Working Paper 01-20
Statistics and Econometrics Series 14
March 2001

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 A SOFTWARE RELIABILITY MODEL USING METRICS INFORMATION.

M.P. Wiper and M.T. Rodríguez Bernal*

Abstract

In this paper, we are concerned with predicting the number of faults N and the time to next failure of a piece of software. Information in the form of software metrics data is used to estimate the prior distribution of N via a Poisson regression model. Given failure time data, and a well known model for software failures, we show how to sample the posterior distribution using Gibbs sampling, as implemented in the package "WinBugs". The approach is illustrated with a practical example.

Keywords: Software metrics; principal components; software reliability; Jelinski Moranda model; Bayesian inference; Gibbs sampling; WinBugs.

*Wiper, Departamento de Estadística y Econometría, Universidad Carlos III de Madrid, C/ Madrid 126, 28903 Getafe -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. The work of Mike Wiper has been partially supported by the project Análisis Estadístico de Grandes Bancos de Datos Económicos y Empresariales con Estructura Compleja funded by the Spanish Ministry of Science and Technology.

Bayesian inference for a software reliability model using metrics information

M P Wiper M T Rodriguez Bernal*

March 8, 2001

Abstract

In this paper, we are concerned with predicting the number of faults N and the time to next failure of a piece of software. Information in the form of software metrics data is used to estimate the prior distribution of N via a Poisson regression model. Given failure time data, and a well known model for software failures, we show how to sample the posterior distribution using Gibbs sampling, as implemented in the package "WinBugs". The approach is illustrated with a practical example.

Keywords: software metrics, principal components, software reliability, Jelinski Moranda model, Bayesian inference, Gibbs sampling, WinBugs.

1 Introduction

Software reliability can be defined as "the probability of failure-free operation of a computer code for a specified mission time in a specified input environment", (Singpurwalla and Wilson 1999).

The majority of software reliability models are concerned with the prediction of interfailure times, say $T_1, T_2, \ldots$, see for example Littlewood (1989) or Singpurwalla and Wilson (1994). It would seem plausible that failure times are related to the number of faults (or bugs) in the software, say N, and various models have been developed which are based on a bug counting approach. Specific examples are the models of Jelinski and Moranda (1972),

*Departamento de Estadística y Econometría, Universidad Carlos III de Madrid.

Goel and Okumoto (1978), Schick and Wolverton (1978) and Musa and Okumoto (1987). See Singpurwalla and Wilson (1999) for a general review and comments about alternative approaches to failure time prediction.

A second method for reliability assessment, which we do not consider in this article, uses random or partition testing methods to estimate the failure rate or probability that an input to the program from the operational profile will produce an erroneous output. See for example Musa (1993) and Hierons and Wiper (1997).

A third approach to assessing software reliability, particularly in the software development process, concentrates on the measurement of characteristics of a piece of software, called software metrics. The most well known software metrics are probably the simple lines of code (LOC) and McCabe's (1976) cyclomatic complexity measure[1]. A great many other metrics have also been defined. See, for example, Fenton and Pfleeger (1997) for a review.

Various models have been used in order to predict the number of bugs (or other measures of quality such as cost, amendements etc.) in a piece of software using software metrics data. For example, Mayer and Sykes (1991), Wiper et al (1992) etc. used (classical) logistic regression approaches, normal regression models have also been used (Compton and Withrow 1990) and Khoshgoftaar et al (1995) introduced a neural net approach.

In this paper we are concerned with prediction of the number of bugs in a piece of software and the time to next failure given both failure time information and metrics data. Our objective is to introduce Bayesian inference techniques in order to combine the two sources of information.

In Section 2, we introduce on of the most well known software reliability models and comment on Bayesian inference for this model. In Section 3, we introduce a Bayesian logistic regression model for the number number of faults in a piece of software and combine this model with that of the previous section and illustrate how to carry out Bayesian inference using Gibbs sampling. In Section 4, we illustrate our approach with an example using real software metrics data and in Section 5 we consider some possible extensions.

2 A simple model for software falures

The Jelinski Moranda (1972) model for software reliability is one of the simplest. It supposes that the distribution of the j'th interfailure time is given

[1]This measure is based on the representation of the code as a directed graph and is equal to be the number of edges minus the number of nodes plus 2.

by

$$T_j|N,\phi \sim \mathcal{E}\left((N-j+1)\phi\right),$$

an exponential distribution with mean $1/(N-j+1)\phi$.

The underlying (possibly unrealistic) assumptions of this model are:

1. The program contains an initial number of faults N.

2. Each bug contributes the same amount to the failure rate.

3. After each observed failure, a bug is detected and corrected.

Bayesian inference for the Jelinski Moranda model is considered by, for example, Meinhold and Singpurwalla (1983) who introduce the following natural independent prior distributions for N and ϕ:

$$\begin{array}{rcll} N & \sim & \mathcal{P}(\lambda) & \text{a Poisson distribution} \\ \phi & \sim & \mathcal{G}(\alpha,\nu) & \text{a gamma density} \end{array}$$

Given the observation of n failures, then the likelihood function is, for $N \geq n$,

$$l(N,\phi|\text{data}) \propto \frac{N!}{(N-n)!}\phi^n \exp\left(-\sum_{j=1}^{n}(N-j+1)t_j\phi\right)$$

so that, for example, the full conditional posterior distributions are easily evaluated:

$$\begin{array}{rcll} N-n|\text{data},\phi & \sim & \mathcal{P}\left(\lambda\exp(-n\bar{t}\phi)\right) & \text{where } \bar{t}=\frac{1}{n}\sum_{j=1}^{n} t_j \\ \phi|\text{data},N & \sim & \mathcal{G}\left(\alpha+n,\nu+\sum_{j=1}^{n}(N-j+1)t_j\right). & \end{array}$$

Thus, it is easy to set up a Gibbs sampling scheme to sample the posterior distribution of (N,ϕ)[2]

One problem with this model is the possible sensitivity to the election of the prior distribution, see, e.g Wiper et al (1998), Wilson and Wiper (2000). In particular, it is unclear how to elect the prior parameters λ,α,ν. Given little prior knowledge, it would be natural to elect an uninformative, improper, prior distribution for (N,ϕ) but it is easy to show that in this case, the posterior distribution is also improper, see e.g. Wilson and Wiper (2000).

In the following section, we illustrate how software metrics data might be used to provide information about the prior distribution for N.

[2]The marginal posteriors of N and ϕ can also be derived up to truncation of a one dimensional summation.

3 Software Metrics

In this section, we consider first a different problem, that of predicting the number of faults in a software program given metrics data.

Suppose that, for a collection of programs m, we calculate k software metrics $\mathbf{X}_i = (X_{i1}, \ldots, X_{ik})$, where $i = 1, \ldots, m$.

Let N_i be the number of faults in program i. Then the following logistic regression is a natural model relating faults to software metrics:

$$
\begin{aligned}
N_i|\lambda_i &\sim \mathcal{P}(\lambda_i) \qquad \text{as in the previous section} \\
\log \lambda_i &= \beta_0 + X_{i1}\beta_1 + \ldots + X_{ik}\beta_k
\end{aligned}
$$

Given fault data $N_1 = n_1, \ldots, N_m = n_m$, various authors have used classical logistic regression to estimate the unknown parameters $\boldsymbol{\beta}$, see e.g. Mayer and Sykes (1991), Wiper et al (1992). In this case we consider a Bayesian approach, which, as far as we know, has not been considered previously for metrics data.

Given a prior distribution for $\boldsymbol{\beta}$, for example a normal distribution $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{b}, \mathbf{B})$ then it can be shown that all of the conditional posterior parameter distributions are log concave (see Dellaportas and Smith 1993) and so the adaptive rejection algorithm (Gilks and Wild 1992) can be used to sample from the posterior distribution. This scheme is preprogrammed into the Gibbs sampling package WinBugs (Spiegelhalter et al 1999).

One problem with metrics data is that typically, software metrics are very highly correlated with program size, see e.g. Wiper et al (1992). Thus, rather than use the raw metrics data, to avoid problems of colinearity, it is normal to transform the metrics using, for example, principal components. Typically, we will have little prior information concerning these components and thus it is normal to use a relatively diffuse prior distribution for $\boldsymbol{\beta}$.

Assuming that both sources of information are available, the logistic regression model for metrics data can be combined with the Jelinski Moranda model for failure times. Figure 1 shows a directed graph (a "Doodle" in "Winbugs") which illustrates the dependence structure of the combined model. (In the diagram, we have used Nnew instead of N for the number of bugs we are trying to predict.) Using the graph, distributional assumptions for each node can be specified within Winbugs.

We illustrate the approach with an example using real metrics data and simulated failure time data in the following section.

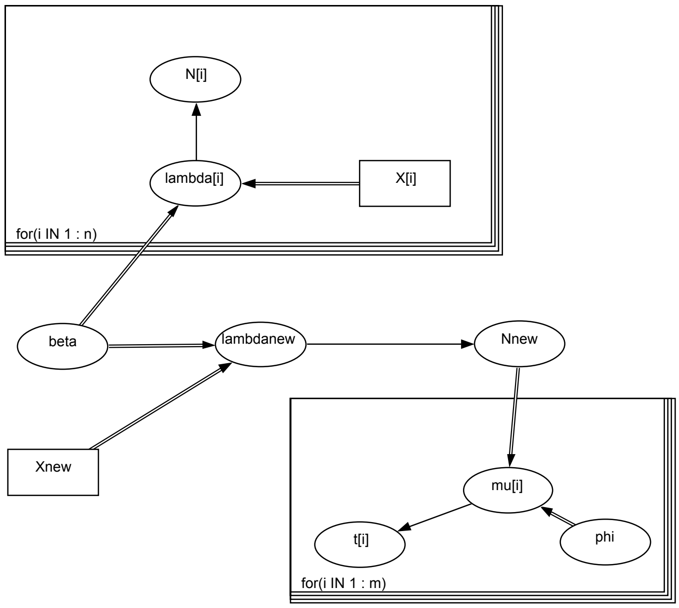

Figure 1: Doodle of Model Structure.

4 Example

Ten metrics (Basili Hutchens, depth of nesting, LOC, McCabe, YAM etc.) and the number of amendments were collected from 36 unstructured software programs. A simple correlations analysis showed that the metrics were very highly correlated and thus, a principal components analysis was undertaken, which showed that virtually all (99%) of the variance in the metrics data was explained by the first five principal components.

Here we use amendments to represent faults and assume the logistic regression model of section 3 for the mean number of amendments where the vector $\mathbf{X}$ contains the first five principal components and $\boldsymbol{\beta} = (\beta_0, ..., \beta_5)$. Independent, relatively uninformative, normal distributions with mean 0 and precision 10^{-5} were chosen for each β_i.

5 interfailure times were also generated using the Jelinski Moranda model (with $\phi = 0.1$) from a program containing 9 amendments / bugs.

A relatively uninformative gamma prior distribution $\mathcal{G}(.001, .001)$ was used for ϕ. A Gibbs run of 5000 iterations to "burn in" the chain and 25000 iterations in equilibrium was used. (Using the usual checks, the chain could be seen to have coverged before 5000 iterations).

In this case, the posterior mean estimate for $\boldsymbol{\beta}$ coincides almost exactly with the classical logistic regression estimate based on the 36 metric data, which we would expect given the use of noninformative prior distributions. In Figures 2 and 3, we illustrate the estimated posterior densities for N (Nnew) and ϕ.

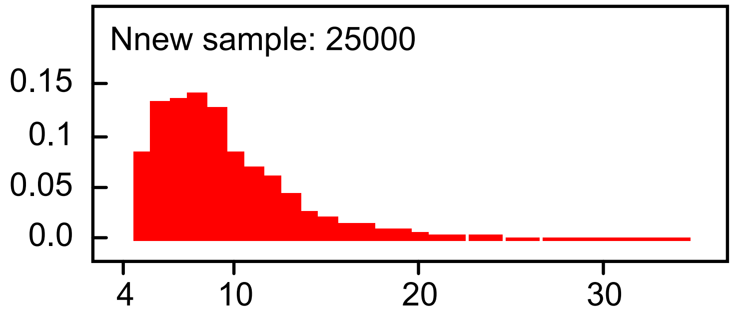

Figure 2: Posterior density of N.

Values of N between 6 and 9 have posterior probabilities greater than 0.1. The posterior mean is $E[N|\text{data}] \simeq 9.4$ (standard deviation 3.7) and a 95% probability interval for N is [5,19]. Thus, the true value of N has been well estimated given the metrics and failure data.

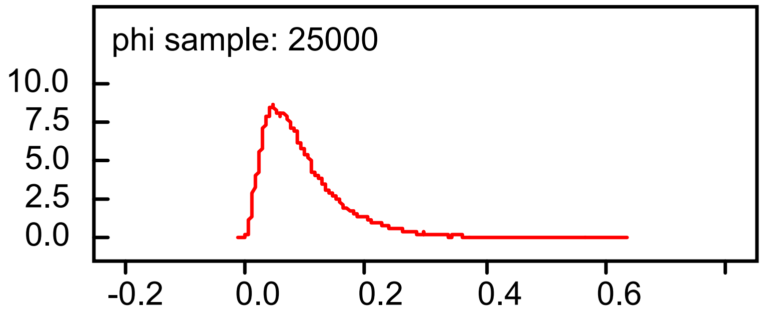

Figure 3: Posterior density of ϕ.

The posterior mean of ϕ is $E[\phi|\text{data}] \simeq 0.102$ (standard deviation 0.072) which again coincides very well with the true generating value.

Finally, Figure 4 illustrates the reliability function for the time to next (sixth) failure of the program: $P(T_6 > t|\text{datos})$. It can be seen that there is a positive probability that the program will not fail again, which is equal to the posterior probability that the program contained only the five discovered faults.

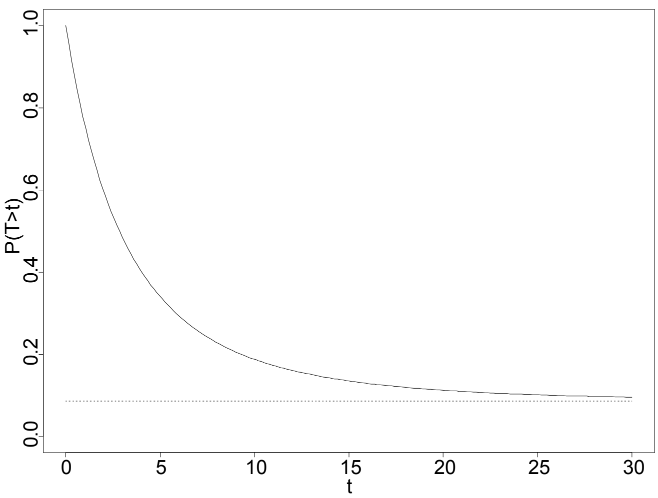

Figure 4: Reliability function for the time to next failure.

5 Conclusions and Extensions

In this paper we have shown how Bayesian inference concerning the number of bugs and the time to next failure of a piece of software can be carried out given both failure time and software metrics data. Here we have supposed the Jelinski Moranda model for failure times, but the same basic techniques may be applied to other models. Work is currently in progress on Bayesian inference for these models.

When there are a selection of possible models, the problem of model choice becomes important. Theoretically, different models may be assessed from a Bayesian perspective via the use of Bayes factors, see e.g. Kass and Raftery (1995). A practical problem is that Bayes factors are difficult to calculate solely from the WinBugs output and thus, the programming task is somewhat more complicated.

6 Acknowledgements

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

Bibliography

Compton, J. and Withrow, C. (1990). Prediction and control of Ada software defects. *Journal of Systems and Software*, **12**, 199-207.

Dellaportas, P. and Smith, A.F.M. (1993). Bayesian inference for generalized linear and proportional hazards models via Gibbs sampling. *Applied Statistics*, **42**, 443-459.

Fenton, N. E. and Pfleeger, S. L. (1997). *Software Metrics. A Rigorous and Practical Approach*, 2nd. edition. Boston: PWS Publishing.

Gilks, W.R. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics*, **41**, 337-348.

Goel, A. L. and Okumoto, K. (1979). Time dependent error detection rate model for software reliability and other performance measures. *IEEE Trans. Rel.*, **R-28**, 206-211.

Hierons, R.M. and Wiper, M.P. (1997). Estimation of Failure Rate using Random and Partition Testing. *Journal of Software Testing, Verification and Reliability*, **7**, 153-164.

Jelinski, Z. and Moranda, P. (1972). Software reliability research. In *Statistical Computer Performance Evaluation*, W. Freiberger, editor, 465–484. New York: Academic Press. *IEEE Trans. Reliability*, **R-34**, 216–218.

Kass, R.E. and Raftery, A.E. (1995). Bayes Factors. *J. Amer. Statist. Assoc.*, **90**, 773-795.

Khoshgoftaar, T.M., Panday, A.S. and Lanning, D.L. (1995). Application of nueral networks for predicting program faults. *Annals of Software Engineering*, **1**, 141-154.

Littlewood, B. (1989). *Forecasting Software Reliability. Lecture Notes in Computer Science, No. 341.* Berlin: Springer-Verlag.

Mayer, A. and Sykes, A. (1991). Statistical Methods for the Analysis of Software Metrics Data. In *Proceedings of the 2nd Annual Oregon Workshop on Software Metrics Data*, Portland.

McCabe, T.A. (1976). A Software Complexity Measure. *IEEE Trans. Software Engineering*, **SE-2**, 308-320.

Meinhold, R.J. and Singpurwalla, N.D. (1983). Bayesian Analysis of a Commonly Used Model for Describing Software Failures. *The Statistician*, **32**, 168-173.

Musa, J.D. (1993). Operational profile in software-reliability engineering. *IEEE Transaction on Software*, **10**, 14-32.

Musa, J. D. and Okumoto, K. (1987). A logarithmic Poisson execution time model for software reliability measurement. *Proceedings of the 7th International Conference on Software Engineering, Orlando*, 230–237.

Schick, G. J. and Wolverton, R. W. (1978). Assessment of Software Reliability. In *Proceedings in Operations Research*, Physica-Verlag, Vienna, 395–422.

Singpurwalla, N. D. and Wilson, S. P. (1994). Software reliability modeling. *Int. Statist. Rev.*, **62**, 3: 289–317.

Singpurwalla, N. D. and Wilson, S. P. (1999). *Statistical Methods in Software Engineering: Reliability and Risk*, New York: Springer-Verlag.

Spiegelhalter, D.J., Thomas, A. and Best, N.G. (1999). *WinBugs Version 1.2 User Manual.* MRC Biostatistics Unit, Cambridge University.

Wilson, S. P. and Wiper, M. P. (2000). Prior Robustness in some Common Types of Software Reliability Model. In *Robust Bayesian Analysis*, (D. Ríos Insua and F. Ruggeri eds.) New York: Springer-Verlag.

Wiper, M. P., Brunenberg, L. and Göbbels, M. (1992). Relation between software metrics. In *Proceedings of Eurometrics '92; European Conference on Quantitative Evaluation of Software & Systems - Practical and Theoretical Aspects*, EC2: Paris, 91–99.

Wiper, M. P., Ríos Insua, D. and Hierons, R. M. (1998). Bayesian inference and optimal release times for two software failure models. *Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales (España)*, **92**, 323–328.