UNIVERSIDAD CARLOS III DE MADRID working papers

Working Paper 02-04
Statistics and Econometrics Series 02
February 2002

Departamento de Estadística y Econometría
Universidad Carlos III de Madrid
Calle Madrid, 126
28903 Getafe (Spain)
Fax (34) 91 624-98-49

BAYESIAN INFERENCE FOR FAULT BASED SOFTWARE RELIABILITY MODELS GIVEN SOFTWARE METRICS DATA

M. T. Rodríguez Bernal and M. P. Wiper*

Abstract

We wish to predict the number of faults N and the time to next failure of a piece of software. Software metrics data are used to estimate the prior mean of N via a Poisson regression model. Given failure time data and a some well known fault based models for interfailure times, we show how to sample the relevant posterior distributions via Gibbs sampling using the package Winbugs. Our approach is illustrated with an example.

Keywords: Software reliability, software metrics, Bayesian inference, Gibbs sampling.

*Wiper, Departamento de Estadística y Econometría, Universidad Carlos III de Madrid, Tfno: 91-6249852, e-mail: mwiper@est-econ.uc3m.es; Rodríguez, Departamento de Estadística y Econometría, Universidad Carlos III de Madrid, e-mail: mrodrigu@est-econ.uc3m.es.

BAYESIAN INFERENCE FOR FAULT BASED SOFTWARE RELIABILITY MODELS GIVEN SOFTWARE METRICS DATA

M. T. RODRÍGUEZ BERNAL
Departamento de Estadística y Econometría , Universidad Carlos III de Madrid, Calle Madrid 126 Getafe, (Madrid) 28903, Spain

and

M. P. WIPER
Departamento de Estadística y Econometría , Universidad Carlos III de Madrid, Calle Madrid 126 Getafe, (Madrid) 28903, Spain

We wish to predict the number of faults N and the time to next failure of a piece of software. Software metrics data are used to estimate the prior mean of N via a Bayesian, Poisson regression model. Given failure time data and a some well known fault based models for interfailure times, we show how to sample the relevant Bayesian posterior distributions via Gibbs sampling using the package *Winbugs*. Our approach is illustrated with an example.

Keywords: Software reliability, software metrics, Poisson regression, Bayesian inference, Gibbs sampling

1. Introduction

Various classes of software reliability models have been studied in the literature. Firstly, a large number of models have been developed for the prediction of the times between successive failures of a program, $T_1, T_2, \ldots$ where it is usually assumed that the program is modified after each failure is observed. Good reviews of this area are given by e.g. Littlewood[1] and Singpurwalla and Wilson[2,3]. One approach to interfailure time modelling assumes that failures are directly related to the number of faults in a piece of software. Specific models based on this idea have been introduced by Jelinski and Moranda[4], Schick and Wolverton[5] and Littlewood[6]. For alternative approaches not based on fault numbers, see Singpurwalla and Wilson[2].

A second approach to software reliability, which is often used in the software development process, involves the measurement of various characteristics of the software called software metrics. Perhaps the best known such metrics are the simple *lines of code* measure and *McCabes cyclomatic complexity*[7]. It is supposed that the software metric values are related to aspects of software quality and, in particular, there have been various articles which have attempted to relate software metrics to the number of faults in a program via, for example, regression type models. For examples, see Akiyama[8], Compton and Withrow[9], Wiper et al[10] and Evanco[11]. A good review of the software metrics field is given in Fenton and Pfleeger[12].

The motivation behind this paper is to try to use statistical methods to estimate the number of faults in a piece of code and the time to next failure given both interfailure time and software metrics data. Up to now, there has been little work in combining software metrics and interfailure time models. One paper of interest, however, is Jeske et al[13] where a similar problem is analyzed using an empirically Bayesian approach combining both maximum likelihood and Bayesian techniques. One problem with such an approach is that it may underestimate the overall uncertainty present in the inference. Here, we will use a fully Bayesian method to combine some fault based, interfailure time models with a regression model relating faults to software metrics.

In Section 2, we introduce three related, fault based interfailure time models and in Section 3, given prior distributions and interfailure time data, we show how to carry out Bayesian inference for the model parameters. We note that one problem with Bayesian inference is that this can be sensitive to the prior parameter estimates. Thus, in Section 4, we show how software metrics data can be incorporated via a Poisson regression model to improve the prior mean estimate of the number of faults used for our interfailure time models. Our approach is illustrated with an example in Section 5 and in Section 6 we draw conclusions and consider some possible extensions.

2. Fault Based Software Reliability Models

Here, we consider models for a sequence of interfailure times $T_1, T_2, \ldots$ where it is assumed that between each failure there is an attempt to correct the bug that has caused the problem.

One of the first software reliability models to be developed is that of Jelinski and Moranda[4]. They assume that the interfailure times T_i are independent random variables with exponential densities

$$f_{JM}(t_i|N,\phi) = (N-i+1)\phi \exp\left(-(N-i+1)\phi t_i\right). \tag{1}$$

The Jelinski and Moranda (JM) model may be interpreted as assuming that the program initially contains N faults each of size ϕ and that a fault is removed whenever a failure occurs.

There have been a number of criticisms of the assumptions underlying the *JM* model, see e.g. Littlewood[1], Singpurwalla and Wilson[2] and a number of other software reliability models have been developed to counter these problems.

Firstly, note that under the *JM* model, the failure rate function of the i'th interfailure time is constant; $r_{JM}(t_i|N,\phi) = (N-i+1)\phi$. Schick and Wolverton[5] supposed that this failure rate should be time dependent, so that under their model (SW) the failure rate is $r_{SW}(t_i|N,\theta) = (N-i+1)\theta t_i$ where N represents the initial number of faults as earlier and θ is a parameter which shows how quickly the failure rate increases with time. This implies that the i'th interfailure time has a Rayleigh distribution of form

$$f_{SW}(t_i|N,\theta) = (N-i+1)\theta t_i \exp\left(-\frac{1}{2}(N-i+1)\theta t_i^2\right). \tag{2}$$

A second criticism of the *JM* model is that it assumes that all faults are the same size. Littlewood[6] modified this assumption using Bayesian ideas and his model leads to a Pareto interfailure time density of form

$$f_L(t_i|N,A,B,\tau_i) = (N-i+1)A\frac{(B+\tau_i)^{(N-i+1)A}}{(B+\tau_i+t_i)^{(N-i+1)A+1}} \tag{3}$$

where N is as earlier, A/B is a prior estimate of the initial average fault size and $\tau_i = \sum_{j=1}^{i-1} t_j$ is the sum of the $i-1$ previously observed interfailure times. The failure rate function under the Littlewood (L) model can be shown to be

$$r_L(t_i|N,A,B,\tau_i) = \frac{(N-i+1)A}{B+\tau_i+t_i}. \tag{4}$$

All three models contain the common parameter N which can be interpreted as the initial number of faults in the program. The other parameters have different interpretations for each model.

Assuming that we observe some interfailure time data for a given program, we will typically wish to carry out inference concerning the number of faults and future failure times. Classical, maximum likelihood techniques could be used but problems with maximum likelihood estimation for the *JM* model have been pointed out by e.g. Forman and Singpurwalla[14] and Meinhold and Singpurwalla[15]. Thus, in the following section, we will consider a Bayesian approach.

3. Bayesian Inference

Bayesian inference for the *JM* and other models has been considered in a number of papers; see for example Meinhold and Singpurwalla[15], Littlewood[1] and Wiper et al[16]. To use such methods, we first need to specify prior distributions for the unknown model parameters.

As noted in the previous section, the *JM*, SW and L models all contain the common parameter and a possible prior distribution for N (under all three models) is Poisson, say $N \sim P(\lambda)$ so that

$$P(N = i) = \frac{\lambda^i e^{-\lambda}}{i!}. \tag{5}$$

Here, λ is the prior mean estimate of the number of faults in the program.

The other unknown parameters are model specific. Possible choices of prior distribution for the parameters ϕ, θ and A are gamma distributions say $\phi \sim G(a_{JM}, b_{JM})$, $\theta \sim G(a_{SW}, b_{SW})$ and $A \sim G(a_L, b_L)$ where $X \sim G(a, b)$ if

$$f(x) = \frac{b^a}{\Gamma(a)} x^{a-1} e^{-bx} \tag{6}$$

and, finally we might specify an exponential distribution for the parameter B of the L model: $f_L(B) = \gamma e^{-\gamma B}$.

Note that the same prior distribution structure for *JM* was used by e.g. Meinhold and Singpurwalla[15] and Wiper et al[16] where a similar prior distribution for the parameters of the L model was used. See also Kuo and Yang[17].

Assume now that we observe the first n interfailure times, say $T_1 = t_1, \ldots, T_n = t_n$. Then, for any of the three models say $\mathcal{M} = JM$, SW or L, we can theoretically evaluate the posterior parameter distributions using Bayes theorem:

$$f(N, \vartheta | \mathcal{M}, \mathbf{t}) \propto f(N, \vartheta | \mathcal{M}) f(\mathbf{t} | \mathcal{M}, N, \vartheta) \tag{7}$$

where ϑ represents the model dependent parameters other than N, $\mathbf{t} = (t_1, \ldots, t_n)$ and $f(N, \vartheta | \mathcal{M})$ is the joint prior parameter distribution, $f(\mathbf{t} | \mathcal{M}, N, \vartheta)$ is the likelihood function and $f(N, \vartheta | \mathcal{M}, \mathbf{t})$ is the posterior distribution.

Thus, for example, we could estimate the number of faults remaining in the software by the posterior mean $E[N | \mathcal{M}, \mathbf{t}] - n$.

Unfortunately, for each of the three models, simple closed form expressions for the integration constants of the posterior parameter distributions are not available. Thus, we need to use numerical integration techniques or simulation methods to generate samples from these posterior distributions.

Here we will consider the use of Gibbs sampling (see e.g. Casella and George[18]) as a method to approximate samples from the posterior distributions. This is an example of a Markov chain Monte Carlo method. The idea is to construct a Markov chain with equilibrium distribution equal to the posterior parameter distribution. Then, starting from arbitrary initial values, if the chain is sampled over a large number of iterations, the generated sample will simulate a sample from the true posterior distribution. See, for example, Gilks et al[19] for an introduction. For an application of Gibbs sampling in software reliability modelling see e.g. Kuo and Yang[17].

Thankfully, there are now programmes that can be used to carry out Gibbs sampling in a large number of problems. One such package is *Winbugs* (Spiegelhalter et al[20]). Figure 1, a *Doodle* in Winbugs, illustrates the structure of the *JM* model.

`Figure 1 about here`

Note that square boxes represent constants that need to be fixed a priori and oval boxes represent variables. Arrows connecting boxes indicate a direct dependence (the double arrow indicates a logical relationship). For example, the parameter of the exponential distribution of T_i, called `scale[i]` in the Doodle, directly depends on the values of N, ϕ and i. This graph can be used to generate the following executable code in Winbugs.

```
model; {
   for( i in 1 : n ) {
      scale[i] <- (N - i + 1) * phi
   }
   phi ~dgamma(a_JM,b_JM)
   N ~dpois(lambda)
   for( i in 1 : n ) {
      T[i] ~dexp(scale[i])
   }
}
```

By inputting the fixed prior parameters, (a_{JM}, b_{JM} and λ), the sample data and initial values for the Gibbs sampling algorithm, it is possible to generate samples from the posterior distribution using Winbugs. Note that similar programs can also be set up in Winbugs in order to sample the posterior parameter distributions supposing the SW and L models.

Given a sample from the posterior distribution, it is also easy to construct posterior parameter estimates or estimates of the predictive distribution of the time to next failure. Thus, supposing that $N^{(1)}, \phi^{(1)}, \ldots, N^{(R)}, \phi^{(R)}$ is a sample of size R from the posterior parameter distribution given the JM model, then we can estimate, for example, the number of faults remaining in the software using

$$E[N - n|\mathcal{M}, \mathbf{t}] \approx \frac{1}{R}\sum_{j=1}^{R} N^{(j)} - n \tag{8}$$

or the expected reliability at time t using

$$\begin{aligned} P(T_{n+1} &> t|\mathbf{t}) \approx \frac{1}{R}\sum_{j=1}^{R} P(T_{n+1} > t|N^{(j)}, \phi^{(j)}) \\ &\approx \frac{1}{R}\sum_{j=1}^{R} \exp\left(-(N^{(j)} - n)\phi^{(j)} t\right). \end{aligned} \tag{9}$$

One final problem of interest is that of model comparison. Assume that we wish to compare the fit of the JM and L models to the failure time data. Then the usual Bayesian method is to calculate a Bayes factor

$$B_L^{JM} = \frac{f(\mathbf{t}|JM)}{f(\mathbf{t}|L)}. \tag{10}$$

The statistic $2\log B_L^{JM}$ can be interpreted as a Bayesian version of the usual log-likelihood ratio statistic, see Kass and Raftery[21]. Note that positive (negative) values of $2\log B_L^{JM}$ would suggest that the JM (L) model is preferable. Kass and Raftery[21] provide a table of values indicating when one model may be thought significantly better than another. Given Gibbs sampled output from the posterior parameter distributions under the two models, Chib[22] provides a method of calculating the Bayes factor.

One problem with the Bayesian inference considered in this section is sensitivity to the choice of prior distribution, see e.g. Wiper et al[16], Wilson and Wiper[23]. In particular, it is unclear how to elect the prior mean λ for the distribution of N in the case where we have little prior knowledge available. In this case, it would be natural to use an uninformative, improper prior distribution for N (and the other model parameters) but then it can be shown that there are problems with the propriety of the posterior distribution, see e.g. Wilson and Wiper[23]. In the following section, we illustrate how software metrics information might be used to estimate the value of λ.

4. Incorporation of Software Metrics Information

We now suppose that we have also recorded the values of k software metrics $\mathbf{X} = (X_1, \ldots, X_k)$. Then, we can consider a Poisson regression model to relate faults to metrics:

$$\begin{aligned} N &\sim P(\lambda) \qquad \text{as earlier} \\ \log\lambda &= \beta_0 + X_1\beta_1 + \ldots + X_k\beta_k \end{aligned} \tag{11}$$

It is clear that such a model will not be useful unless either substantial prior information concerning the regression parameters β is available or unless we have metrics and fault data from various other programs which can be used to estimate the regression parameters. Here, we will assume that this is the case so that we have also recorded fault numbers say $M_1, \ldots, M_J$ and metrics data $\mathbf{Z}_1, \ldots, \mathbf{Z}_k$ for a set of J further programs, where $\mathbf{Z}_i = (Z_{i1}, \ldots, Z_{ik})$ represents the vector of metrics for the i'th program.

Given metrics and fault data, various authors have used classical Poisson regression models to estimate the unknown regression coefficients $\boldsymbol{\beta}$, see e.g. Wiper et al.[10], Compton and Withrow[9] or Evanco[11]. Here we consider a fully Bayesian approach which allows us to combine directly the metrics and fault data with failure time data.

We will assume little prior knowledge about the regression parameters $\boldsymbol{\beta}$ and thus consider a relatively diffuse prior distribution, for example a normal distribution with large variance. Given this prior and data structure, including one of the models for interfailure times discussed earlier, we are able to set up a Gibbs sampling algorithm in Winbugs as previously.

Figure 2 shows the representation of the complete system, assuming the *JM* interfailure time model in Winbugs (where we have assumed that $k = 2$ for simplicity). As earlier, executable code can be generated from the doodle which, given the data and initial values for the unknown model parameters $\boldsymbol{\beta}, N, \phi$ can be used to generate a Gibbs sample from the posterior distribution $f(N, \phi, \boldsymbol{\beta}|\text{data})$.

`Figure 2 about here`

As earlier, similar structures can be set up for the *SW* and *L* models.

5. Example

Ten metrics (lines of code, McCabe, Basili Hutchens etc.) and numbers of amendments were collected for 36 unstructured programs. A simple correlations analysis showed that the metrics were very highly correlated and thus, in order to reduce problems of colinearity, a principal components analysis was undertaken and it was observed that the first five principal components, or domain metrics, explained over 99% of the model variation. Thus, these domain metrics were substituted for the original data. Note that the first domain metric could be regarded as a measure of the program size, giving high weights to those metrics which would be expected to increase when the program is larger (e.g. lines of code) and the second domain metrics seemed to be a measure of complexity giving high weights to metrics such as McCabe. The remaining domain metrics were not so easily interpretable. These results mirror those of Wiper et al[10], where a similar interpretation of the domain metrics was found, although the data analysed were somewhat different. Note finally that the use of domain metrics has been considered in a number of other articles, e.g. Khoshgoftaar et al[24] or Lanubile and Visaggio[25].

Here, we use amendments to represent faults and assume the Poisson regression model of Section 4 for the mean number of amendments where the explanatory variable contains the first five principal components and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_5)$. Independent, relatively uninformative, normal distributions each with mean 0 and precision 10^{-6} were chosen as prior distributions for each β_i, $i = 0, 5$.

The remaining model dependent parameters ϕ, θ, A and B were all given relatively uninformative prior distributions with large variances.

5 interfailure times were also generated (assuming the *JM* model with $\phi = 0.1$) from a program containing $N = 9$ amendments / faults.

Given these data, the posterior mean for $\boldsymbol{\beta}$ was estimated to be virtually equal under all three models, and very close to the classical regression estimates. This is as to be expected as the prior distribution used for $\boldsymbol{\beta}$ in each case was relatively uninformative, and the interfailure time data are only indirectly related to the fault and metrics data. Note also that the posterior mean values of the coefficients β_1 and β_2 were both positive, indicating that, for the observed data, fault numbers are generally higher in larger, more complex programs.

There were slight differences in the posterior estimates from each model of the number of faults remaining, $N - 5$. The posterior mean was 4.4 (standard deviation 3.7) supposing the *JM* model, 4.9 (4.3) supposing the *SW* model and 6.2 (4.2) given the L model. The posterior distributions of the number of faults remaining, $N-5$ estimated given all three models are given in Figure 3. It can be seen that the distributions are quite similar in each case. Thus, we can conclude that all three models have done reasonably well in predicting the true number of faults remaining, $N-5=4$.

`Figure 3 about here`

Note also that supposing the JM model, the posterior mean estimate of the parameter ϕ was approximately equal to the true value of 0.1. Thus we can see that even with very few failure data, both model parameters have been well estimated for this model.

Figure 4 ilustrates the predictive reliability functions under all three models. Here we can see some differences between them. It may seem surprising at first that assuming the L model, the software is predicted to be more reliable than under the other two models, given that the mean estimate for the number of faults left in the software is the highest for this model. However, we should recall that under the L model, it is supposed that the largest faults are removed first and that the remaining faults have much smaller rates, whereas under the SW and JM models, all faults have essentially the same importance.

`Figure 4 about here`

Finally, in order to compare the three models, Bayes factors were calculated. These suggested that the model most supported by the data was JM and the least supported was L although the difference between the three models was not great enough to be important. This is as we might expect given that only a very few interfailure times were observed.

6. Conclusions and Extensions

In this paper, we have shown how to combine software metrics data with interfailure time data to improve the predictions of fault numbers and reliability of a program using a Bayesian approach. It has been illustrated that the inference can be carried out using the statistical package Winbugs so that it is not necessary to carry out all of the complex integrations required for Bayesian inference by hand.

A number of extensions are possible. Firstly, in this paper we have considered the case were we have metrics and fault data available for a number of programs and metrics and failure time data available for just one other program. It is straightforward to extend the basic model to the case were interfailure time data are available for more than one program.

Secondly, note that we could easily apply our approach to the interfailure time of Goel and Okumoto[26]. The only problem in this case comes when we wish to estimate Bayes factors comparing the fit of this model with others. This is hard to do directly using the output produced from the package Winbugs but could be carried out using specially written Gibbs sampling software.

Also, in this article we have assumed that the prior distribution for the number of faults in a program takes the form of a Poisson distribution. Under the Poisson model, the prior mean is equal to the variance and an alternative which allows the variance to be greater than the mean would be to use a negative binomial prior distribution. The metrics information could then be incorporated using a negative binomial regression model. It would also be straightforward to set up this system using Winbugs. Work on this extension is underway.

Within the regression model of faults on metrics, we chose here to use the so called domain metrics or principal components rather than the original metrics. It is not clear that this is always a good idea and we should also consider other methods of reducing the dimensionality of the problem, for example by selecting a subset of the metrics which best explain the data. Also, we selected a linear regression model for $\log \lambda$ given the metric values. There have been a number of articles that have proposed various polynomial and other non - linear relationships between fault numbers and, in particular, the lines of code metric (e.g. Gaffney[27]) and it would be interesting to use regression models incorporating some of these.

Finally it would be interesting to consider the use of other information sources with interfailure time models. One group of strategies for software testing, called random and partition testing, uses random and stratified sampling techniques in order to estimate, for example, the probability that a piece of software is failure free or the failure rate of the software. See for example Hierons and Wiper[28]. Given the results from a random or partition test, it would be possible to use these to attempt to improve the estimation of fault numbers in a software program in a similar way to that considered in this paper. Research on how to do this is currently underway.

Acknowledgments

The work of Mike Wiper was partially supported by the project "Análisis Estadístico de Grandes Bancos de Datos Económicos y Empresariales con Estructura Compleja" funded by the Spanish Ministry of Technology.

References

1. B. Littlewood, *Forecasting Software Reliability. Lecture Notes in Computer Science, No. 341.* (Springer Verlag, Berlin, 1989).
2. N.D. Singpurwalla and S.P. Wilson, *Int. Statist. Rev.*, **62**, 289-317 (1994).
3. N.D. Singpurwalla and S.P. Wilson, *Statistical Methods in Software Engineering: Reliability and Risk.* (Springer Verlag, New York, 1999).
4. Z. Jelinski and P. Moranda, in *Statistical Computer Performance Evaluation*, ed. W. Freiburger, (Academic Press, New York, 1972), p. 465-484.
5. G.J. Schick and R.W. Wolverton, in *Proceedings in Operations Research*, (Physica Verlag, Vienna, 1978), p. 395-422.
6. B. Littlewood, *IEEE Trans. Reliab.*, **R-30**, p. 313-320 (1981).
7. T.J. McCabe, *IEEE Transactions on Software Engineering*, **SE-2**, 308-320 (1976).
8. F. Akiyama, *Information Processing Journal*, p. 353-379 (1971).
9. J. Compton and C. Withrow, *J. Systems and Software*, **12**, p. 199-207 (1990).

10. M.P. Wiper, L. Brunenberg and M. Göbbels, in *Proc. Eurometrics '92: European Conference on Quantitative Evaluation of Software & Systems – Practical and Theoretical Aspects*, (EC2, Paris, 1992), p. 91-99.
11. W. Evanco, *J. Systems and Software*, **38**, p. 27-35 (1997).
12. N.E. Fenton and S.L. Pfleeger *Software Metrics. A Rigorous and Practical Approach, 2nd ed.* (PWS Publishing, Boston, 1997).
13. D.R. Jeske, M.A. Qureshi and E. Muldoon, *Int. J. Reliab., Qual. and Safety Eng.*, **7**, p. 153-168. (2000).
14. E.H. Forman and N.D. Sinpurwalla, *J. Amer. Statist. Assoc.*, **72**, p. 750-757 (1977)
15. R.J. Meinhold and N.D. Singpurwalla, *The Statistician*, **32**, p. 168-173 (1983).
16. M.P. Wiper, D. Ríos Insua and R.M. Hierons, *Revista de la real Académia de Ciencias Exactas, Físicas y Naturales (Spain)*, **92**, p. 323-328 (1998).
17. L. Kuo and T. Yang, *J. Amer. Statist. Assoc.*, **91**, p. 763-773 (1996).
18. G. Casella and E.I. George, *The American Statistician*, **46**, p. 167-174 (1992).
19. W. Gilks, S. Richardson and D. Spiegelhalter, in *Markov Chain Monte Carlo in Practice*, eds. W. Gilks, S. Richardson and D. Spiegelhalter. (Chapman and Hall, London, 1996).
20. D.J. Speigelhalter, A. Thomas and N.G. Best, "Winbugs Version 1.2, User Manual", MRC Biostatistics Unit, Cambridge University, 1999.
21. R. Kass and A. Raftery, *J. Amer. Statist. Assoc.*, **90**, p. 773-795 (1995).
22. S. Chib, *J. Amer. Statist. Assoc.*, **90**, p. 1313-1321 (1995)
23. S.P. Wilson and M.P. Wiper, in *Robust Bayesian Analysis*, eds. D. Ríos Insua and F. Ruggeri, (Springer Verlag, New York, 2000), p. 385-400.
24. T. Khoshgoftaar, E. Allen, K. Kalaichelvan and N. Goel, *IEEE Software*, p. 65-71 (1996)
25. F. Lanubile and G. Visaggio, *J. Systems and Software*, **38**, p. 225-234 (1997).
26. A.L. Goel and K. Okumoto, *IEEE Trans. Rel.*, **R-28**, p. 206-211 (1979).
27. J.E. Gaffney, *IEEE Trans. Software Eng.*, **SE-10** (1984).
28. R.M. Hierons and M.P. Wiper, *Software Testing, Verification and Reliability*, **7**, p. 153-164 (1997).

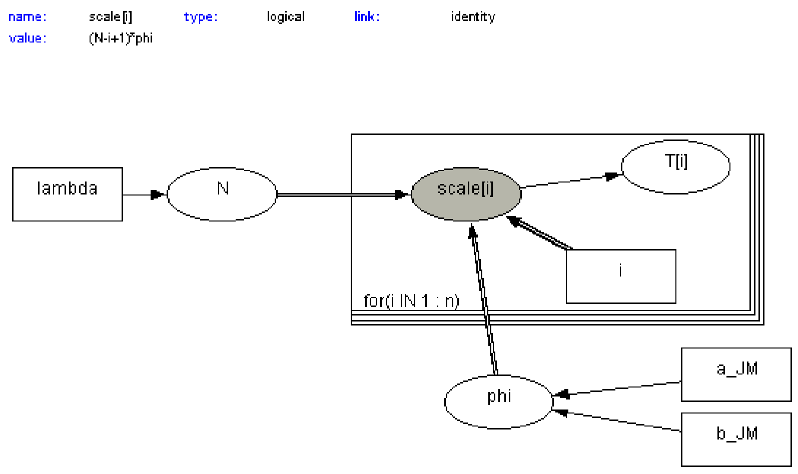
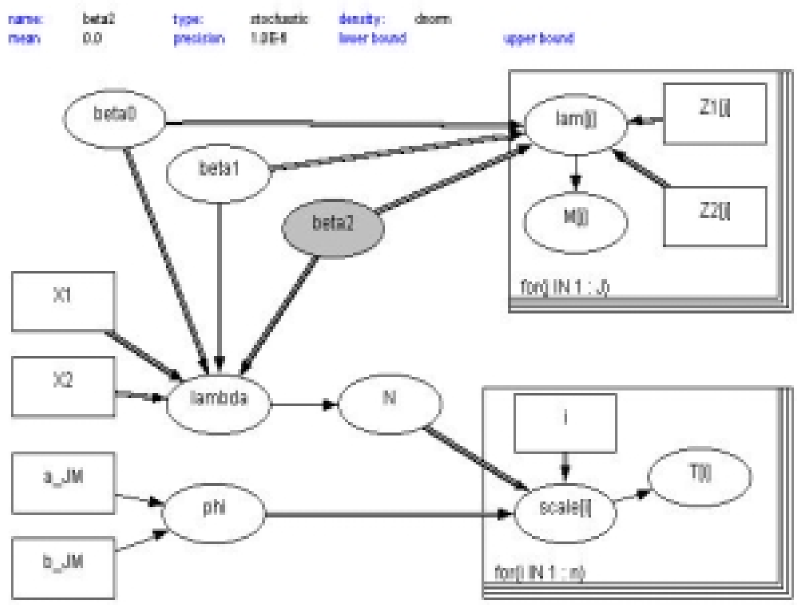

name: scale[i] type: logical link: identity
value: (N-i+1)*phi

Fig. 1. Doodle representing the *JM* model structure



```
name:   beta2   type:       stochastic   density:      dnorm
mean:   0.0     precision:  1.0E-6       lower bound               upper bound
```

Fig. 2. Doodle showing the full model

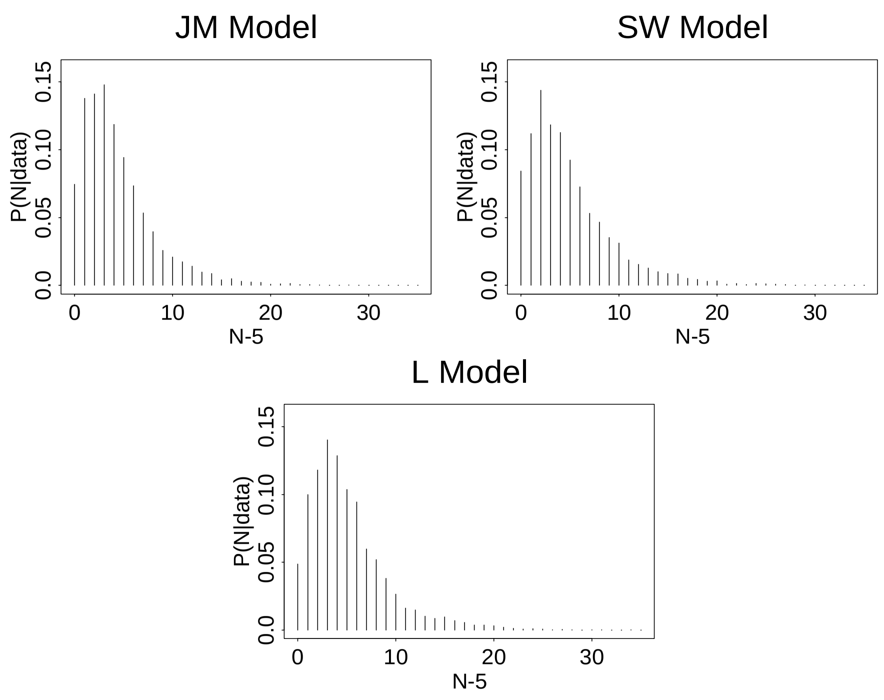


Fig. 3. Posterior distributions of the numbers of faults remaining assuming the *JM*, *SW* and *L* models.

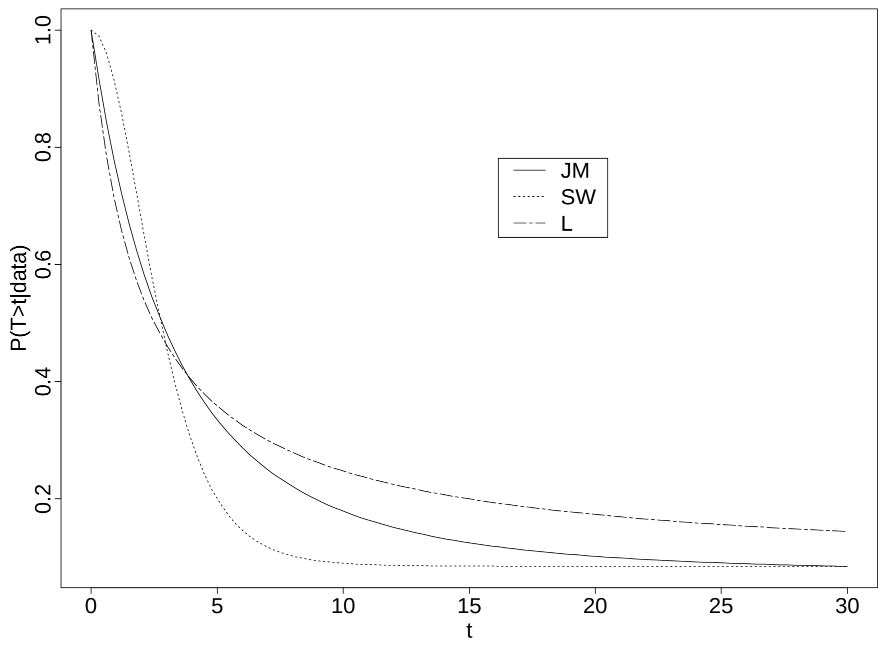


Fig. 4. Predictive reliability functions for the *JM*, *SW* and *L* models.