Journal of Statistics Applications & Probability
An International Journal

http://dx.doi.org/10.12785/jsap/040108

Simulation-based Analysis of Degradation Data: Implementation of a Multilevel Approach

Romana Shehla[*] *and Athar Ali Khan*

Department of Statistics and Operations Research, Aligarh Muslim University, Aligarh-202002, India

Received: 27 Aug. 2014, Revised: 14 Dec. 2014, Accepted: 20 Dec. 2014
Published online: 1 Mar. 2015

Abstract: Data often arrive with hierarchical structure and multilevel regression modeling is the most popular approach to handle such data. This paper demonstrates how multilevel model can be analyzed in Bayesian framework, with reference to a practical degradation data problem. Assuming a varying-intercept, varying-slope model for the data, the exact as well as the approximate inference procedures have been developed using R and JAGS and their performance have been compared. Further, the concept of Bayesian p-values have been discussed to assess the adequacy of the proposed model.

Keywords: Multilevel model, Bayesian inference, Degradation, JAGS, Gibbs sampler, lme4, Bayeian p-value

1 Introduction

Lifetime analysis has always been the cornerstone of reliability assessments. For products that are highly reliable, assessment of reliability using lifetime data becomes quite cumbersome. Recently, degradation data has emerged to be a superior alternative to highly censored lifetime data. The analyst neither have to wait for failures to occur nor have to look for any accelerating relationship, (see e.g.[1]). Failures usually occur from a degradation mechanism working continuously within the items for which there are several characteristics that degrade (or grow) over time. The experimenter needs to choose one of the degrading characteristics that can be appropriately related to failure. Thus, with the degradation data, we define the failure of item in terms of observable characteristics. The item is said to be failed when the amount of degradation exceeds that some pre-specified threshold level of degradation. These kind of failures are termed as soft failures.

Much literature is available on modeling of degradation (or soft failures) data. There are two major approaches for degradation data modeling. The first approach assumes degradation to be a random process in time. Many authors have worked in this area. Few of them have been mentioned here. [2] used a Wiener process model to analyze degradation data. [3] considered that the degradation process in the model is taken to be a Wiener diffusion process with a time scale transformation. The alternative approach is to use general degradation path models. [4] developed statistical methods using degradation measures to estimate a time-to-failure distribution for a broad class of degradation models. Nonlinear mixed effects model was considered by them and point estimates and confidence intervals of percentiles of the failure time distribution were obtained by a two-stage method. [1] presented a case study which used degradation data and a fractional factorial design to improve the reliability of fluorescent lamps. After the construction of appropriate degradation path model, parameters of it were estimated. [5] proposed an approximated maximum likelihood estimator of the parameters of multivariate normal random effects. The functions `LME` and `NLME` were written in S-PLUS specifically for this purpose. The approach so far in dealing with these kind of problems was to estimate model parameters thereby using them to define failure-time distribution.

Difficulties with the existing approach

The closed form of failure-time distribution can be obtained for simple path models but the complexity arises when the functional form of actual degradation path is non-linear and the model has more than one random parameter. The

[*] Corresponding author e-mail: romana.stats@gmail.com

specification of failure-time distribution in this situation becomes a challenge and one has to evaluate it numerically by using any of the several simulation techniques. Moreover, the methods used so far mostly relied on maximum likelihood or least squares estimation of the model parameters which tend to work well when the sample size is moderate to large but for small sample sizes, the procedure becomes biased. Thus, Bayesian methods are the only alternative that can work efficiently even in small sample size situation. Although there is a vast literature available on the degradation analysis with Bayesian approach, we have worked here differently. In this piece of work, an attempt has been made to demonstrate the approximate as well as exact Bayesian analysis of degradation data. For the purpose of illustrations, a real degradation data has been considered. The whole demonstration is made using the function `lmer` available with the lme4 package of R (see [6]), as one of the optimization tool and the function `jags` present in R2jags package which implements Gibbs sampling for the posterior analysis and serves as a simulation tool. Also, the performance of both the tools have been juxtaposed. Conclusions are made directly on the basis of multilevel linear regression analysis of the degradation data problem.

2 Practical motivating situation: Drug potency data

Potency of a drug is measured in terms of the amount of it required to produce an effect of specified intensity. Since, it is a degrading quantity, the companies perform a stability study to determine survival time of a drug being produced. [7] carried out a stability study on 24 batches of a drug over a 36-month period. The lifetime (shelf life) of a drug is the length of time it takes for the drug's potency to decrease to 90% of its original stated potency. Table 1 presents observed degradation at different time periods for each of 24 batches.

Table 1: Drug potency (in percent of original stated potency)

	Time(months)					Time(months)			
Batch	0	12	24	36	Batch	0	12	24	36
1	99.9	98.9	95.9	92.9	13	99.8	98.8	93.8	89.8
2	101.1	97.1	94.1	91.1	14	100.1	99.1	93.1	90.1
3	100.3	98.3	95.3	92.3	15	100.7	98.7	93.7	91.7
4	100.8	96.8	94.8	90.8	16	100.3	98.3	96.3	93.3
5	100.0	98.0	96.0	92.0	17	100.2	98.2	97.2	94.2
6	100.1	98.1	98.1	95.1	18	99.8	97.8	95.8	90.8
7	99.6	98.6	96.6	92.6	19	100.8	98.8	95.8	94.8
8	100.4	99.4	96.4	95.4	20	100.0	98.0	96.0	92.0
9	100.9	98.9	96.9	96.9	21	99.6	99.6	92.6	88.6
10	100.5	99.5	94.5	93.5	22	100.2	98.2	97.2	94.2
11	101.1	98.1	93.1	91.1	23	99.8	97.8	95.8	90.8
12	100.9	97.9	95.9	93.9	24	100.0	99.0	95.0	92.0

Hamada et al. [8] analysed the same dataset considering a linear degradation path model for this data as

$$D_i(t) = D(t, \theta_i) = 100 - (1/\theta_i)t$$

They considered normal measurement errors for each degradation observations. The analysis was done in Hierarchical Bayes' framework with log-normal prior densities for each θ_i. Failure-time distribution was evaluated for reliability assessments.

The graphic summary of the data can be obtained with Figure 1. On the basis of this graphic summary, we propose a different framework to analyze the same drug potency degradation data. We can detect a linear relationship between the amount and time of degradation of drug amongst various batches. Moreover, it is evident that the intercepts and the slopes are varying by batches. Keeping this in view, we propose a model discussed in Section 3.

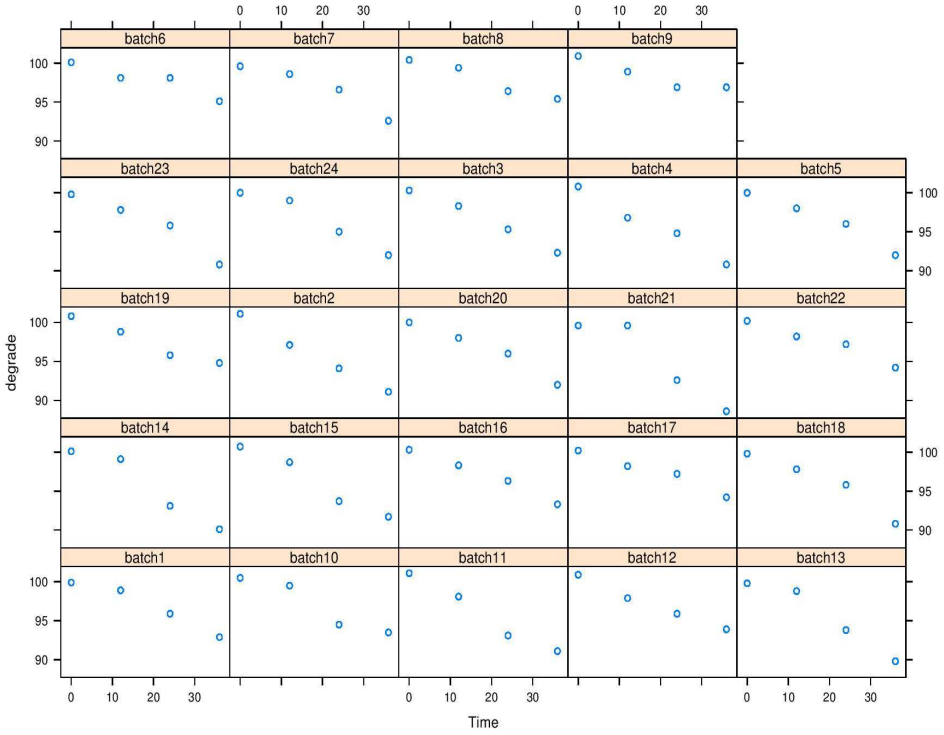

Fig. 1: Trellis plot of degradation-time relationship in 24 drug batches

3 Bayesian Analysis of the data

We proceed with the Bayesian analysis for the above dataset. In the first section, exact Bayesian analysis using simulation tool has been presented whereas in the next section, we employ the optimization tool for the analysis of the data.

3.1 Exact Bayesian Analysis using simulation technique

It can be seen that the data is structured hierarchically: units within batches. Four units from each of the batches were observed for degradation for different time periods. Here, we have unit-level predictor as the time (in months). Our goal in analyzing this data is to find the degradation trend within the batches and conclude which one of the 24 batches of drug is the most suitable for releasing into the market. A separate regression model can be fit within each batch and thus we have unit-level and batch-level as the two levels of multilevel regression. Since, the intercepts and the slopes are differing amongst 24 batches, in this situation the resulting model will have three levels: data y, intercepts α and slopes β , each having a different variance components σ_y, σ_α and σ_β respectively. Computation of explained variance for both the unit-level and batch-level model is greatly affected by the uncertainty in α and β parameters which motivated us to fit a single fitted multilevel model for this dataset.

The fitting of multilevel model brings a challenge of estimating the data level regression and group-level regression at once. Thus, multilevel models are commonly evaluated in Bayesian framework where the data-level model is treated as the likelihood function and the group-level model provides the prior information to estimate the individual-level coefficients. So, a multilevel model has its own hierarchy, with the parameters of the data-level model at the bottom, controlled by the hyperparameters of the group-level model. Therefore, a multilevel model is also referred to as *hierarchical model* (see [9]). Using Bayes' rule, we combine the prior information about the unknown parameters along with the likelihood to compute the posterior density which represents an updated knowledge about the parameters.

We generally get stuck in analytically deriving the posteriors for more complicated models. With the great advances in computing power, simulation has become the major attraction for Bayesian data analysts. Making use of the simulations from the resulting posterior density, inferences for the vector of parameters are derived. Metropolis-Hastings algorithms often provide effective methods for simulating from intractable posterior densities. However, the success of these methods depends on the choice of reasonable proposal densities which in certain cases, can be much difficult. In some situations, replacing generic proposal densities in Metropolis-Hastings algorithms by the conditional distribution of the parameter component that is to be sampled, proves a better MCMC method. This very strategy of MCMC algorithms are known as Gibbs samplers [10].

Gibbs sampler

The basic idea of Gibbs sampling is to partition the set of unknown parameters and then estimate them one at a time, with each parameter or group of parameters estimated conditional on all the others. Suppose that the parameter vector θ is divided into q components or subvectors, $\theta = (\theta_1, \theta_2, \ldots, \theta_q)$. At each iteration t, an ordering of the d components of θ is chosen, and each θ_j^t is sampled from the conditional disitribution given all other components of θ:

$$p(\theta_j | \theta_{-j}^{t-1}, y)$$

where, θ_{-j}^{t-1} represents all subvectors of θ other than θ_j at their current values:

$$\theta_{-j}^{t-1} = (\theta_1^t, \ldots, \theta_{j-1}^t, \theta_{j+1}^{t-1}, \ldots, \theta_d^{t-1})$$

Thus, each component of θ is updated conditional on the current values of the other components of θ.

R2jags

JAGS is an acronym for Just Another Gibbs Sampler. It is a program developed by statisticians that allows the user to fit various Bayesian models including the complicated ones. It analyses Bayesian models using Markov Chain Monte Carlo (MCMC) simulations and is licensed under GNU General Public License version 2. In order to work closely with the R language, another package R2jags has been developed which runs JAGS via R making the posterior analysis comparatively easier. Thus, our main tool for fitting multilevel models is R2jags that can be called from R with the help of the function `jags()`. The arguments of this function are:

```
jags(data, inits, parameters.to.save, model.file="model.bug",
n.chains=3, n.iter=2000, n.burnin=floor(n.iter/2), n.thin=max(1, floor((n.iter -
n.burnin) / 1000)),DIC=TRUE, working.directory=NULL,
jags.seed = 123,refresh = n.iter/50, progress.bar = "text", digits=5,
RNGname = c("Wichmann-Hill", "Marsaglia-Multicarry","Super-Duper",
"Mersenne-Twister"))
```

where `data` is either a named list of the data for the specified model, `inits` is a list, each element of which is itself a list of initial values for the parameters that are to be estimated or it can be function for creating initial values (possibly random), `parameters.to.save` stands for the character vector of the names of the parameters to be saved and monitored, `model.file` specifies the file that contains the model written in BUGS code. It may either have '.bug' or '.txt'extension, `n.chains` accepts an integer that determines the number of Markov chains to be run and defaults to 3, `n.iter` indicates the number of total iterations per chain (including burn in samples, default is 2000). Remaining details can be seen from the pdf manual available with the R2jags package [11].
The software package R2jags is an interface between R and JAGS. Data is created in R, simulation is done in JAGS and finally output is reported with R.

3.1.1 Fitting a Varying-intercept, varying-slope model

Setting up the data in R

We start by loading in the observed degradation measurements y_{ij} reported in Table 1 for i^{th} unit at j^{th} time for all the batches. These 24 batches are assumed to be a random sample from a large population of batches. Data is entered as a vector *y*, each element of which represents an individual measurements in each of the 24 batches at a particular time *t*. `Time` is a variable for different time periods. The batches are entered as a categorical vector `batch`. The total number of batches are specified by *J* whereas *n* stands for the number of degradation measurements. Data is created with the following commands.

```
batch<-rep(paste("batch",1:24,sep=""),4)
batch<-as.integer(factor(batch))
Time<-c(rep(0,24),rep(12,24),rep(24,24),rep(36,24))
y<-c(99.9,101.1,100.3,100.8,100,100.1,99.6,100.4,100.9,100.5,101.1,
100.9,99.8,100.1,100.7,100.3,100.2,99.8,100.8,100,99.6,100.2,99.8,
100,98.9,97.1,98.3,96.8,98,98.1,98.6,99.4,98.9,99.5,98.1,97.9,98.8,
99.1,98.7,98.3,98.2,97.8,98.8,98,99.6,98.2,97.8,99,95.9,94.1,95.3,
94.8,96,98.1,96.6,96.4,96.9,94.5,93.1,95.9,93.8,93.1,93.7,96.3,97.2,
95.8,95.8,96,92.6,97.2,95.8,95,92.9,91.1,92.3,90.8,92,95.1,92.6,95.4,
96.9,93.5,91.1,93.9,89.8,90.1,91.7,93.3,94.2,90.8,94.8,92,88.6,94.2,
90.8,92)
n<-length(y)
J<-length(unique(batch))
```

Formulation of multilevel regression model

The next step in multilevel modeling is to allow more than one regression coefficient to vary by batch. We commence with a varying-intercept, varying-slope model including variable `Time` as unit-level predictor. Thus, our data-level model is,

$$y_i \sim \mathrm{N}(\alpha_{j[i]} + \beta_{j[i]}\mathrm{Time}_i, \sigma_y^2)$$

Each y_i contributes towards the likelihood function through normal probability density. We further assume that y_i's are conditionally independent. Thus, the overall likelihood function is the product of these contributions. The measurement error term e_i's are considered to be independently and identically normally distributed with mean zero and unknown variance σ_y^2 . Although JAGS permits composite expressions in its distributional specifications yet, for clarity, we split the model into two parts

$$y_i \sim \mathrm{N}(\hat{y}_i, \sigma_y^2)$$

$$\hat{y}_i = \alpha_{j[i]} + \beta_{j[i]}\mathrm{Time}_i$$

The codes defining the model are fractured into subparts for step-wise illustrations and have been reassembled at the end.

Codes defining the likelihood function or data-level model

```
cat(" model {
  for (i in 1:n){
    y[i]~dnorm(y.hat[i], tau.y)
    y.hat[i]<-a[batch[i]] + b[batch[i]]*Time[i] }
```

The specification of normal distribution in R2jags is made with the inverse-variance parameter $(\tau = 1/\sigma^2)$ instead of the usual variance parameter. The function `cat` behaves pretty much like `paste` with the exception that the result is not a character object and the codes are directly written to a file we specify.

The group-level model or prior distributions

Here, we model the random intercepts α_j and slopes β_j from the same population with multivariate normal distribution with mean vector μ and variance-covariance matrix Σ. Ignoring the correlation between these random coefficients could

be quite impractical, so we allowed for the presence of correlation ρ between them.

$(\alpha_j, \beta_j) \sim \text{MVN}(\mu, \Sigma)$ # Bivariate normal random effects

where, $\mu = (\mu_\alpha, \mu_\beta)^T$ # Mean vector

$\Sigma = \begin{bmatrix} \sigma_\alpha^2 & \sigma_{\alpha\beta}^2 \\ \sigma_{\alpha\beta}^2 & \sigma_\beta^2 \end{bmatrix}$ # Variance-covariance matrix

The diagonal elements of var-cov matrix Σ are variances of intercepts α_j and slopes β_j respectively whereas the off-diagonal elements are the covariance between them.

```
for (j in 1:J) {
   a[j]<-B[j,1]
   b[j]<-B[j,2]
   B[j,1:2]~dmnorm(B.hat[j,], Tau.B[,])
   B.hat[j,1]<-mu.a
   B.hat[j,2]<-mu.b
}
```

We have used uppercase letters for matrix parameters while lowercase letters for scalars and vectors. We are still left with two quantities `tau.y` and `rho` which must be assigned priors before moving ahead. The inverse-variance τ_y is being defined deterministically in terms of standard deviation parameter σ_y, which is then given a probability distribution. Correlation parameter ρ is given uniform prior distribution in the range $(-1, 1)$.

```
tau.y<- pow(sigma.y, -2)
sigma.y ~ dunif (0, 100)
rho ~ dunif(-1,1)
```

Hyperprior distributions

The probability distribution associated with the parameter of the prior distribution is known as hyperprior distribution. Since, we have assigned an informative bivariate normal prior to random intercepts and slopes, we have two hyperparameters in the matrix form. We follow a common practice and use weak-informative hyperprior distributions for the parameters of bivariate normal prior. These are specified outside the group-level model.

```
mu.a ~dnorm(0,0.0001)
mu.b ~dnorm(0,0.0001)
```

Thus, the terms μ_α and μ_β are each given univariate normal prior distributions with mean 0 and standard deviation 100. This roughly means that we are expecting these coefficients to lie in the range $(-100, 100)$. If the estimates are in this range, the prior distribution is contributing almost negligibly in the inference. We define the variance-covariance matrix Σ of bivariate normal random effects as inverse of `Tau.B` i.e. `Sigma.B)` in the JAGS code. Non-informative hyper priors for σ_α^2 and σ_β^2 have been coded as following:

```
    Tau.B[1:2,1:2]<-inverse(Sigma.B[,])
    Sigma.B[1,1]<-pow(sigma.a,2)
    sigma.a ~ dunif(0, 100) # SD of intercepts
    Sigma.B[2,2]<-pow(sigma.b,2)
    sigma.b~ dunif(0,100)   # SD of slopes
```

The off-diagonal elements $\sigma_{\alpha\beta}^2$ are defined in terms of correlation parameter ρ

```
    Sigma.B[1,2]<-rho*sigma.a*sigma.b
    Sigma.B[2,1]<-Sigma.B[1,2]
  }", file="degradation.txt")
```

The above model codes can be reassembled in the following way so as to make it work.

```
cat(" model {
  for (i in 1:n){
    y[i]~dnorm(y.hat[i], tau.y)
    y.hat[i]<-a[batch[i]] + b[batch[i]]*Time[i] }
    for (j in 1:J) {
   a[j]<-B[j,1]
   b[j]<-B[j,2]
   B[j,1:2]~dmnorm(B.hat[j,], Tau.B[,])
   B.hat[j,1]<-mu.a
   B.hat[j,2]<-mu.b
}
tau.y<- pow(sigma.y, -2)
sigma.y ~ dunif (0, 100)
rho ~ dunif(-1,1)
mu.a ~dnorm(0,0.0001)
mu.b ~dnorm(0,0.0001)
Tau.B[1:2,1:2]<-inverse(Sigma.B[,])
    Sigma.B[1,1]<-pow(sigma.a,2)
    sigma.a ~ dunif(0, 100)
    Sigma.B[2,2]<-pow(sigma.b,2)
    sigma.b~ dunif(0,100)
     Sigma.B[1,2]<-rho*sigma.a*sigma.b
    Sigma.B[2,1]<-Sigma.B[1,2]
  }", file="degradation.txt")
```

Data, initial values, and parameters

We have earlier defined all the data variables and finally we unite them in a listed form as per the requirement of R2jags.

```
drug.data<-list("n","y","J","Time","batch")
```

Supplying initial values for all the model parameters that are to be simulated, is the next important task to be undertaken. The function `jags` accepts a listed data object of initial values (preferably random initial values using random-number generators) for the parameters . If the initial values are not supplied, `jags` generates them itself. However, it often crashes when using self-generated initial values.

```
inits<-function() {list(mu.a=rnorm(1,100,1),mu.b=rnorm(1),
                        sigma.y=runif(1,0,100),sigma.a=runif(1),
                        sigma.b=runif (1),rho=runif(1,-1,1))}
```

Next, we specify the names of the parameters that we want to save from the JAGS run within the vector `params`

```
params<-c("a","b","mu.a","mu.b","sigma.a","sigma.b", "sigma.y",
          "rho")
```

Calling JAGS from R

After setting up the codes in jags, we finally run the model via the function`jags`. We assess convergence by checking if the distributions of the different simulated chains mix; thus atleast 2 chains must be simulated. We preferred to simulate 3 chains.

```
output<-jags(drug.data,inits,params,model.file="degradation.txt",
 n.iter=12000, n.chains=3)
```

Summarizing the output

The results are printed with the function `print`, which prints detailed summary of results and it is not possible to show here. However, its relevant parts are summarized in Table 2.

Table 2: Posterior summary of model parameters

Parameters	Mean	sd	Quantiles					Rhat
			0.025	0.250	0.50	0.75	0.975	
α_1	100.576	0.191	100.155	100.472	100.570	100.697	100.947	1.010
α_2	100.595	0.196	100.188	100.483	100.585	100.717	100.982	1.011
α_3	100.663	0.233	100.212	100.518	100.638	100.789	101.170	1.051
α_4	100.577	0.199	100.152	100.471	100.565	100.694	100.972	1.011
α_5	100.660	0.240	100.178	100.510	100.638	100.797	101.195	1.050
α_6	100.670	0.247	100.200	100.513	100.650	100.803	101.219	1.051
α_7	100.638	0.211	100.207	100.508	100.621	100.769	101.080	1.039
α_8	100.571	0.201	100.131	100.470	100.565	100.696	100.960	1.009
α_9	100.538	0.205	100.095	100.429	100.545	100.669	100.920	1.008
α_{10}	100.614	0.208	100.167	100.494	100.605	100.738	101.046	1.035
α_{11}	100.559	0.204	100.117	100.453	100.554	100.684	100.958	1.009
α_{12}	100.651	0.233	100.183	100.507	100.628	100.782	101.170	1.057
α_{13}	100.593	0.195	100.161	100.480	100.585	100.721	100.967	1.017
α_{14}	100.703	0.285	100.182	100.517	100.667	100.859	101.368	1.084
α_{15}	100.537	0.208	100.083	100.426	100.541	100.671	100.927	1.008
α_{16}	100.616	0.209	100.177	100.494	100.605	100.743	101.039	1.031
α_{17}	100.608	0.199	100.181	100.492	100.596	100.726	101.018	1.023
α_{18}	100.605	0.198	100.189	100.492	100.595	100.728	101.008	1.027
α_{19}	100.642	0.219	100.202	100.506	100.621	100.773	101.111	1.038
α_{20}	100.590	0.196	100.141	100.485	100.584	100.710	100.985	1.018
α_{21}	100.504	0.233	99.980	100.392	100.515	100.650	100.905	1.014
α_{22}	100.570	0.194	100.148	100.465	100.564	100.693	100.947	1.012
α_{23}	100.532	0.218	100.050	100.413	100.538	100.669	100.949	1.007
α_{24}	100.500	0.250	99.951	100.368	100.508	100.653	100.966	1.011
β_1	-0.205	0.019	-0.243	-0.217	-0.205	-0.193	-0.167	1.001
β_2	-0.208	0.019	-0.246	-0.221	-0.208	-0.195	-0.171	1.003
β_3	-0.269	0.020	-0.310	-0.282	-0.269	-0.256	-0.231	1.012
β_4	-0.194	0.020	-0.232	-0.207	-0.194	-0.182	-0.156	1.002
β_5	-0.278	0.020	-0.317	-0.291	-0.278	-0.265	-0.241	1.006
β_6	-0.280	0.020	-0.323	-0.292	-0.279	-0.266	-0.241	1.012
β_7	-0.250	0.019	-0.288	-0.262	-0.249	-0.237	-0.213	1.005
β_8	-0.197	0.020	-0.236	-0.210	-0.198	-0.185	-0.158	1.001
β_9	-0.173	0.020	-0.211	-0.187	-0.173	-0.161	-0.134	1.002
β_{10}	-0.245	0.019	-0.282	-0.258	-0.245	-0.232	-0.207	1.007
β_{11}	-0.177	0.019	-0.215	-0.190	-0.176	-0.163	-0.139	1.002
β_{12}	-0.264	0.020	-0.305	-0.276	-0.263	-0.250	-0.226	1.016
β_{13}	-0.223	0.019	-0.261	-0.236	-0.223	-0.210	-0.185	1.006
β_{14}	-0.307	0.020	-0.347	-0.321	-0.307	-0.294	-0.268	1.022
β_{15}	-0.173	0.019	-0.212	-0.186	-0.174	-0.160	-0.135	1.002
β_{16}	-0.245	0.020	-0.283	-0.258	-0.245	-0.232	-0.208	1.002
β_{17}	-0.229	0.019	-0.267	-0.242	-0.230	-0.216	-0.191	1.002
β_{18}	-0.225	0.019	-0.262	-0.238	-0.225	-0.212	-0.188	1.003
β_{19}	-0.262	0.019	-0.300	-0.275	-0.262	-0.249	-0.225	1.007
β_{20}	-0.223	0.019	-0.261	-0.236	-0.223	-0.210	-0.186	1.001
β_{21}	-0.149	0.020	-0.187	-0.163	-0.150	-0.136	-0.109	1.001
β_{22}	-0.203	0.019	-0.241	-0.216	-0.203	-0.190	-0.165	1.003
β_{23}	-0.157	0.020	-0.194	-0.170	-0.157	-0.144	-0.119	1.001
β_{24}	-0.130	0.020	-0.170	-0.143	-0.130	-0.116	-0.090	1.003
bpvalue	0.535	0.499	0.000	0.000	1.000	1.000	1.000	1.001
ρ	-0.318	0.548	-0.976	-0.787	-0.469	0.069	0.882	1.016
μ_α	100.597	0.147	100.314	100.505	100.587	100.689	100.887	1.038
μ_β	-0.220	0.013	-0.245	-0.228	-0.220	-0.211	-0.196	1.012
σ_α	0.150	0.117	0.004	0.058	0.123	0.216	0.423	1.033

σ_β	0.052	0.010	0.036	0.045	0.051	0.058	0.076	1.009
σ_y	0.826	0.072	0.698	0.775	0.822	0.872	0.984	1.001
deviance	233.834	9.080	218.750	227.351	232.939	239.209	253.788	1.001

DIC info (using the rule, p_D = var(deviance)/2)
$pD = 41.2$ $DIC = 275.1$
DIC is an estimate of expected predictive error (lower deviance is better)

Table 2 summarizes the inference for the model parameters on the basis of 6000 iterations out of 12000. The convergence statistic, $\hat{R}$ in the last column is approximately the square root of the variance of the mixture of all the chains, divided by the average within-chain variance. $\hat{R}$ being less than equal to 1.1 indicates that the chains have mixed well. The intercept α_1 has a posterior mean of 100.576 and a standard standard deviation of 0.191. It has a posterior median of 100.57 with a 95% credible interval of $[100.155, 100.947]$. While, for α_{24}, it is in the range $[99.951, 100.966]$. For the `Time` coefficient β_5, posterior median is calculated to be -0.278 with 50% credible interval of $[-0.291, -0.265]$. In this model, the unexplained within-batch variation has posterior standard deviation of $\hat{\sigma}_y = 0.826$; the posterior standard deviation of the batch intercepts is $\hat{\sigma}_\alpha = 0.15$ and that of batch slopes is $\hat{\sigma}_\beta = 0.052$. The correlation ρ between intercepts and slopes is -0.318. The estimated effective number of parameters, p_D has been reported immediately after the output table. The deviance for this model is 233.834.

3.1.2 Assessment of goodness of fit: A posterior predictive check

Checking the adequacy of the fit of the model is crucial to statistical analyses. The basic technique for assessing the goodness of fit of the model to the data is to compare the simulated values drawn from the posterior predictive distribution of the replicated data y^{rep} with the observed data y. If the model fits, then y^{rep} is similar to y. Any systematic differences between the simulations and the data indicates a lack of fit of the model. In order to evaluate the fit of the posterior distribution of the Bayesian model, we compute Bayesian p-value. It quantifies the proportion of times that the replicated data could be more extreme than the observed data as measured by the test quantity:

$$p_B = Pr(T(y^{\text{rep}}, \theta) \geq T(y, \theta)|\theta)$$

where, the probability is taken over the posterior distribution of θ and the posterior predictive distribution of y^{rep}:

$$p_B = \iint I_{T(y^{\text{rep}},\theta)\geq T(y,\theta)} p(y^{\text{rep}}|\theta) p(\theta|y)\, \mathrm{d}y^{\text{rep}}\, \mathrm{d}\theta,$$

where, the property $p(y^{\theta rep}|\theta, y) = p(y^{\text{rep}}|\theta)$ has been considered and I is the indicator function.

Generally, this complex computation is handled through S simulations. Let we have simulations from posterior density of θ. For each simulated θ-value, draw one y^{rep} from the predictive distribution. This leads to the joint posterior distribution, $p(y^{\text{rep}}, \theta|y)$. The estimated p-value is just the proportion of these S simulations for which the test quantity equals or exceeds its realized value, i.e

$$T(y^{\text{reps}}, \theta^s) \geq T(y, \theta^s), \qquad s = 1, \ldots\ldots S$$

In R2jags, it is very simple to evaluate Bayesian p-value. We have used sum-squares type discrepancy measure and computed sum of squared residuals (SSR) for actual and observed data set. The `step` function is used to test whether the new data set is more extreme. The mean of resulting logical vector leads to the required Bayesian p-value. All these commands must be encoded within the model and the object "bpvalue" must be saved in the vector `params` in order to get its value in the output. The p-value near 0.5 indicates a good fit of the model to the data while values close to 0 or 1 suggests a doubtful fit.

```
for (i in 1:n) {

residual[i]<-y[i]-y.hat[i]                  # Residuals for observed
                                            #  data
predicted[i]<-y.hat[i]                      # Predicted values
```

```
sq[i]<-pow(residual[i], 2)

y.new[i]~dnorm(y.hat[i], tau.y)              # one new data set at each
                                             # MCMC iteration
sq.new[i]<-pow(y.new[i]-predicted[i], 2)     # Squared residuals for
                                             # new data
}

fit<-sum(sq[])                               # Sum of squared residuals
                                             # for actual data set
fit.new<-sum(sq.new[])                       # Sum of squared residuals
                                             # for new dataset
test<-step(fit.new - fit)                    # Test whether new data set
                                             # more extreme
bpvalue<-mean(test)                          # Bayesian p-value
```

Also, we carried out the posterior predictive check graphically. A plot of lack of fit for the replicated data versus the lack of fit for the observed data is made and has been presented in Figure 2. If the model fits the data, then half of the points in the plot will lie above and half of them below a 1 : 1 line.

```
out<-output$BUGSoutput
plot(out$sims.list$fit,out$sims.list$fit.new)
abline(0,1)
```

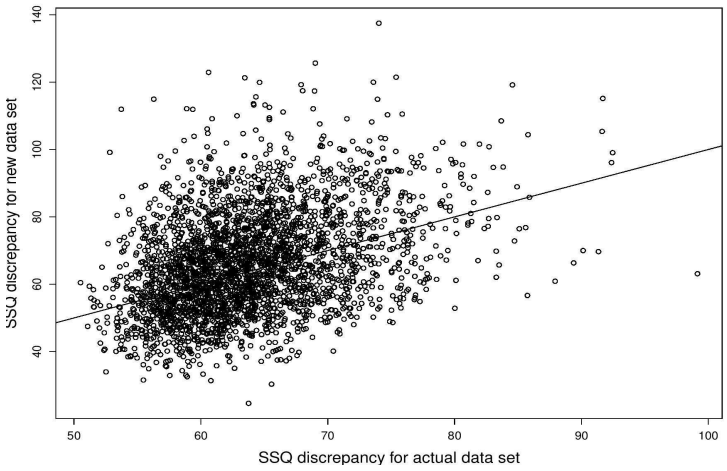

Fig. 2: Graphical posterior predictive check of the model adequacy for the degradation data analysis plotting predictive vs. realized sum of squares discrepancies.

3.2 Analysis of the data using optimization tool

The lme4 package of R has a function `lmer` which is an acronym for linear mixed-effects models with R, to fit linear mixed-effects models and implements Laplace approximation. The rationale for using it as a Bayesian tool is that the computational methods implemented in this function treats the parameter as random which is contrary to classical set up. In addition to this, it purveys the restricted maximum likelihood estimation (see [12] and [13]) where instead of working on original data vector, a linear combination of observations is chosen to define the likelihood function such that it is invariant to the values of fixed effect parameters. Integrating out the likelihood function against the parameters itself

suggests that paradigm is Bayesian. We have employed `lmer` to approximate the target density. The arguments of the function are:

```
lmer(formula, data = NULL, REML = TRUE, control = lmerControl(), start = NULL,
verbose = 0L, subset, weights, na.action, offset, contrasts = NULL, devFunOnly =
FALSE, ...)
```

Here, we define some of its arguments in order to save the space. Rest of the details can be obtained from its pdf manual available with the lme4 package [14]. The argument `formula` stands for a two-sided formula object describing both random and fixed-effects part of the model. The reponse variable is placed at left of a $\sim$ operator while the input variables are on its right side, separated by a + operator. Random-effects terms are distinguished by vertical bars ("—") separating expressions for design matrices from grouping factors, `data` refers to an optional dataframe containing the variables named in the formula. If not supplied, it takes the variables from the environment from which `lmer` is called. The argument `REML` specifies the logical scalar. When TRUE (default), the estimates are chosen to maximize the REML criterion.

3.2.1 Fitting a varying-intercept and varying-slope model using R

Setting up the data

The model is the same, defined earlier. Since, we have considered normal probability density for each potency measurement, the function `lmer` would be most appropriate for the analysis. Batch is the categorical variable. Therefore, it is entered into the workspace as a factor vector.

```
batch<-rep(paste("batch",1:24,sep=""),4)
batch<-factor(batch)
Time<-c(rep(0,24),rep(12,24),rep(24,24),rep(36,24))
y<-c(99.9,101.1,100.3,100.8,100,100.1,99.6,100.4,100.9,100.5,101.1,
100.9,99.8,100.1,100.7,100.3,100.2,99.8,100.8,100,99.6,100.2,99.8,
100,98.9,97.1,98.3,96.8,98,98.1,98.6,99.4,98.9,99.5,98.1,97.9,98.8,
99.1,98.7,98.3,98.2,97.8,98.8,98,99.6,98.2,97.8,99,95.9,94.1,95.3,
94.8,96,98.1,96.6,96.4,96.9,94.5,93.1,95.9,93.8,93.1,93.7,96.3,97.2,
95.8,95.8,96,92.6,97.2,95.8,95,92.9,91.1,92.3,90.8,92,95.1,92.6,95.4,
96.9,93.5,91.1,93.9,89.8,90.1,91.7,93.3,94.2,90.8,94.8,92,88.6,94.2,
90.8,92)
```

Analysis with lmer

We chose to adopt the REML estimation criterion. With all the data variables defined, we finally fit the model as

```
M1<-lmer(y~Time+(Time|batch))
```

Summarizing the output

The function `display` available in the package arm prints all the relevant posterior quantities for the purpose of inference.

Table 3: Parameter estimates

	coef.est	coef.se
(Intercept)	100.61	0.14
Time	-0.22	0.01

Table 4: Error terms

Groups	Name	Std. Dev.	Correlation
batch	(Intercept)	0.17	
	Time	0.05	-1.00
Residual		0.80	

number of obs: 96, groups: batch, 24
AIC = 299, DIC = 268
deviance = 277.5

4 Juxtaposition of exact and approximate Bayesian Analysis of the data

The correlation between the random intercepts and slopes resulted by `lmer` is perfect negative correlation which is much impractical, rather, an ideal situation. But, with the JAGS model, we have got a reasonable negative correlation. Further, the deviance of the JAGS model is 232.9 which is much lesser in comparison to what is yielded by the function `lmer` i.e. 277.5. Thus, it can be seen that the Bayesian analysis of the degradation model using Gibbs sampling procedure gives a better solution as compared to the analysis based on approximations.
It can be found that the intercepts are not varying much between the batches. Therefore, it is the slope which would decide the rate of degradation mechanism within various batches. The posterior medians of slopes obtained by the two methods have been presented parallel to each other in Table 5.

Table 5: Comparison of slopes resulted by `lmer` and `jags` for all the 24 batches

	Estimates of slopes by			Estimates of slopes by	
Batch	lmer	jags	Batch	lmer	jags
1	-0.20394	-0.205	13	-0.22350	-0.223
2	-0.20725	-0.208	14	-0.31489	-0.307
3	-0.27413	-0.269	15	-0.16968	-0.173
4	-0.19254	-0.194	16	-0.24795	-0.245
5	-0.28387	-0.278	17	-0.23006	-0.229
6	-0.28553	-0.28	18	-0.22515	-0.225
7	-0.25292	-0.25	19	-0.26591	-0.262
8	-0.19579	-0.197	20	-0.22350	-0.223
9	-0.16968	-0.173	21	-0.14357	-0.149
10	-0.24795	-0.245	22	-0.20229	-0.203
11	-0.17299	-0.177	23	-0.15178	-0.157
12	-0.26757	-0.264	24	-0.12242	-0.13

From Table 5 it can be found that batch 24 of the drug is undergoing minimum degradation with a slope of -0.13 whereas the maximum degradation is occuring in batch 14 with the slope value as -0.307.

5 Conclusions and final remarks

In this paper, we introduce multilevel modeling in a practical degradation data problem. The analysis has been presented in Bayesian paradigm using both simulation technique via function `jags` and has been juxtaposed with the approximate results evaluated in R via `lmer`. The work presented in this paper is not mathematical but rather conceptual and computational. Although the approximate solution given by `lmer` and the exact solution yielded by `R2jags` has come out to be much closer for the parameters, yet, the correlation could not be estimated appropriately by `lmer`. Moreover, with `lmer` , much programming effort is required to simulate predictions and predicted data. Also, with the small number of groups, the precise estimation of variance components with `lmer` might not be possible in the absence of enough information. Whereas, in addition to the routine posterior quantities and quantiles, JAGS allows predictions of new units in the same framework. The modular form of JAGS allows the programmers to bring together all sorts of Bayesian models such as linear and generalized linear model and thus, facilitates much complicated modeling.

Thus, Bayesian methods using simulation technique seems to be a reasonable choice especially for more complicated degradation model. Further, working with simulations rather than simply getting point estimates of parameters, the user can directly capture inferential uncertainty and incorporate it into prediction of future samples.

References

[1] S. T. Tseng, M. Hamada & C. H. Chiao, Using Degradation Data to Improve Fluorescent Lamp Reliability, Journal of Quality Technology, 27, 363-369 (1995).
[2] K. A. Doksum, Degradation Rate Models for Failure Time and Survival Data, CWI Quart, 4, 195-203 (1991).
[3] G. A. Whitmore & F. Shenkelberg, Modelling accelerated degradation data using Wiener diffusion with a time scale transformation, Lifetime Data Analysis, 3, 27-45 (1997).
[4] C. J. Lu & W. Q. Meeker, Using Degradation Measures to Estimate a Time-to-Failure Distribution, Technometrics, 35, 161-174 (1993).
[5] J. C. Pinheiro, & D. C. Bates, Approximations to the log-likelihood function in the nonlinear mixed-effects model, Journal of Computational and Graphical Statistics,1(4), 12-35 (1995).
[6] R Development Core Team, R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2012).
[7] S. C. Chow, J. Shao, Estimating drug self-life with random batches, Biometrics, 47:753-763 (1991).
[8] M. S. Hamada et. al. , Bayesian Reliability. Springer Science+Business Media, LLC, 233 Spring Street, New York 10013, USA (2008).
[9] A. Gelman and J. Hill, Data Analysis Using Regression and MultilevelHierarchical Models, Cambridge University Press, New York (2007).
[10] A.F.M. Smith and G.O. Roberts, Bayesian Computation via the Gibbs Sampler and Related Markov Chain Monte Carlo Methods, J. R. Statist. Soc. Ser. B, 55, 3-23 (1993).
[11] Yu-Sung Su, Masanao Yajima, R2jags: A Package for Running jags from R, Version 0.03-11 (2013).
[12] H. D. Patterson, R. Thompson, Recovery of inter-block information when block sizes are unequal, Biometrika, 58 (3), 545-554 (1971).
[13] D. A. Harville, Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems, Journal of the American Statistical Association 72 (358): 320-338 (1977).
[14] Bates et. al., Linear mixed-effects models using Eigen and S4, R package version 1.1-7, URL http://lme4.r-forge.r-project.org/ project.org/web/p (2014).