Journal of Modern Applied Statistical Methods

Volume 17 | Issue 2 Article 13

3-28-2019

JMASM 51: Bayesian Reliability Analysis of Binomial Model – Application to Success/Failure Data

M. Tanwir Akhtar
Saudi Electronic University, Jeddah, Saudi Arabia, tanwira08@gmail.com

Athar Ali Khan
Aligarh Muslim University, Aligarh, India, atharkhan1962@gmail.com

Recommended Citation
Akhtar, M. T., & Khan, A. A. (2018). JMASM 51: Bayesian reliability analysis of binomial model – Application to success/failure data. *Journal of Modern Applied Statistical Methods, 17*(2), eP2623. doi: 10.22237/jmasm/1553803862

Journal of Modern Applied Statistical Methods
November 2018, Vol. 17, No. 2, eP2623
doi: 10.22237/jmasm/1553803862

בס"ד

ISSN 1538 − 9472

JMASM 51: Bayesian Reliability Analysis of Binomial Model – Application to Success/Failure Data

M. Tanwir Akhtar
Saudi Electronic University
Jeddah, Saudi Arabia

Athar Ali Khan
Aligarh Muslim University
Aligarh, India

Reliability data are generated in the form of success/failure. An attempt was made to model such type of data using binomial distribution in the Bayesian paradigm. For fitting the Bayesian model both analytic and simulation techniques are used. Laplace approximation was implemented for approximating posterior densities of the model parameters. Parallel simulation tools were implemented with an extensive use of R and JAGS. R and JAGS code are developed and provided. Real data sets are used for the purpose of illustration.

Keywords: Bayesian Analysis, binomial model, Laplace approximation, simulation, posterior, R

Introduction

Reliability data are generated in the form of success/failure. For example, in a missile system, it is recorded whether a launched missile executes the mission successfully. For modeling such type of data, the binomial distribution is used. It is appropriate for a fixed number of tested components, n, where the tests are assumed to be conditionally independent given the success probability θ (Hamada, Wilson, Reese, & Martz, 2008).

The non-Bayesian analysis of success/failure data is not an easy task, whereas it can be implemented in principle when dealing in a Bayesian paradigm, provided simulation tools are used. For the purpose of Bayesian modeling of success/failure reliability data, two important techniques, the asymptotic approximation and simulation methods, are implemented using the LaplacesDemon (Statisticat, LLC, 2018) and JAGS (Plummer, 2003) packages in R (R Core Team, 2018). LaplacesDemon facilitates high-dimensional Bayesian inference, posing as its own

doi: 10.22237/jmasm/1553803862 | Accepted: August 29, 2017; Published: March 28, 2019.
Correspondence: M. Tanwir Akhtar, tanwira08@gmail.com

intellect that is capable of impressive analysis, which is written in the R environment and has a provision for user defined probability model.

The function `LaplaceApproximation` in the LaplacesDemon package approximates the results asymptotically while the function `LaplacesDemon` simulates the results from the posterior by using one of the several Metropolis algorithms of Markov chain Monte Carlo (MCMC). JAGS approximates the posterior parameter using a Metropolis-within-Gibbs (MWG) simulation method. These techniques are used both for intercept as well as regression models. Real data sets are used in subsequent analysis for the purpose of illustrations. Thus, the Bayesian reliability analysis of binomial models has been made with the objectives: (a) define a Bayesian model, the specification of likelihood and prior distribution; (b) write R and JAGS code for approximating the posterior densities using analytic and simulation tools; and (c) finally, illustrate numeric as well as graphic summaries of the posterior densities.

The Binomial Distribution

The binomial distribution is a single-parameter distribution which provides a natural model that arises from a sequence of *n* exchangeable (or independent and identically distributed Bernoulli) trials or draws from a large population where each trial gives rise to one of two possible outcomes, conveniently labelled *success* and *failure*. For the success outcome the value of the random variable is assigned 1, otherwise the variable is assigned 0. Because of the exchangeability, the data can be summarized by the total number of success in *n* trials, which is denoted here by *y*. Converting from a formulation in terms of the exchangeable trials to one using independent and identically distributed (iid) random variables is achieved quite naturally by letting the parameter θ represent the proportion of success in the population or, equivalently, the probability of success in each trial. The binomial probability distribution states that

$$\mathrm{p}\left(y \mid \theta\right) = \mathrm{Binomial}\left(n, \theta\right) = \binom{n}{y} \theta^{y} \left(1-\theta\right)^{n-y}, \quad 0 \le \theta \le 1, \tag{1}$$

where on the left side we suppress the dependence on *n* because it is regarded as a part of the experimental design that is considered fixed; all the probabilities discussed for this problem are assumed to be conditional on *n* (Gelman, Carlin, Stern, & Rubin, 2004). The binomial distribution is not an appropriate model if the

tests are dependent, and it only applies if all the items have the same success probability. For $n = 1$, the binomial is called the *Bernoulli* distribution.

The Prior Distributions

In the Bayesian paradigm, it is needed to specify prior information regarding the value of the parameter of interest or information that is available before analyzing the experimental data by using a probability distribution function. This probability distribution function is called the prior probability distribution, or simply the prior, because it reflects information about parameter prior to observing experimental data. Two prior distributions are discussed according to their uses in subsequent Bayesian reliability analysis.

Weakly Informative Priors

The Weakly Informative Prior (WIP) distribution uses prior information for regularization and stabilization, providing enough prior information to prevent results that contradict knowledge or problems such as an algorithmic failure to explore the state-space. Another goal is for WIPs to use less prior information than is actually available. A WIP should provide some of the benefit of prior information while avoiding some of the risk from using information that doesn't exist (Statisticat, LLC, 2018). A popular WIP for a centered and scaled predictor (Gelman, 2008) may be

$$\theta \sim \mathrm{N}\left(0,10000\right)$$

where θ is normally-distributed with a mean of 0 and a large variance of 10000, which is equivalent to a standard deviation of 100, or precision of 1.0×10^{-4}. In this case, the density for θ is nearly flat. Prior distributions that are not completely flat provide enough information for the numerical approximation algorithm to continue to explore the target density, the posterior distribution.

The Half-Cauchy Weakly Informative Prior Distribution

The probability density function of the half-Cauchy distribution with scale parameter α is given by

$$\mathrm{f}\left(x\right)=\frac{2\alpha}{\pi\left(x^{2}+\alpha^{2}\right)}\quad x>0,\alpha>0\,.$$

The mean and variance of the half-Cauchy distribution do not exist, but its mode is equal to 0. The half-Cauchy distribution with scale $\alpha = 25$ is a recommended, default, weakly informative prior distribution for a scale parameter (Gelman, 2006). At this scale $\alpha = 25$, the density of half-Cauchy is nearly flat but not completely (see Figure 1); prior distributions that are not completely flat provide enough information for the numerical approximation algorithm to continue to explore the target density, the posterior distribution. The inverse-gamma is often used as a prior distribution for the scale parameter; however, this model creates problem for scale parameters near zero. Gelman and Hill (2007) recommend that the uniform or, if more information is necessary, the half-Cauchy is a better choice (Akhtar & Khan, 2014a, b; N. Khan, Akhtar, & Khan, 2016; N. Khan, Akhtar, & Khan, 2017). Thus, the half-Cauchy distribution with scale parameter $\alpha = 25$ is used as a weakly informative prior distribution.

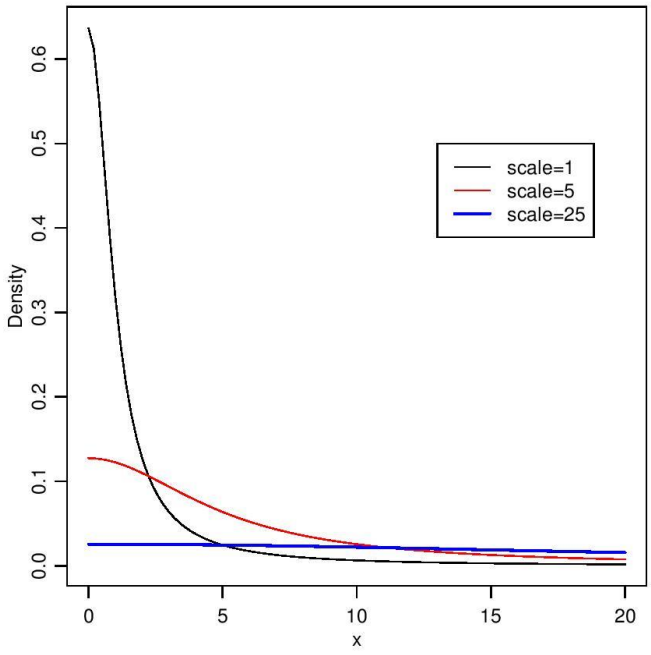

Figure 1. It is evident from the above plot that for scale = 25 the half-Cauchy distribution becomes almost uniform

Tools and Techniques

The technical problem of evaluating quantities required for Bayesian inference typically reduces to the calculation of a ratio of two integrals (Bernado & Smith, 2000). In all cases, the technical key to the implementation of the formal solution given by Bayes' theorem is the ability to perform a number of integrations (Bernado & Smith, 2000). Except in certain rather stylized problems, the required integrations will not be feasible analytically and thus, efficient approximation strategies are required (Statisticat, LLC, 2018). There are many types of numerical approximation algorithms in Bayesian theory. An incomplete list of broad categories of Bayesian numerical approximation may include asymptotic approximation methods like Laplace approximations and Markov chain Monte Carlo simulation methods. The Laplace approximation is implemented using the `LaplaceApproximation` function of LaplaceDemon and Markov chain Monte Carlo algorithms are implemented using the `LaplacesDemon` and `jags` functions of LaplacesDemon and R2jags (Su & Yajima, 2014), respectively.

The Laplace Approximation

The integrals involved in Bayesian analysis are often quite complex. Often an analytic approximation such as *Laplace's method*, based on the normal distribution, can be used to provide an adequate approximation to such integrals. To evaluate the integral

$$\mathrm{E}\left(\mathrm{h}\left(\theta\right)|\, y\right) = \int \mathrm{h}\left(\theta\right)\mathrm{p}\left(\theta\,|\, y\right) d\theta$$

using Laplace's method, express the integrand in the form exp[log(h(θ)p($\theta \mid y$))] and then expand log(h(θ)p($\theta \mid y$)) as a function of θ in a quadratic Taylor series approximation around its mode. The resulting approximation for h(θ)p($\theta \mid y$) is proportional to a (multivariate) normal density in θ, and its integral (Gelman et al., 2004) gives an approximation of E(h(θ) | y):

$$\mathrm{h}\left(\theta_0\right)\mathrm{p}\left(\theta_0\,|\, y\right)\left(2\pi\right)^{d/2}\left|-u''\left(\theta_0\right)\right|^{1/2}$$

where d is the dimension of θ, u(θ) = log(h(θ)p($\theta \mid y$)), and θ_0 is the point at which u(θ) is maximized. If h(θ) is fairly smooth function, this approximation is often reasonable in practice, due to the approximate normality of the posterior distribution, p($\theta \mid y$), for large sample size. Because Laplace's method is based on

normality, it is most effective for unimodal posterior densities or when applied separately to each mode of a multimodal density (Gelman et al., 2004). More details about Laplace approximation can be found in Tierney and Kadane (1986), Tierney, Kass, and Kadane (1989), Mosteller and Wallace (1964), and Tanner (1996). The same Laplace's approximation is implemented in the LaplacesDemon package as the `LaplaceApproximation` function, and this function is used to approximate the posterior densities analytically.

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods were initially introduced into physics in 1953 in a simplified version by Metropolis, Rosenbluth, Rosenbluth, and Teller (1953), with intermediate generalization by Hastings (1970) and the development of the Gibbs sampler by Geman and Geman (1984). Nevertheless, it took about 35 years until MCMC methods were rediscovered (Gelfand, Hills, Racine-Poon, & Smith, 1990; Gelfand & Smith, 1990; Tanner & Wong, 1987) and became one of the main computational tools in modern statistical inference.

MCMC methods permits use of highly complicated models and estimate the corresponding posterior distributions with accuracy. MCMC methods have contributed to the development and propagation of Bayesian theory. MCMC methods are based on the construction of a Markov chain that eventually converges to the target distribution (called *stationary* or *equilibrium*), which in the current case is the posterior distribution, $p(\theta \mid y)$. This is the main way to distinguish MCMC algorithms from direct simulation methods, which provide samples directly from the target – posterior distribution. Moreover, the MCMC produces a dependent sample since it is generated from a Markov chain in contrast to the output of direct methods, which is an independent sample. These MCMC methods incorporate the notion of an iterative procedure; for this reason, they are frequently called iterative methods since in every step they produce values depending on the previous one. These techniques are implemented using the function `LaplacesDemon` of the LaplacesDemon package.

Analysis of Intercept Model

Binomial data are frequently encountered in modern science, especially in the field of reliability application, where the observed response is usually binary indicating whether a component fails or not during an experiment or binomial such as the number of failures over a specified time. For modeling of such type of binary data

binomial distribution is used and its Bayesian analysis can easily be performed using tools like R and JAGS. Thus, the Bayesian analysis of binomial model for binary reliability data is discussed here.

The Model

Consider a set of binomial data y_i that expresses the number of successes over n_i trials (for $i = 1, 2,\ldots, J$),

$$y_i \sim \text{Binomial}\left(n_i, \theta\right),$$

resulting to the likelihood given by

$$\text{p}\left(y \mid \theta\right) = \prod_{i=1}^{J}\left[\binom{n_i}{y_i}\theta^{y_i}\left(1-\theta\right)^{n_i-y_i}\right] = \prod_{i=1}^{J}\binom{n_i}{y_i}\times\theta^{\Sigma y_i}\left(1-\theta\right)^{n-\Sigma y_i} \tag{2}$$

where $n = \sum_{i=1}^{J} n_i$ is the total number of Bernoulli trials in the sample. In order to perform Bayesian inference in the binomial model, specify the prior distribution for θ. A convenient choice of the prior distribution for θ is the beta conjugate prior distribution with parameters α and β, denoted by

$$\theta \sim \text{Beta}\left(\alpha, \beta\right)$$

and probability density function

$$\text{p}\left(\theta\right) = \frac{\Gamma\alpha\Gamma\beta}{\Gamma\left(\alpha+\beta\right)}\theta^{\alpha-1}\left(1-\theta\right)^{\beta-1}, \tag{3}$$

where α is interpreted as the prior number of successful component tests and β as the prior number of failed component tests, that is, $\alpha + \beta$ is like a prior sample size. Then, the resulting posterior distribution for success probability θ is given by

$$\begin{aligned} \mathrm{p}(\theta \mid y) &\propto \mathrm{p}(\theta)\mathrm{p}(y \mid \theta) \\ &\propto \frac{\Gamma\alpha\Gamma\beta}{\Gamma(\alpha+\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \times \prod_{i=1}^{J}\binom{n_i}{y_i} \times \theta^{\Sigma y_i}(1-\theta)^{n-\Sigma y_i} \\ &\propto \theta^{\alpha+\Sigma y_i-1}(1-\theta)^{\beta+n-\Sigma y_i-1} \end{aligned} \tag{4}$$

which is the beta distribution

$$\theta \mid y \propto \mathrm{Beta}\left(\alpha+\sum y_i, n+\beta-\sum y_i\right), \tag{5}$$

with shape parameters ($\alpha + \Sigma y_i$) and ($n + \beta - \Sigma y_i$). The posterior mean and variance are, respectively,

$$\mathrm{E}(\theta \mid y) = \frac{\alpha+\sum y_i}{n+\alpha+\beta}, \tag{6}$$

$$\mathrm{V}(\theta \mid y) = \frac{\left(\alpha+\sum y_i\right)\left(n+\beta-\sum y_i\right)}{(n+\alpha+\beta)^2(n+\alpha+\beta+1)}. \tag{7}$$

The posterior mean can also be expressed as a weighted average of the prior and sample proportion

$$\begin{aligned} \mathrm{E}(\theta \mid y) &= \left(\frac{n}{n+\alpha+\beta}\right)\left(\frac{\sum y_i}{n}\right) + \left(\frac{\alpha+\beta}{n+\alpha+\beta}\right)\left(\frac{\alpha}{\alpha+\beta}\right) \\ &= \omega\left(\frac{\sum y_i}{n}\right) + (1-\omega)\left(\frac{\alpha}{\alpha+\beta}\right) \end{aligned} \tag{8}$$

where $\omega = n / (n + \alpha + \beta)$, $\Sigma y / n$ is the sample proportion, and $\alpha / (\alpha + \beta)$ is the mean of a beta prior with parameters α and β. A beta distribution with equal and low parameter values can be considered as a weakly informative prior (e.g., $\alpha = \beta = 10^{-3}$). Other choices usually adopted are Beta(1/2, 1/2) or Beta(1, 1), which are equivalent to the uniform distribution U(0, 1). The latter can be considered as a weakly informative prior distribution since this prior gives the same probability to any interval of the same range. Nevertheless, this prior will be influential when the sample size is small (Ntzoufras, 2009). This might not necessarily be a

disadvantage since, for small sample sizes, the posterior will also reflect the low available information concerning the parameter of interest θ.

Implementation

To implement the Bayesian analysis of the binomial model, a binary data set is taken from Johnson, Moosman, and Cotter (2005). The data are reported below. Using these data different aspects of Bayesian modeling are discussed. Two different functions, `bayesglm` and `jags` of R and JAGS, respectively, are used to fit the model and to estimate the parameter of interest, the success probability. Numerical as well as graphical summaries of the corresponding results are also reported.

Launch Vehicle Data Set

Represented in Table 1 are the responses of a set of success/failure data. These are the launch outcomes of new aerospace vehicles conducted by new companies during the period 1980-2002. A total of 11 launches occurred in which 3 were successes and 8 were failures. Reliability is the probability of successful launch (Hamada et al., 2008).

Table 1. Outcomes for 11 launches of new vehicles performed by new companies with limited launch-vehicle design experience, 1980-2002

Vehicle	Outcome
Pegasus	Success
Percheron	Failure
AMROC	Failure
Conestoga	Failure
Ariane 1	Success
India SLV-3	Failure
India ASLV	Failure
India PSLV	Failure
Shavit	Success
Taepodong	Failure
Brazil VLS	Failure

Fitting with JAGS

Consider the Bayesian analysis of the launch vehicle data with JAGS using its interface of R, that is, the R2jags package of R, which includes the posterior simulation and convergence diagnostic of the model. For modeling of these data in JAGS, one must specify a model to run, load data which is created in a separate file, and specify the initial values of the model parameters for a specified number of Markov chains. The R2jags package makes use of this feature and provides convenient functions to call JAGS directly from within R. Furthermore, it is possible to work with the results after importing them into R again, for example to create a posterior predictive simulation or, more generally, graphical displays of data and posterior simulation.

Data Creation The first thing to provide to R2jags is the data. Create the data inputs which R2jags needs. This can be a list containing the name of each vector. The data set given in Table 1 can be created in R format. In the case of the vehicle launch data, the outcome is the binary response variable; for each success outcome, the value of the response variable is assigned the value 1, whereas for each failure outcome, the value of the variable is assigned 0. Thus, for the vehicle launch outcomes, the data are created as follows:

```
n <- 11
y <- c(1,0,0,0,1,0,0,0,1,0,0)
jags.data <- list("n","y")
```

where `n` is the total number of binary outcomes, `y` is the response for each trial, 1 for success and 0 for failure, and all these are combined in a list with object `jags.data`.

Model Specification For modeling the launch vehicle data, the response is binary outcomes and hence assumed to follow a binomial distribution, that is,

$$y_i \sim \text{Binomial}(\theta, 1), \quad i = 1, 2, \text{K}, n,$$

where θ is the success probability, the parameter of interest. Alternatively, the Bernoulli distribution (with command `dbern(θ)`) can also be used instead of the binomial with $n = 1$ without any problem. In order to perform Bayesian analysis, the parameter θ is assumed to follow a beta distribution

$$\theta \sim \text{Beta}(1,1)$$

Thus, the specification of the above defined model in JAGS language must be put in a separate file which is then read by JAGS. When working in R, this is most conveniently done using the `cat` function of R, which behaves pretty much like paste with the exception that the result is not a character object but directly written to a specified file. The JAGS code specifying the model, using the `cat` function to put it in the file `model1.jags`, is

```
cat("model{
    for(i in 1:n){
        y[i]~dbin(theta, 1)
        }
    theta~dbeta(1,1)
    }", file="model1.jags.txt")
```

Here, `y` denotes the observed response variable which is `n` long and follows a binomial distribution with parameter `theta` drawn from Beta (1, 1). The Beta(1, 1) distribution is a commonly-used conjugate prior distribution for binomial likelihood with low information.

Initial Values The starting values used to initialize the chain are simply called the *initial values*. To start the MCMC simulation, usually it is necessary to specify a starting value for the chains. In most cases, however, JAGS will be able to generate the initial values itself. In order to be able to monitor convergence, run several chains for each parameter. The starting value for the chains is a named list; names are the parameters used in the model. Each element of the list is itself a list of starting values for the JAGS model or a function creating (possible random) initial values. In this case, there is only one parameter, called `theta`, in the model:

```
inits <- function(){list(theta=runif(1))}
```

Model Fitting Once these structures have been set up, JAGS is called to compile the model and run MCMC simulations to get the posterior inference for `theta`. Before running, it must be decided how many chains are to be run (`n.chain=3`) and for how many iterations (`n.iter=1000`). If the length of burn-in is not specified, `n.bern=floor(n.iter/2)` is used; that is, 500 in this case. Additionally, it is necessary to specify which parameters we are interested in and

set a monitor on each of them. In this case, `theta` is the only parameter which should be monitored. Thus, to start the simulation, the function `jags` of R2jags is used, and its results are assigned to object `Fit`. The results in object `Fit` can conveniently be printed by `print(Fit)`, which prints a detailed summary of the results which are summarized below.

```
Fit <- jags(jags.data,inits,parameter=c("theta"), n.iter=1000,
       n.chain=3, model.file="model1.jags.txt",)
print(Fit)
```

Summarizing Output The output of the R function `jags` is a list which includes several components; most notable are the summary of the inference and convergence and a list containing the simulation draws of all the saved parameters. In this case, the `jags` call is assigned to the R object `Fit`, and so typing `print(Fit)` from the R console will display the summary of the fitted model shown below. The print method displays information on the mean, standard deviation, 95% credible interval (CI) estimates, the effective sample size, and potential scale reduction factor $\hat{R}$ of the Brook-Gelman-Rubin (BGR) statistics (Brooks & Gelman, 1998; Gelman & Rubin, 1992). The BGR statistic is an analysis of variance (ANOVA)-type diagnostic that compares within- and among-chain variance (Kéry, 2010). Values around 1 indicate convergence, with 1.1 considered to be an acceptable limit (Gelman & Hill, 2007).

Represented in Table 2 is the numerical summary output from the `jags` function after being fitted to the binomial model for the success probability of the launch vehicle data. The first five columns of numbers give inferences for the model parameters. In this case, the model parameter theta has a mean estimate 0.311 and a standard error of 0.124. The median estimate of theta is 0.300 with a 50% uncertainty interval of [0.219, 0.391] and a 95% interval of [0.100, 0.584]. At the bottom, pD shows the estimated effective number of parameters in the model, and DIC, the deviance information criterion, an estimate of predictive error. Consider the right-most columns of the output, where Rhat gives information about convergence of the algorithm. At convergence, the number at this column should be equal to 1. If Rhat is less than 1.1 for all parameters, then we judge the algorithm to have approximately converged (Gelman & Hill, 2007) in the sense that the parallel chains have mixed well. The final column n.eff is the effective sample size of the simulations.

Table 2. Summary of JAGS simulations after being fitted to the logistic model for the launch vehicle data

Parameter	Mean	SD	2.50%	Median	97.50%	Rhat	n.eff
theta	0.311	0.124	0.100	0.300	0.584	1.002	1300
deviance	13.792	1.279	12.892	13.303	17.432	1.007	1100
pD	0.8			DIC	14.6		

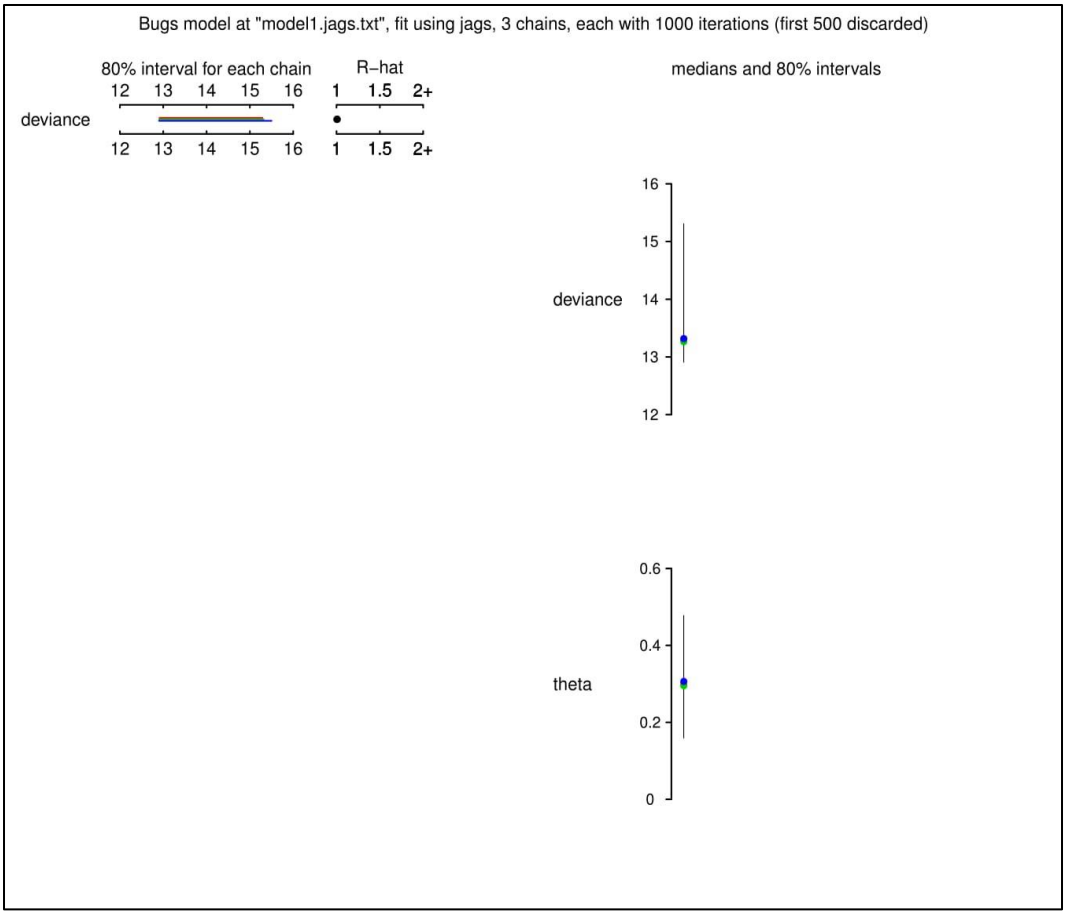

Figure 2. Graphical summary plot of JAGS for the binomial model, fit to the launch vehicle data; R-hat is near to one for parameter theta, indicating good convergence

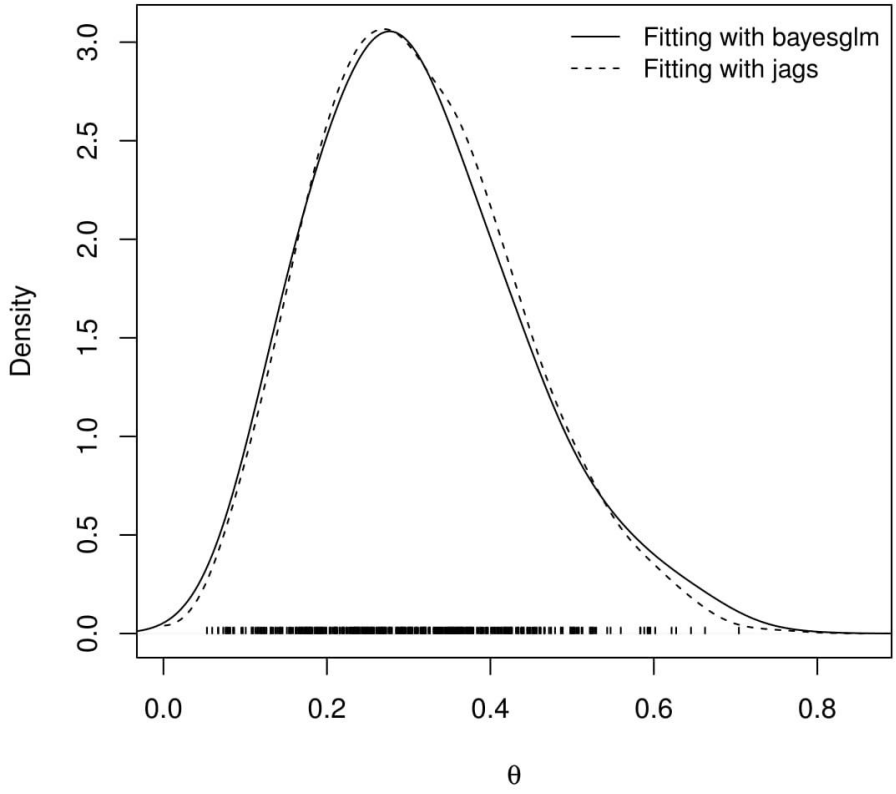

Figure 3. Posterior density plot of the model parameter theta; results from different methods have different styles; the closeness of the two approaches look evident in the graphics

---

To see the complete picture of the results, a plot can be generated by typing `plot(Fit)` and the resulting plot is given in Figure 2. In this plot the left column shows a quick summary of inference and convergence; that is, $\hat{R}$ is close to 1.0 for parameter `theta`, indicating well mixing of the three chains and thus good convergence. The right column shows inference for the parameter `theta`. Figure 3 shows the density plot of the model parameter theta fitted with `bayesglm` and `jags`, respectively.

Analysis of Regression Model

The most popular model for success/failure or binary data is the *logistic regression model*, in which a logit-link function is usually adopted. In this section, Bayesian analysis of the logistic regression model for reliability data is discussed.

The Model

Consider the response variable y_i for $i = 1, 2,\ldots, n_i$ to be binomially distributed ($n_i = 1$ for binary data and $y_i = 1$ or 0) with success probability θ_i, which is denoted by

$$y_i \sim \text{Binomial}\left(n_i, \theta_i\right).$$

The logistic regression model relates θ to the covariates through the link function as

$$\text{logit}\left(\theta_i\right) = \log\left(\frac{\theta_i}{1-\theta_i}\right) = x_i^T \beta \left(= \eta_i\right). \tag{9}$$

A desirable feature of the logit transformation of distribution parameter θ is that it is defined on $(-\infty, \infty)$, so that there are no restrictions on β and, hence, provides flexibility in specifying prior distributions for β. The inverse transformation of equation (9) gives an expression

$$\theta_i = \frac{\exp\left(x_i^T \beta\right)}{1+\exp\left(x_i^T \beta\right)} = \frac{e^{\eta_i}}{1+e^{\eta_i}}, \tag{10}$$

called the *inverse logit function*, having the form of logistic cumulative distribution function which means that there is symmetry about zero. The likelihood contribution for binomial response y_i of the logistic regression model is given by

$$\begin{aligned}
\mathrm{p}\left(y \mid x_i^T \beta\right) &= \prod_{i=1}^{n} \theta_i^{y_i} \left(1-\theta_i\right)^{n_i - y_i} \\
&= \prod_{i=1}^{n} \left(\frac{\exp\left(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\right)}{1+\exp\left(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\right)} \right)^{y_i} \left(\frac{1}{1+\exp\left(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\right)} \right)^{n_i - y_i} \\
&= \exp\left[n\bar{y}\beta_0 + \sum_{i=1}^{n} \left(\sum_{j=1}^{k} \beta_j x_{ij} \right) y_i - \sum_{i=1}^{n} n_i \log\left(1+\exp\left(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \right) \right) \right]
\end{aligned}$$

For $j = 0, 1$ (i.e., β_0, β_1) and $n_i = 1$ (for binary data) the likelihood will be

$$\mathrm{p}\left(y \mid \beta_0, \beta_1\right) = \exp\left[n\bar{y}\beta_0 + \beta_1 \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} \log\left(1+\exp\left(\beta_0 + \beta_1 x_i\right)\right) \right], \qquad (11)$$

where x_i is the vector of covariate values associated with (y_i, n_i).

Regarding the choice of prior distribution for the regression coefficients β, if low information is available about each of the coefficients, one better choice of prior is

$$\beta_j \sim \mathrm{N}\left(0, 10^k\right); \qquad (12)$$

that is a suitably weak prior for sufficiently large values of k, commonly 4 to 6. If more information is available, then the normal distribution with mean possibly different from zero and small variance is a better choice.

When applying Bayes' theorem, the prior is multiplied by the likelihood function and thus the posterior density is

$$\mathrm{p}\left(\beta_0, \beta_1 \mid y, \mathbf{X}\right) \propto \mathrm{p}\left(y \mid \beta_0, \beta_1, \mathbf{X}\right) \mathrm{p}\left(\beta_0\right) \mathrm{p}\left(\beta_1\right),$$

where **X** is a model matrix containing a column of 1s and a column of covariates **x**. Consequently, marginal posterior densities for β_0 and β_1 can be expressed as

$$\mathrm{p}\left(\beta_0 \mid y, \mathbf{X}\right) = \int \mathrm{p}\left(\beta_0, \beta_1 \mid y, \mathbf{X}\right) d\beta_1$$

and

$$\mathrm{p}(\beta_1 \mid y, \mathbf{X}) = \int \mathrm{p}(\beta_0, \beta_1 \mid y, \mathbf{X})\, d\beta_0 \,,$$

which are not in closed form. So, it is very difficult to compute or plot these marginal densities. At this stage one is forced to use analytical or/and simulation tools to implement Bayesian analysis.

Implementation

To implement the above logistic regression model by choosing a normal distribution with large variance as a weakly informative prior for regression coefficients, let us consider a data set taken from Grant et al. (1999); the same data is also discussed in Hamada et al. (2008). All the concepts and computations will be discussed around that data. These data were modeled using a logistic regression model. For the computation of marginal posterior densities of each β, which is in a complex integral form, the Laplace approximation technique is used via the `LaplaceApproximation` function to approximate the integral. Parallel simulation tools are also implemented to draw the samples from marginal posterior densities to approximate the results with the sampling importance resampling (SIR) method (Gordan, Salmond, & Smith, 1993) and one of the MCMC algorithms.

These techniques are implemented using LaplacesDemon and JAGS. The MCMC algorithms used to approximate the integrals via the IM algorithm and MWG algorithm use the `LaplacesDemon` function of LaplacesDemon and `jags` function of R2jags, respectively. An important thing about the IM algorithm is that it updates the model with Laplace approximation, and then supplies the posterior mode and covariance matrix to the IM algorithm. The Laplace approximation is already a well-known approximation technique (Tierney & Kadane, 1986) which accurately approximates the integrals. To use these functions, one must specify a model, a prior for the parameter, and a data object which is required for fitting (Y. Khan, Akhtar, Shehla, & Khan, 2015).

High-Pressure Coolant Injection System Demand Data

The reliability of U.S. commercial nuclear power plants is an extremely important consideration in managing public health risk. The high-pressure coolant injection (HPCI) system is a frontline safety system in a boiling water reactor (BWR) that injects water into a pressurized reactor core when a small break loss-of-coolant accident occurs. Grant et al. (1999) listed 63 unplanned demands to start for HPCI systems at 23 U.S. commercial BWRs during 1987-1993. Table 3 presents these demands in which all failures are counted together, including failure to start, failure

to run, failure of the injection valve to reopen after operating successfully earlier in the mission, and unavailability because of maintenance (Hamada et al., 2008). In data table, asterisks (*) identify the 12 demands for which HPCI system failed.

Fitting with LaplaceApproximation

Laplace approximation is a family of asymptotic techniques used to approximate integrals. It seems to accurately approximate unimodal posterior moments and marginal posterior densities in many cases. Use the function `LaplaceApproximation` for fitting of a logistic regression model, which is an implementation of Laplace's approximations of the integrals involved in the Bayesian analysis of the parameters in the modeling process. This function deterministically maximizes the logarithm of the un-normalized joint posterior density using one of the several optimization techniques. The aim of Laplace approximation is to estimate posterior mode and variance of each parameter. For getting posterior modes of the log-posteriors, a number of optimization algorithms are implemented. This includes the Levenberg-Marquardt (LM) algorithm, which is the default. However, it was found the Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) is a better alternative in Bayesian scenario. The L-BFGS algorithm is a quasi-Newton optimization algorithm that compactly approximates the Hessian matrix. Rather than storing the dense Hessian matrix, L-BFGS stores only a few vectors that represent the approximation. It may be noted that Newton-Raphson is the last choice as it is very sensitive to the starting values; it creates problems when starting values are far from the targets and calculating and inverting the Hessian matrix can be computationally expensive, although it is also implemented in `LaplaceApproximation`. The main arguments of `LaplaceApproximation` can be seen by using the function `args` as

Table 3. Dates of unplanned HPCI system demands and failures during 1987-1993

01/05/87*	08/03/87*	03/05/89	08/16/90*	08/25/91
01/07/87	08/16/87	03/25/89	08/19/90	09/11/91
01/26/87	08/29/87	08/26/89	09/02/90	12/17/91
02/18/87	01/10/88	09/03/89	09/27/90	02/02/92
02/24/87	04/30/88	11/05/89*	10/12/90	06/25/92
03/11/87*	05/27/88	11/25/89	10/17/90	08/27/92
04/03/87	08/05/88	12/20/89	11/26/90	09/30/92
04/16/87	08/25/88	01/12/90*	01/18/91*	10/15/92
04/22/87	08/26/88	01/28/90	01/25/91	11/18/92
07/23/87	09/04/88*	03/19/90*	02/27/91	04/20/93
07/26/87	11/01/88	03/19/90	04/23/91	07/30/93
07/30/87	11/16/88*	06/20/90	07/18/91*	
08/03/87*	12/17/88	07/27/90	07/31/91	

```
function(Model, parm, Data, Interval=1e-06, Iterations=100,
         Method="LM", Samples=1000, sir=TRUE, Stop.Tolerance=1e-
         05)
NULL
```

The first argument `Model` defines the model to be implemented, which contains specification of likelihood and prior. `LaplaceApproximation` passes two argument to the model function, `parm` and `Data`, and receives five arguments from the model function: `LP` (the logarithm of the unnormalized joined posterior density), `Dev` (the deviance), `Monitor` (the monitored variables), `yhat` (the variables for the posterior predictive checks), and `parm`, the vector of parameters, which may be constrained in the model function. The argument `parm` requires a vector of initial values equal in length to the number of parameters, and `LaplaceApproximation` will attempt to optimize these initial values for the parameters, where the optimized values are the posterior modes. The `Data` argument requires a list of data which must be include variable names and parameter names. The argument `sir=TRUE` stands for implementation of the sampling importance resampling algorithm, which is a bootstrap procedure to draw independent samples with replacement from the posterior sample with unequal sampling probabilities. Contrary to `sir` in the `LearnBayes` package, here proposal density is multivariate normal and not *t* (Akhtar & Khan, 2014a, b).

The first thing is to provide data as LaplacesDemon needs. For this, the data set given in Table 3 can be created in R format as follows:

Data Creation In an HPCI system demand data set, the binary response variable is failure denoted by y = 1 (0), where 1 stands for failure and 0 for success, and an explanatory variable or covariate is time, which denotes the number of elapsed days from a chosen reference data 01/01/87, for 63 demands. To calculate the number of elapsed days for each demand from the reference date, the function `difftime` of R is used. This function takes two date-time objects as its arguments to calculate the time difference between two dates and returns an object of class `difftime` with an attribute indicating the units. Then the function `as.numeric` is used to convert the obtained time difference (that is, the number of elapsed days) into numeric form as only the number of days is needed to use in analysis. Since an intercept term will be included, a vector of 1s is inserted into the model matrix **X**. Thus, $J = 2$ indicates that, there are two columns in model matrix; the first column for the intercept term and the second column for the regressor in the design matrix.

```
N <- 63; J <- 2
y <- c(1,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,
       0,1,0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,
       0,0,0,0,0)
time <- difftime(c("1987-01-05","1987-01-07","1987-01-26",
                   "1987-02-18","1987-02-24","1987-03-11",
                   "1987-04-03","1987-04-16","1987-04-22",
                   "1987-07-23","1987-07-26","1987-07-30",
                   "1987-08-03","1987-08-03","1987-08-16",
                   "1987-08-29","1988-01-10","1988-04-30",
                   "1988-05-27","1988-08-05","1988-08-25",
                   "1988-08-26","1988-09-04","1988-11-01",
                   "1988-11-16","1988-12-17","1989-03-05",
                   "1989-03-25","1989-08-26","1989-09-03",
                   "1989-11-05","1989-11-25","1989-12-20",
                   "1990-01-12","1990-01-28","1990-03-19",
                   "1990-03-19","1990-06-20","1990-07-27",
                   "1990-08-16","1990-08-19","1990-09-02",
                   "1990-09-27","1990-10-12","1990-10-17",
                   "1990-11-26","1991-01-18","1991-01-25",
                   "1991-02-27","1991-04-23","1991-07-18",
                   "1991-07-31","1991-08-25","1991-09-11",
                   "1991-12-17","1992-02-02","1992-06-25",
                   "1992-08-27","1992-09-30","1992-10-15",
                   "1992-11-18","1993-04-20","1993-07-30"),
                   "1987-01-01")
time <- as.numeric(time)
time <- time-mean(time)
X <- cbind(1,as.matrix(time))
mon.names <- "LP"
parm.names <-as.parm.names(list(beta=rep(0,J)))
PGF <- function(Data) return(rnormv(Data$J,0,1000))
MyData <- list(N=N, J=J, PGF=PGF, X=X, mon.names=mon.names,
          parm.names=parm.names, y=y)
```

In this case, there are two parameters, `beta[1]` and `beta[2]`, which are specified in a vector `parm.names`. The logposterior `LP` is included as a monitored variable in vector `mon.names`. The total number of observations is specified by `N`, i.e., 63.

Finally, all these things are combined with object name `MyData` that returns data in a list.

Initial Values Initial values are a starting point for the iteration of the optimization of a parameter. `LaplacesApproximation` requires a vector of initial values for each parameter to start the iterations. Both the β parameters have been set equal to zero with object name `Initial.Values` as

```
Initial.Values <- rep(0,J)
```

For initial values, the function `GIV` (which stands for "Generate Initial Values") may also be used to randomly generate initial values.

Model Specifications For modeling these HPCI data where the response variable is binary (since each demand results in either an HPCI failure or success), use the binomial distribution with $n = 1$. An explanatory variable or regressor is the number of elapsed days denoted by `time`. Thus, the model specified is a logistic regression and can be described as

$$y_i \sim \text{Binomial}(\theta_i, 1), \quad i = 1, 2, \text{K}, 63,$$

where $y_i = 1$ (0) denotes an HPCI failure (success). The logit link function (other link functions like probit or complementary log-log are also possible) is used to relate the model parameter and regressor time, that is,

$$\text{logit}(\theta_i) = \log\left(\frac{\theta_i}{1-\theta_i}\right) = \beta_0 + \beta_1 \text{time}_i,$$

where time is centered, that is, `time=time-mean(time)`. The linear predictor is made up of an intercept β_0 and a regressor time with regression coefficient β_1.

The independent and weakly informative normal prior with mean 0 and standard deviation 1000 is assumed for each β parameter:

$$\beta_j = \text{N}(0, 1000), \quad j = 1, 2, \text{K}, J.$$

The large variance and small precision indicates a lot of uncertainty for each β, and is hence a weakly informative prior. Finally, the specification of the above defined

logistic regression model for fitting with `LaplaceApproximation` is defined as follows:

```
Model <- function(parm, Data){
### Parameters
beta <- parm[1:Data$J]
### Log(Prior Densities)
beta.prior <- sum(dnorm(beta, 0, 1000, log=TRUE))
### Log-Likelihood
mu <- tcrossprod(Data$X, t(beta))
theta <- invlogit(mu)
LL <- sum(dbinom(Data$y, 1, theta, log=TRUE))
### Log-Posterior
LP <- LL + beta.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP,
            yhat=rbinom(length(theta), 1, theta), parm=parm)
return(Modelout)}
```

The function `Model` contains two arguments, `parm` and `Data`, where `parm` is the set of parameters and `Data` is the list of data. The regression parameters have priors `beta.prior`. The object `LL` stands for loglikelihood and `LP` stands for logposterior. The function `Model` returns the object `Modelout`, which contains five objects in listed form that includes logposterior `LP`, deviance `Dev`, monitoring parameters `Monitor`, predicted values `yhat`, and estimates of parameters `parm`.

Model Fitting For fitting of the logistic regression model with weakly informative priors for the regression parameters, the function `LaplaceApproximation` of LaplacesDemon is called, and its results are assigned to the object `Fit`. Summaries of results are printed using the function `print`, and its relevant parts are summarized in the next section.

```
Fit <- LaplaceApproximation(Model=Model, parm=Initial.Values,
       Data=MyData, Method="NM", Iterations=10000, sir=TRUE)
print(Fit)
```

Table 4. Summary of the analytic approximation using the function LaplaceApproximation; it may be noted that these summaries are based on asymptotic approximation, and hence Mode stands for posterior mode, SD for posterior standard deviation, and LB, UB are 2.5% and 97.5% quantiles, respectively

Parameters	Mode	SD	LB	UB
beta [1]	−1.4902	0.3332	−2.1565	−0.8238
beta [2]	−0.0006	0.0005	−0.0015	0.0004

Table 5. Summary of the simulated results due to sampling importance resampling method using the same function, where Mean stands for posterior mean, SD for posterior standard deviation, MCSE for Monte Carlo standard error, ESS for effective sample size, and LB, Median, UB are 2.5%, 50%, 97.5% quantiles, respectively

Parameters	Mean	SD	MCSE	ESS	LB	Median	UB
beta [1]	−1.5404	0.3397	0.0107	1000	−2.1981	−1.5248	−0.8238
beta [2]	−0.0006	0.0005	0.0000	1000	−0.0016	−0.0006	0.0004
Deviance	62.1583	2.2389	0.0708	1000	60.0814	61.4733	69.2003
LP	−46.7325	1.1195	0.0354	1000	−50.2535	−46.3900	−45.6941

Summarizing Output The summary information provided by `LaplaceApproximation`, which approximates posterior density of the fitted model, is summarized in the two tables which follow. Table 4 represents the summary matrix of the analytic results using Laplace's approximation method. Rows are parameters and columns include Mode, SD (Standard Deviation), LB (Lower Bound), and UB (Upper Bound). The bound constitutes a 95% credible interval. Table 5 represents the simulated results due to the sampling importance resampling algorithm conducted via the `SIR` function to draw independent posterior samples, which is possible only when `LaplaceApproximation` has converged.

From the summary output of Laplace's approximation method reported in Table 4, it may be noted that the posterior mode of parameter β_0 is -1.4902 ± 0.3332 with 95% credible interval $(-2.1565, -0.8238)$, which is statistically significant, whereas the posterior mode of β_1 is 0.0006 ± 0.0005 with 95% credible interval $(-0.0015, 0.0004)$, which is statistically not significant. The simulated results due to sampling importance resampling algorithm using the same function reported in Table 5 indicates the posterior mean of β_0 is -1.5404 ± 0.3397 with 95% credible interval $(-2.1981, -0.9335)$, whereas the posterior mean of β_1 is -0.0006 ± 0.0005 with 95% credible interval $(-0.0016, 0.0004)$. The deviance 62.16 is a measure of goodness-of-fit.

Fitting with LaplacesDemon

The simulation method is applied to analyze the same data with the function `LaplacesDemon`, which is the main function of Laplace's Demon. Given data, a model specification, and initial values, `LaplacesDemon` maximizes the logarithm of the unnormalized joint posterior density with MCMC algorithms, also called samplers, and provides samples of the marginal posterior distributions, deviance, and other monitored variables. Laplace's Demon offers a large number of MCMC algorithms for numerical approximation. Popular families include: Gibbs sampling, Metropolis-Hasting (MH), Random-Walk-Metropolis (RWM), slice sampling, Metropolis-within Gibbs (MWG), Adaptive-Metropolis-within-Gibbs (AMWG), and many others. However, details of the MCMC algorithms are best explored online at https://web.archive.org/web/20150227012508/http:/www.bayesian-inference.com/mcmc, as well as in the "LaplacesDemon Tutorial" vignette (Statisticat, LLC, 2018). The main arguments of the `LaplacesDemon` function can be seen by using the function `args` as

```
function(Model, Data, Initial.Values, Covar= NULL, Iterations=
         1e+05, Status= 1000, Thinning= 100, Algorithm= "RWM",
         Specs= NULL,...)
NULL
```

The arguments `Model` and `Data` specify the model to be implemented and list of data, respectively, which are specified in the previous section. `Initial.Values` requires a vector of initial values equal in length to the number of parameters. The argument `Covar=NULL` indicates that a variance vector or covariance matrix has not been specified, so the algorithm will begin with its own estimates. Next two arguments `Iterations=100000` and `Status=1000` indicate that the `LaplacesDemon` function will update 10000 times before completion and status is reported after every 1000 iterations. The `thinning` argument accepts integers between 1 and the number of iterations, and indicates that every 100th iteration will be retained, while the others are discarded. Thinning is performed to reduce autocorrelation and the number of marginal posterior samples. Further, `Algorithm` requires the abbreviated name of the MCMC algorithm in quotes. In this case `RWM` is short for the Random-Walk-Metropolis. Finally, `Specs= Null` is a default argument, and accepts a list of specifications for the MCMC algorithm declared in the `Algorithm` argument (Akhtar & Khan, 2014a, b).

Initial Values Before fitting the model with `LaplacesDemon` it is necessary to specify initial values for each parameter as a starting point for an adaptive chain or a non-adaptive Markov chain. If initial values for all parameters are set to zero, then `LaplacesDemon` will attempt to optimize the initial values with `LaplacesApproximation` using a resilient backpropagation algorithm. Hence, it is better to use the last fitted object `Fit` with the function `as.initial.values` to get a vector of initial values from `LaplacesApproximation` for fitting with `LaplacesDemon`. Thus, to get a vector of initial values the R command is

```
Initial.Values <- as.initial.values(Fit)
```

Model Fitting `LaplacesDemon` is an implementation of stochastic approximation methods; therefore, it is better to set a seed with the `set.seed` function for pseudo-random number generation before fitting the model, so results can be reproduced. Call the `LaplacesDemon` function to maximize the first component in the list output from the pre-specified `Model` function, given a data set called `Data` with the following setting. The fitted model specifies the `IM` (Independent Metropolis) algorithm for updating. Simulation results are assigned to the object `FitDemon`, and its relevant parts are summarized in the next section.

```
set.seed(666)
FitDemon <- LaplacesDemon(Model=Model, Data=MyData, Initial.Values,
    Covar=Fit$Covar, Iterations=50000, Status=1000, Algorithm="IM",
    Specs=list(mu=Fit$Summary1[1:length(Initial.Values),1]))
print(FitDemon)
```

Summarizing Output The simulated results are shown in Table 6 in a matrix form that summarizes the marginal posterior densities of parameters due to stationary samples which includes Mean, SD (Standard Deviation), MCSE (Monte Carlo Standard Error), ESS (Effective Sample Size), and 2.5%, 50%, and 97.5% quantiles.

Laplace's Demon is appeased, because all five criteria `LaplacesDemon` needs are satisfactory. The final algorithm must be non-adaptive, so that the Markov property holds. The acceptance rate of most algorithms is considered satisfactory if it is within the interval [15%, 50%]. MCSE is considered satisfactory for each target distribution if it is less than 6.27% of the standard deviation of the target distribution. This allows the true mean to be within 5% of the area under a Gaussian distribution around the estimated mean. ESS is considered satisfactory for each target

distribution if it is at least 100, which is usually enough to describe 95% probability intervals. Each variable must be estimated as stationary.

From Table 6, it is noticed all criteria were met: MCSEs are sufficiently small, ESSs are sufficiently large, all parameters were estimated to be stationary, the algorithm is non-adaptive independent Metropolis and the Markov property holds, and acceptance rate 0.49 (i.e., 49%) of the algorithm lies within the interval [15%, 50%].

Fitting with JAGS

In this section, JAGS is called from within R via R2jags to conduct the same Bayesian analysis of a logistic regression model for the HPCI data. After fitting the model with `jags`, a comparison will be made with the results obtained from `LaplaceApproximation` and `LaplacesDemon`, illustrated in previous sections.

Data Creation The first step is to provide data to JAGS in a list statement. Provide the vector of HPCI response on demands over time. The specification of these data containing the name of each vector, as `jags` needs, is as follows:

```
n <- 63
y <- y
time <- as.vector(time)
time <- time-mean(time)
data.jags <- list(n=n, y=y, time=time)
```

Here, `n` is the total number of observations, `y` is the observed response for each demand, and `time` is the number of elapsed days for 63 demands, which is centered for improving convergence.

Table 6. Summary of the MCMC results due to Independent Metropolis algorithm using the LaplacesDemon function

Parameters	Mean	SD	MCSE	ESS	LB	Median	UB
beta [1]	−1.4956	0.1979	0.0028	4666	−1.8852	−1.4932	−1.1164
beta [2]	−0.0006	0.0003	0.0000	5000	−0.0011	−0.0006	0.0000
Deviance	60.7504	0.6960	0.0103	4361	60.0622	60.5393	62.6243
LP	−46.0286	0.3480	0.0052	4361	−46.9605	−45.9230	−45.6845

Initial Values Initial values are used to start the MCMC sampler and may be provided for all stochastic nodes except for the response data variable. The initial values for parameters to start the chains are specified by writing a function as follows:

```
inits <- function(){list(beta.1=0, beta.2=0)}
```

Model Specification For modeling the HPCI data, the binomial distribution is adopted to model the binary response variable. The model is defined as

$$y_i \sim \text{Binomial}(\theta_i, 1), \quad i = 1, 2, \text{K}, 63$$

with logit-link function

$$\text{logit}(\theta_i) = \log\left(\frac{\theta_i}{1-\theta_i}\right) = \beta_0 + \beta_1 \text{time}_i$$

where θ_i is the success probability. The weakly informative normal priors with mean 0 and precision 0.001 are defined for each β parameter as

$$\beta_j \sim \text{N}(0, 0.001), \quad j = 1, 2$$

The JAGS code specifying the above logistic regression model using cat function to put in a file with name `modelHPCI.txt` is

```
Cat("model{
      for(i in 1:n){
            y[i]~dbin(theta[i], 1)
            logit(theta[i])<-beta.1+beta.2*time[i]
            }
      # Priors
      beta.1~dnorm(0, 0.001)
      beta.2~dnorm(0, 0.001)
      }", file="modelHPCI.txt")
```

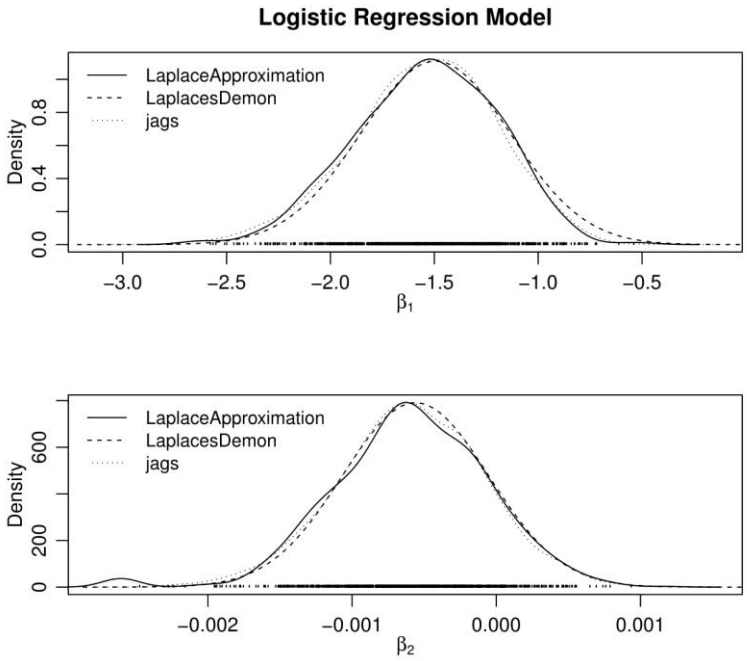

Figure 4. Plot of posterior densities of the parameters β_1 and β_2 for the Bayesian analysis of logistic regression model using the functions LaplaceApproximation, LaplacesDemon, and jags, respectively; it is evident from these plots that the posterior densities obtained from three different methods are very close to each other

Table 7. Posterior summary of the JAGS simulations after being fitted to the logistic regression model for HPCI data

Parameters	Mean	SD	2.50%	50%	97.50%	Rhat	n.eff
beta.1	−1.5470	0.3490	−2.2894	−1.5330	−0.9223	1.00	3000
beta.2	−0.0006	0.0005	−0.0016	−0.0006	0.0004	1.00	3000
deviance	62.0873	2.0489	60.095	61.4526	67.6542	1.00	2700

Model Fitting A Bayesian model with weakly informative priors is fitted using the `jags` function and its results are assigned to object `Fit.Jags`. A summary of results is reported in next section.

```
set.seed(123)
Fit.jags <- jags(data.jags, inits, parameters=c("beta.1",
     "beta.2"), n.iter=20000, model.file="modelHPCI.txt")
print(Fit.Jags)
```

Summarizing Output Shown in Table 7 is the summary output of the posterior densities after being fitted to the logistic regression model for HPCI data. From the JAGS output, it is noticed that the value of the posterior mean of both beta parameters are very close to the values obtained from the SIR and IM algorithms, although β_0 is statistically significant whereas β_1 is not significant. Its deviance of 62.09 is almost equal to the deviance 62.16 via SIR, and slightly differs from the deviance 60.75 of IM. The values of Rhat are less than 1.1 for each parameter, which indicates that chains are mixed well, implying good convergence. The values of n.eff are also satisfactory.

Conclusion

Component reliability is the foundation of reliability assessment and refers to the reliability of a single component. The Bayesian approach is used to model success/failure component reliability data for both intercept and regression models. The logit-link function is used to relate the model parameter and covariates for a linear regression model, which is known as a logistic regression model. Two important techniques, that is, Laplace approximation and simulation methods are implemented using the R, LaplacesDemon, and JAGS software packages. For modeling success/failure data, the complete R and JAGS code are written and provided with detailed descriptions. It is observed that the results obtained from Laplace approximation and simulation methods using different software packages are very close to each other. The benefits of Laplace approximation and simulation methods seem clear in the plot of posterior densities. Moreover, it is evident from the summaries of results that the Bayesian approach based on weakly informative priors is simpler to implement than the frequentist approach. Finally, the wealth of information provided in these numeric as well as graphic summaries is not possible in classical framework.

References

Akhtar, M. T., & Khan, A. A. (2014a). Bayesian analysis of generalized log-Burr family with R. *SpingerPlus, 3*, 185. doi: 10.1186/2193-1801-3-185

Akhtar, M. T., & Khan, A. A. (2014b). Log-logistic distribution as a reliability model: A Bayesian analysis. *American Journal of Mathematics and Statistics, 4*(3), 162-170. doi: 10.5923/j.ajms.20140403.05

Bernado, J. M., & Smith, A. F. M. (2000). *Bayesian theory*. Chichester, UK: John Wiley & Sons, Ltd. doi: 10.1002/9780470316870

Brooks, S. P., & Gelman, A. (1998). General method for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics, 7*(4), 434-455. doi: 10.1080/10618600.1998.10474787

Gelfand, A. E., Hills, S. E., Racine-Poon, A., & Smith, A. F. M. (1990). Illustration of Bayesian inference in normal data models using Gibbs sampling. *Journal of the American Statistical Association, 85*(412), 972-985. doi: 10.1080/01621459.1990.10474968

Gelfand, A. E., & Smith, A. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association, 85*(410), 398-409. doi: 10.1080/01621459.1990.10476213

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian Analysis, 1*(3), 515-533. doi: 10.1214/06-BA117A

Gelman, A. (2008). Scaling regression inputs by dividing by two standard deviations. *Statistics in Medicine, 27*(15), 2865-2873. doi: 10.1002/sim.3107

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis* (2nd ed.). Boca Raton, FL: Chapman & Hall/CRC Press.

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. New York, NY: Cambridge University Press.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science, 7*(4), 457-511. doi: 10.1214/ss/1177011136

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6*(6), 721-741. doi: 10.1109/TPAMI.1984.4767596

Gordan, N. J., Salmond, D. J., & Smith, A. F. M. (1993). Novel approach to non-linear/non-Gaussian Bayesian state estimation. *IEEE Proceedings F (Radar and Signal Processing), 140*(2), 107-117. doi: 10.1049/ip-f-2.1993.0015

Grant, G. M., Roesener, W. S., Hall, D. G., Atwood, C. L., Gentillon, C. D., & Wolf, T. R. (1999). Reliability study: High-pressure coolant injection (HPCI) system, 1987-1993 (technical report NUREG-CR-5500, Vol. 4, INEL-94-0158). Idaho Falls, ID: Idaho National Engineering and Environmental Laboratory.

Hamada, M. S., Wilson, A. G., Reese, C. S., & Martz, H. F. (2008). *Bayesian reliability*. New York, NY: Springer. doi: 10.1007/978-0-387-77950-8

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57*(1), 97-109. doi: 10.1093/biomet/57.1.97

Johnson, V. E., Moosman, A., & Cotter, P. (2005). A hierarchical model for estimation the early reliability of complex systems. *IEEE Transactions on Reliability, 54*(2), 224-231. doi: 10.1109/TR.2005.847262

Kéry, M. (2010). *Introduction to WinBUGS for ecologists: A Bayesian approach to regression, ANOVA, mixed models and related analyses*. Burlington, MA: Academic Press. doi: 10.1016/C2009-0-30639-X

Khan, N., Akhtar, M. T., & Khan, A. A. (2016). A Bayesian approach to survival analysis of inverse Gaussian model with Laplace approximation. *International Journal of Statistics and Applications, 6*(6), 391-398. Retrieved from http://article.sapub.org/10.5923.j.statistics.20160606.08.html

Khan, N., Akhtar, T., & Khan, A. A. (2017). Bayesian analysis of Marshall and Olkin family of distributions. *International Journal of Recent Scientific Research, 8*(7), 18692-18699. Retrieved from http://recentscientific.com/bayesian-analysis-marshall-and-olkin-family-distributions

Khan, Y., Akhtar, M. T., Shehla, R., & Khan, A. A. (2015). Bayesian modeling of forestry data with R using optimization and simulation tools. *Journal of Applied Analysis and Computation, 5*(1), 38-51. doi: 10.11948/2015004

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., & Teller, A. H. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21*(6), 1087-1092. doi: 10.1063/1.1699114

Mosteller, F., & Wallace, D. C. (1964). *Inference and disputed authorship:* The federalist. Reading, MA: Addison-Wesley.

Ntzoufras, I. (2009). *Bayesian modeling using WinBUGS*. New York, NY: John Wiley & Sons.

Plummer, M. (2003, March). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In K. Hornik, F. Leisch, & A. Zeileis (Eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.r-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf

R Core Team. (2018). R: A language and environment for statistical computing [computer program]. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.r-project.org/

Statisticat, LLC. (2018). LaplacesDemon: Complete environment for Bayesian inference [R package, version 16.1.1]. Retrieved from https://cran.r-project.org/package=LaplacesDemon

Su, Y. & Yajima, M. (2014). R2jags: A package for running JAGS from R [R package, version 0.04-01]. Retrieved from https://cran.r-project.org/package=R2jags

Tanner, M. A. (1996). *Tools for statistical inference: Methods for the exploration of posterior distributions and likelihood functions*. New York, NY: Springer-Verlag. doi: 10.1007/978-1-4612-4024-2

Tanner, M. A., & Wong, W. H. (1987). The calculation of the posterior distributions by data augmentation. *Journal of the American Statistical Association, 82*(398), 82-86. doi: 10.1080/01621459.1987.10478458

Tierney, L., & Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association, 81*(393), 82-86. doi: 10.1080/01621459.1986.10478240

Tierney, L., Kass, R. E. & Kadane, J. B. (1989). Fully exponential Laplace approximations to expectations and variances of nonpositive functions. *Journal of the American Statistical Association, 84*(407), 710-716. doi: 10.1080/01621459.1989.10478824