analyzed in accordance with the Bayesian paradigm using WinBUGS. It does not, however, discuss how the models themselves are developed and selected for consideration, nor does it discuss how the likelihood function is specified when the sample data are incomplete—for instance, when there are left-truncated (due to deductibles) and right-censored losses (i.e., capped by policy limits).\(^1\) Provided that the resulting likelihood function can be defined using the mathematical operators available in WinBUGS, the

\(^1\)These issues are discussed in Klugman et al., (1998), and Guihai (2001).
size of loss model always can be coded in WinBUGS by using the 'ones' or 'zeroes' tricks described in Section 2.4 above.

Another topic not discussed is the data preparation steps that may be required prior to model fitting. In practice, the data must be examined and corrected for data entry and reporting errors. Some of the data may belong to more current periods and some to older periods, so that some trending may be required to bring the data to current levels. In some contexts, it is also possible that some losses have not completely settled so that some adjustments to ultimate values also may be required. Some of these issues are discussed in McClenahan (1996) and Brown and Gottlieb (2001). It is also possible for some or all of these steps to be included as part of the complete probabilistic model. For instance, a random component representing missing data (e.g., reported but not settled claim amounts) could be included in the model and then the complete model be analyzed using a Bayesian method. See Ntzoufras and Dellaportas (2002) for a discussion and analysis of four such models.

References


