Appendix F

Population Variability Modeling

Population variability modeling solves the problem of how best to estimate facility-specific risk-model parameters, given that facility-specific data are limited but industry data are more plentiful, albeit most likely inhomogeneous. From a broader perspective, it is also of interest to understand just how variable certain parameters are, and the values over which those parameters can reasonably be expected to range, in a given population of facilities.

Population variability modeling was introduced by Kaplan [F-1] in the 1980s and applied to modeling the frequency of loss of offsite power (LOOP) at nuclear power plants. Most plants lose power very seldom, but the fleet as a whole has several such events per year; the rate varies quite significantly from one plant to another, and plant risk is sensitive to this parameter, so we do not wish to use a generic value derived by pooling all the losses and dividing by the total exposure time. In Kaplan's original work, the population variability distribution (PVD) portrayed the relative fraction of plants having a given LOOP frequency. In plant-specific analysis, this PVD was then used as a prior distribution for LOOP frequency, and updated with plant-specific data, the resulting posterior being then used in probabilistic risk assessment (PRA) as the state-of-knowledge distribution of LOOP frequency for that plant. Note the assumptions being used here: that it makes sense to draw a PVD in the first place for the population that we are trying to work with, that the facility we are interested in can be viewed as a member of this population, recognizing that it is characterized by variation in LOOP frequency, and so on. If we accept the basic ideas, then in using this framework, we end up imputing to our plant a distribution for this parameter that reflects our experience as well as the experience of the operating fleet, in both the central tendency of that parameter for our plant and the epistemic distribution of likely values of that parameter for our plant. If we have very little plant-specific data, our state-of-knowledge curve will look like the PVD curve describing the whole plant population.

Therefore, when presented with a set of data of component performance, the most pressing question that arises is can we pool the data, and if not, can we find a curve that fits the variability of the data sufficiently to quantify a prior distribution that adequately represents the performance of the component population.

F-1. POPULATION ANALYSIS STEPS

Figure F-1 represents the basic steps used for population analysis.

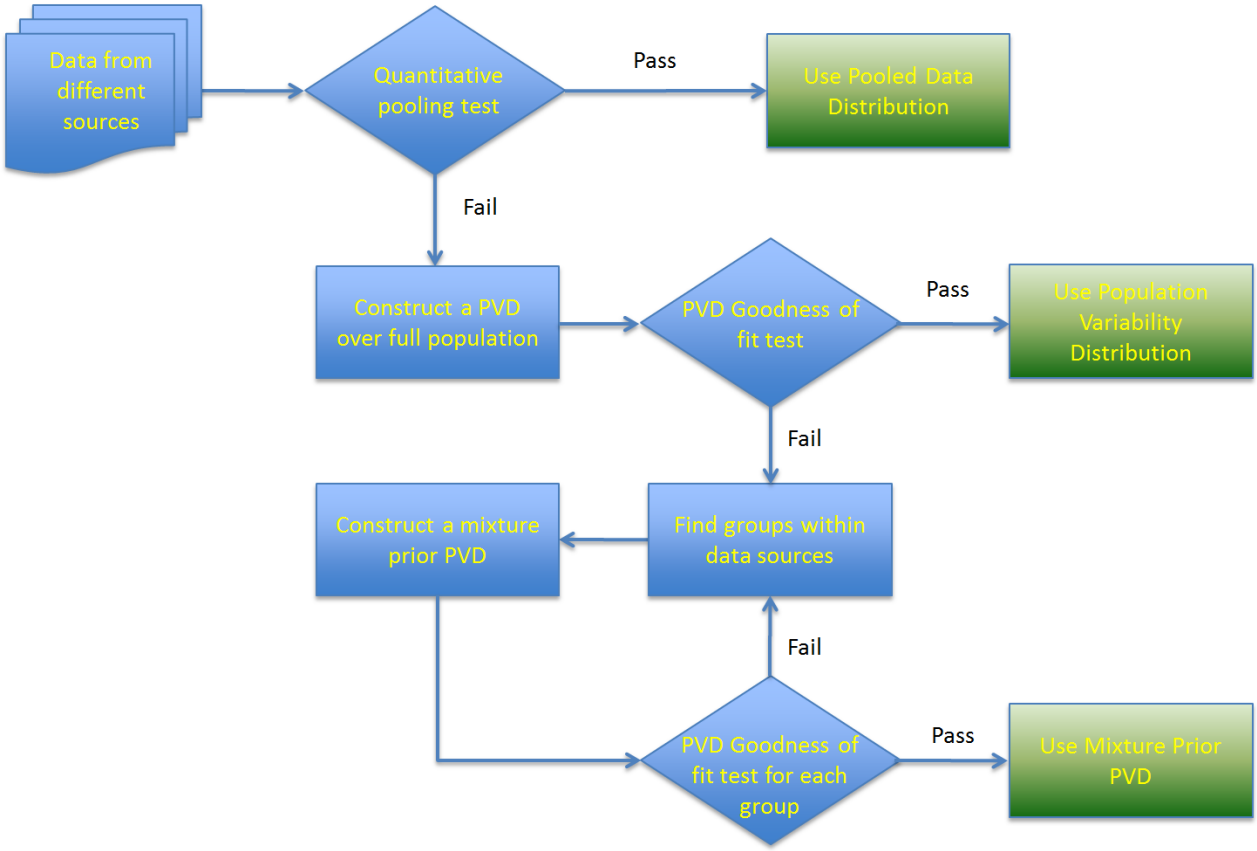


Figure F-1. Population analysis steps.

It is always best to use a pooled data set if the data set fits a pooled goodness-of-fit test. The first analysis is to take the full population and perform a quantitative pooling test. If the goodness-of-fit test passes for the distribution inferred by the pooled data, then the prior fitting the population is found and can be used to update with the facility component data.

If the pooled data test fails, then a PVD is constructed for the entire population. Hierarchical Bayes is used in this analysis, with the prior parameters used as the first hierarchy (hyperpriors). These hyperprior parameters are started as flat or completely diffuse, ideally chosen to spread out along the entire realm of possibility. Generally, a Markov Chain Monte Carlo (MCMC) program is used with initial guesses for the starting points of the parameters to run the inference until the parameters converge to values that are used as the parameters in the PVD. The parameters may not converge to an answer depending on the degree of variability within the full population data set. If the PVD parameters converge to values, then the results for the PVD inference are used to compare to replicated values for the individual data sets in the same manner as for the pooling test and a goodness-of-fit score is produced. If the goodness-of-fit test passes for this PVD, then the prior fitting the population is found and this PVD can be used to update with the facility component data.

If the full population PVD test fails to find converged values or fails the goodness-of-fit test, then a data source grouping analysis must be performed. This consists of both qualitative and a quantitative analysis. In a qualitative manner, as much information about the data set is determined as possible beyond just the number of failures over time or demands. If data sources are known to be from one manufacturer of component, for instance, that is noted. The same holds for any other pertinent information such as environment used, application, etc. The quantitative part of the analysis is to use a machine learning data mining algorithm to cluster the data into groups by these attributes identified in the qualitative analysis.

The cluster analysis can work on as little as two points of information (covariates) such as failures and time or failures and demands, however, the more covariates that can be provided, the more information can be gleaned from the cluster analysis.

Using the clusters obtained from the grouping analysis an individual PVD analysis is performed on each cluster. If a cluster cannot be fitted with a PVD, then the grouping process can be re-performed with different parameters for the algorithm or by changing the number of covariates (increase or decrease). If clusters are marked as outliers and a PVD cannot be fitted, then a simple Bayesian update on the individual data sources with a Jeffrey's prior will suffice. Once all clusters have a PVD, then they are weighted using a mixture prior for use in updating facility component data. The weights are equivalent to their percentage of the population.

F-1.1 Markov Chain Monte Carlo Sampling

The use of Markov Chain Monte Carlo (MCMC) programs greatly simplifies the calculation of the problems encountered in population variability analysis. Bayesian inference typically involves several integrals in the denominator of the equation and MCMC avoids the need to empirically solve the multidimensional integral. The basic process of MCMC uses a random number to sample directly from the posterior distribution. It then has one "answer." Another random sample is taken, and another, and so forth until an entire set of samples can be used to determine a numerical distribution that represents the posterior distribution.

The basic premise of a Markov chain is that it is constructed such that the chain converges to a joint posterior distribution. The chain uses a sequence of random variables X_0, X_1, X_2, … to sample and create the posterior distribution. The distribution of X_{n+1} only depends on X_n, which is a property of Markov chains. The chain "forgets" its initial state and the next sample builds on the resulting distribution of the prior sample. The Markov function, $f(x_{n+1}|x_n)$, is known as a "transition kernel." Once the distribution is stable from sample to sample (known as "stationary"), samples can be taken to estimate the parameters of interest. Various methods exist to construct the transition kernel. Gibbs sampling, Slice Sampling, and Metropolis-Hastings are a few of the most popular.

F-1.1.1 OpenBUGS and JAGS

OpenBUGS (Open-source Bayesian Updating Using Gibbs Sampling) and JAGS (Just Another Gibbs Sampler), two Bayesian inference MCMC programs, are vetted open-source programs that are good to use for these types of problems. Both programs use the BUGS language and are nearly identical. Any MCMC program capable of Bayesian inference can be used, however, these programs were used for the sample analyses presented here.

The publicly available NASA publication NASA/SP-2009-569, [F-2] Appendix C, provides a tutorial in the basic use of OpenBUGS.

F-1.2 Pooled Data Analysis

The pooled data test uses a non-informative prior to infer parameters over the entire data set. Each data source is duplicated using the pooled data distribution parameters, and compared to the results obtained by updating individually with the non-informed prior. A goodness-of-fit such as a Chi-squared test is applied to determine if the replicated data matches the data using the pooled data distribution.

The Bayesian update rule for a single degree of freedom can be expressed as:

$$\pi_1(\theta|E) = \frac{L(E|\theta)\pi_0(\theta)}{\int L(E|\theta)\pi_0(\theta)d\theta} \quad \text{(F-1)}$$

where:

E is the evidence

θ is the parameter of interest

$\pi_0(\theta)$ is the prior distribution

$L(E|\theta)$ is the likelihood function

$\pi_1(\theta|E)$ is the posterior distribution (the updated estimate).

The parameter of interest in a pooled data update is the prior distribution $\pi_0(\theta)$ and how well it replicates the posterior distributions of each data source.

Figure F-2 displays the directed acyclic graph of a pooled Bayesian inference of Poisson distributed data (failures over time).

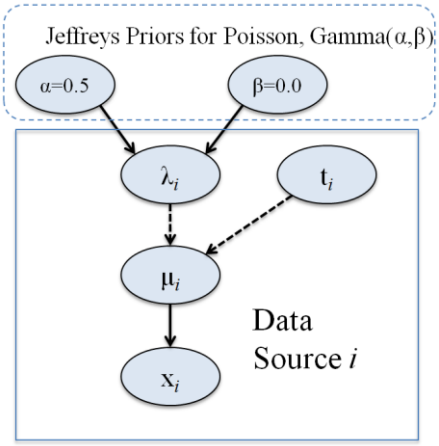


Figure F-2. Bayesian inference for a pooled analysis of failures versus time Poisson analysis.

Parameters α and β are parameters of a Gamma distribution, which is represented by the solid arrows leading to λ. This Gamma distribution starts as a non-informative prior; the Jeffreys prior is commonly used. Updating this model for each data source starting with the Jeffreys prior until the entire population is updated gives a posterior numerical distribution prediction for the entire population. Each data source uses this update to replicate its number of failures. If the replication is nearly or exactly equivalent to the number of failures for the data set, then the goodness-of-fit test passes. If this is the case, then a distribution is fitted to the properties of the numerical distribution (such as mean, percentiles, or standard deviation) and used to update with facility component performance data.

F-1.3 Population Variability Distribution Analysis

It is preferable to pool data when it is appropriate to do so; however, if goodness-of-fit tests prove that the data cannot be pooled, then an attempt is made to model the variability in the sources of the data within the data set. A viable PVD represents the variability from source to source of component data. Unfortunately, there may not exist a simple-looking PVD: the data themselves may belie a simple picture. However, constructing an honest variability distribution is preferable to pooling data that are patently inhomogeneous.

A PVD is a distribution that adequately represents the variance in the data sources of the data set. This is the top level of the hierarchical Bayesian inference required for this type of problem. The PVD distribution is on the input data at one level and the likelihood distribution for individual source outcomes in on a second level, which describes the hierarchy. Recall that the update rule for a single degree of freedom is shown in Equation (F-1).

Another integral is added to the denominator for each degree of freedom, whether that be a second parameter in the likelihood function or a second level of hierarchy.

A Poisson distribution is commonly used when one is examining a rate-based problem (failures experienced over an operating time). The distribution requires inputs for failures (x) and time (t) and will produce a rate (λ). Simply the Poisson is described as "x is distributed as (~) the Poisson of μ," which is equal to λt.

$$x \sim Poisson(\mu), where\ \mu = \lambda t \tag{F-3}$$

Figure F-3 displays the directed acyclic graph of a hierarchical Bayesian inference using Poisson parameters.

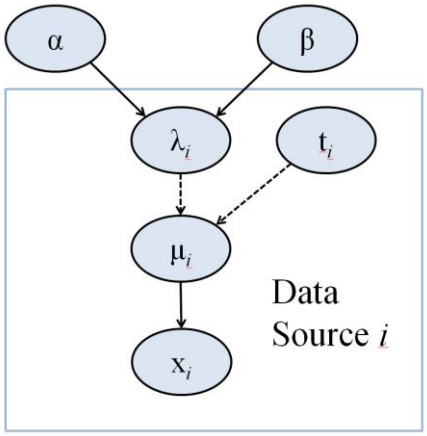


Figure F-3. Hierarchical Bayesian inference using the Poisson parameters.

The hyperpriors, α and β, are parameters of a Gamma distribution, which is represented by the solid arrows leading to λ. This Gamma distribution is the PVD and defines λ for the first hierarchy. The second hierarchy is the Poisson distribution, which provides the posterior for each data source.

Generally, in hierarchical Bayes, if the parameter of interest is denoted as π(θ), then the prior distribution is written as:

$$\pi(\theta) = \int \pi_1(\theta|\varphi)\, \pi_2(\varphi) d\varphi \tag{F-4}$$

where $\pi_1(\theta|\varphi)$ is the first stage prior that represents the population variability in θ for a given value of φ, which is the vector $(\alpha, \beta)^T$.

Further broken down into terms of α and β, the first stage prior is defined as:

$$\pi_1(\lambda) = \iint \pi_0(\lambda|\alpha,\beta)\, \pi_0(\alpha,\beta) d\alpha d\beta \tag{F-5}$$

The hyperpriors, α and β, are not defined as discrete values in the non-pooled inference model. Instead, they are defined as diffuse, or flat values over the breadth of possible values. A Gamma distribution with α and β both equal to zero is a good example of a diffuse prior for use in hierarchical Bayes. Using MCMC, α and β are given starting points and samples are taken until the values converge to the Gamma parameters of the prior for use in the PVD.

F-1.4 Data Source Grouping Analysis

A quantitative tool to help in identifying groups within a population data set is called cluster analysis. Many algorithms have been developed for use in the area of data mining. One such algorithm proposed

for use in heterogeneous populations is Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [F-3]. DBSCAN uses the data points in a matrix to determine the groups that are closely packed together (points that have many nearby neighbors) [F-4].

Two parameters are used in DBSCAN, the first is called the Epsilon Neighborhood of a point. This specifies the distance at which to determine if two data points representing a set of covariates are within the same neighborhood.

$$N_{Eps}(p) = \{q \in D | dist(p,q) \leq Eps\} \tag{F-6}$$

The second parameter is the Minimum Points, which is the minimum number or points to lie within a neighborhood to determine if the points are within a group.

$$p \in N_{Eps}(q)$$
$$\left|N_{Eps}(q)\right| \geq MinPts \tag{F-7}$$

The concept of direct density reachability in the DBSCAN algorithm states that the two points, *p* and *q*, are in the same Epsilon Neighborhood and there are a specified number of points within that neighborhood to call it a cluster. Further, a third point, *o*, is also density reachable with respect to N_{Eps} and *MinPts*. For more information, see [F-3] and [F-4].

DBSCAN is available via many statistical analysis program packages, including the open source and free program R. The data is set into a matrix of covariates. This can be as simple as failures and time or failures and demands. It can also include more covariates such as environmental parameters such as temperature, humidity, manufacturer, etc. However, the data matrix must be entered numerically.

The clusters that DBSCAN provides can be further analyzed in a qualitative manner to determine what caused the data to cluster in that way. Was it an environmental difference? Was it a manufacturer? Was the component used in a different manner? Can data sources be discarded as outliers? These are all questions that cluster analysis can help answer.

An output graph showing the grouped points and outliers is presented in Figure F-4. The data set grouped here consisted of only two covariates, time (years in y-axis) and failures (x-axis). Listing the indices from the matrix for the groups then allows a PVD analysis to be performed for each group and a mixture prior set up for use in updating facility data.

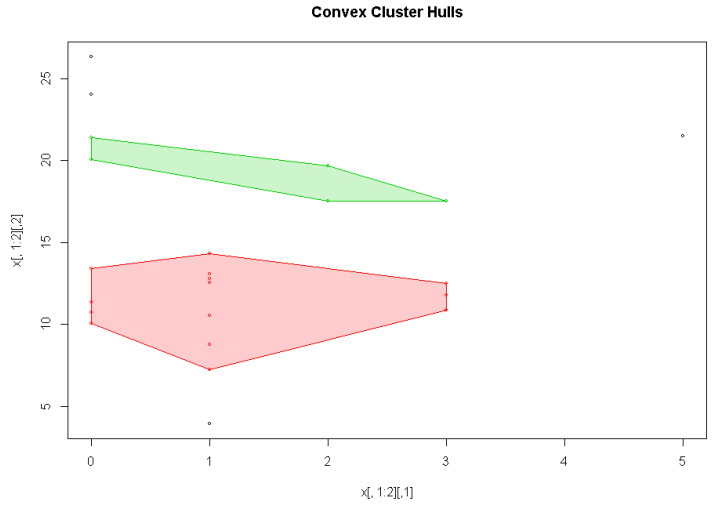


Figure F-4. Convex cluster hulls.

F-1.5 Mixture Priors

A mixture prior uses all the information from the data set in a weighted manner. The prior in the Bayesian inference formula (π_0) is a sum of the component parts that are PVDs of the groups found through cluster analysis.

$$\pi_1(\theta|E) = \frac{L(E|\theta)\pi_0(\theta)}{\int L(E|\theta)\pi_0(\theta)d\theta} \quad \text{(F-8)}$$

$$\pi_0 = \sum_n^{i=1} w_i\pi_i + w_{i+1}\pi_{i+1} + \cdots w_n\pi_n \quad \text{(F-9)}$$

$$\sum_n^{i=1} w_i + w_{i+1} + \cdots w_n = 1.0 \quad \text{(F-10)}$$

where:

w is the weight of the group as a ratio of the entire population. If 10 indices of the matrix out of 100 are in the group, then the weight will be 0.10. Weights must sum to 1.0.

π_0 is the mixture prior distribution for use in the Bayesian inference formula to find the posterior update of the facility component performance.

F-2. EXAMPLES OF POPULATION ANALYSIS

The following examples use rate-based data based on time used and failures experienced. Following these examples are another set of examples using demand-based data.

F-2.1 Example of a Population Pooling Test

Data for a component based on time used and failures experienced are presented in Table F-1. This could be any failure mode for any component. This particular data set is from a textbook example in [F-5].

Table F-1. Component failure rate data.

Source	Failures	Exposure Time (years)
1	2	15.986
2	1	16.878
3	1	18.146
4	1	18.636
5	2	18.792
6	0	18.979
7	12	18.522
8	5	19.040
9	0	18.784
10	3	18.868
11	0	19.232

This example uses the data presented in Table F-1 and is for a set of data with failure counts over time. It uses OpenBUGS as the analysis tool. The OpenBUGS script is shown in Figure F-5.

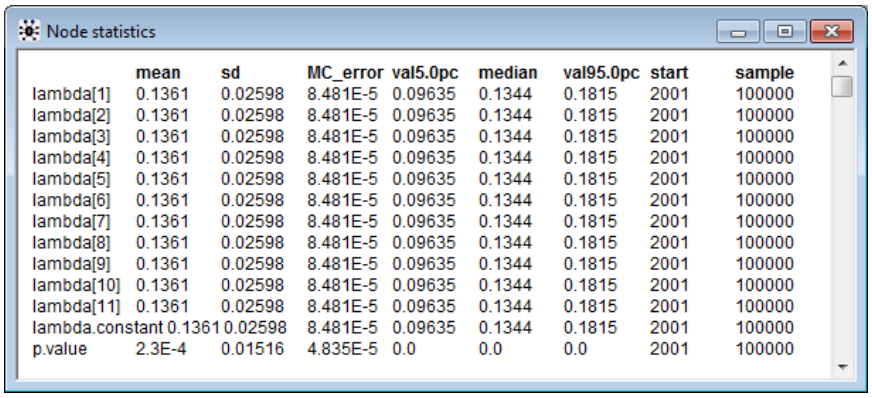
Key parameters in this model are:

- x: Failures for each source listed in the data
- time: Time in years for each source listed in the data
- mean: The Poisson parameter = lambda * time
- lambda: The rate in failures per year
- lambda.constant: The defined rate which is inferred upon the lambda for each data set in the pooled test
- x.rep: The replicated Poisson result for x for each source to use in the Chi-squared comparison.

```
model  {
for (i in 1 : N)  {
   x[i] ~ dpois(mean[i])                  #Poisson dist. for events
   mean[i] <- lambda[i] * time[i]         #Poisson parameter for each unit
   x.rep[i] ~ dpois(mean[i])       #Replicate value from posterior predictive distribution
   lambda[i] <- lambda.constant       #Check for poolability

#Generate inputs for Bayesian p-value calculation
   diff.obs[i] <- pow(x[i] - mean[i], 2)/mean[i]
   diff.rep[i] <- pow(x.rep[i] - mean[i], 2)/mean[i]

   }

# Calculate Bayesian p-value
chisq.obs <- sum(diff.obs[])
chisq.rep <- sum(diff.rep[])
p.value <- step(chisq.rep - chisq.obs) # Mean of this node should be near 0.5
lambda.constant ~ dgamma(0.5, 0.0001) #Jeffreys prior for lambda
}

inits
list(lambda.constant = 0.001)

data
x[]    time[]
2      15.986
1      16.878
1      18.146
1      18.636
2      18.792
0      18.976
12     18.522
5      19.04
0      18.784
3      18.868
0      19.232
END

list(N=11)
```

Figure F-5. OpenBUGS model to check for poolability of rate-based data.

The results of running the model with 2,000 samples to "burn-in" and converge, with 100,000 samples taken for results are shown in Figure F-6.



	mean	sd	MC_error	val5.0pc	median	val95.0pc	start	sample
lambda[1]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[2]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[3]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[4]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[5]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[6]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[7]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[8]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[9]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[10]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda[11]	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
lambda.constant	0.1361	0.02598	8.481E-5	0.09635	0.1344	0.1815	2001	100000
p.value	2.3E-4	0.01516	4.835E-5	0.0	0.0	0.0	2001	100000

Figure F-6. Rate-based pooled model results.

Note that the individual sources are using the constant lambda (lambda.constant) that has defined values for its Gamma distribution based on the Jeffreys prior. The lambda.constant is set with an initial value of 0.001 in the MCMC in order toto initialize the model, but it converges to the 0.1361 per year value based on the inference from the data set. The replicated Poisson result from each source is compared to the pooled result to determine the P-value (p.value), which is the measure of model fit to the data. A perfect P-value would be 0.5, with values closer to zero or one indicating a poor fit. In this case, the P-value of 2.3E-04 (on the bottom left of Figure F-6) indicates that the pooled model is not a good fit to the data.

F-2.2 Example of Full Population Variability Distribution Estimation

For the data set in Table F-1, the pooling test indicates that the data should not be pooled. The next step is to see if a PVD can be fit to the data by using a hyperprior distribution to represent the variation of data sources as discussed above.

A hyperprior that is commonly used with Poisson distributed data is a Gamma distribution. Other distributions can be utilized, such as lognormal. The reader is directed to [F-2] and [F-5] for further guidance. A Gamma distribution was used for this analysis.

The hyperprior should not influence the model; rather, the model should drive the parameters of the hyperprior distribution to the values that fit the data. For this reason, the parameters of the Gamma distribution are in turn represented hyperpriors of diffuse Gamma values that produce as flat a distribution over the realm of values as possible, given an initial starting point, and then the MCMC uses the Bayesian inference to drive the Gamma parameters to converged values. This example uses the data presented in Table F-1 and is for a set of data with failure counts over time. It uses OpenBUGS as the analysis tool. The OpenBUGS script is presented in Figure F-7.

Key parameters in this model are:

- x: Failures for each source listed in the data
- time: Time in years for each source listed in the data
- mean: The Poisson parameter = lambda * time
- lambda: The rate in failures per year
- lambda.constant: The defined rate which is inferred upon the lambda for each data set in the pooled test
- x.rep: The replicated Poisson result for x for each source to use in the Chi-squared comparison.

```
model {
for (i in 1 : N) {
   x[i] ~ dpois(mean[i])                  #Poisson dist. for events
   mean[i] <- lambda[i] * time[i]         #Poisson parameter for each unit
   x.rep[i] ~ dpois(mean[i])       #Replicate value from posterior predictive distribution
   lambda[i] ~ dgamma(alpha, beta) #Model variability in rate

#Generate inputs for Bayesian p-value calculation
   diff.obs[i] <- pow(x[i] - mean[i], 2)/mean[i]
   diff.rep[i] <- pow(x.rep[i] - mean[i], 2)/mean[i]

   }

# Hyperprior
lambda.pred ~ dgamma(alpha, beta)I(0,5)
alpha ~ dgamma(0.0001, 0.0001)   #Vague hyperprior for alpha
beta ~ dgamma(0.0001, 0.0001)    #Vague hyperprior for beta

# Calculate Bayesian p-value
chisq.obs <- sum(diff.obs[])
chisq.rep <- sum(diff.rep[])
p.value <- step(chisq.rep - chisq.obs) # Mean of this node should be near 0.5
}

inits
list(alpha = 1, beta = 1)
list(alpha = 0.5, beta = 0.5)

data
x[]    time[]
2      15.986
1      16.878
1      18.146
1      18.636
2      18.792
0      18.976
12     18.522
5      19.04
0      18.784
3      18.868
0      19.232
END

list(N=11)
```

Figure F-7. Hierarchical Bayes BUGS Language Model for source population variability in rate-based data.

F-2.2.1 Checking the Model for Convergence

Any MCMC program must converge before having confidence in the samples taken for results. Qualitative checks for convergence include looking at a graph of the histories of the key parameters. Running this model with two initial values for alpha and beta allows the check of convergence in these parameters so that there is confidence in the samples taken for results. In addition to the qualitative checks, a more quantitative test for convergence used in the OpenBUGS program is the Brooks-Gelman-Rubin statistic (BGR). Convergence is represented graphically in BGR by an R-value that is consistently at 1.0 and a B-value and W-value that are equivalent values to each other.

Figure F-8 shows the BGR test results for alpha and beta parameters in the model. Note that the number of iterations is half the numbers of samples since there are two chains compiled in the model. Alpha and beta R-values appear to settle solidly at 1 by 30,000 iterations with the other two parameters tracking together. This particular example takes longer to converge for a "picky" analyst than most do. For models that do not converge, the R-value will typically not settle on 1.0, and will wander significantly away from this value.

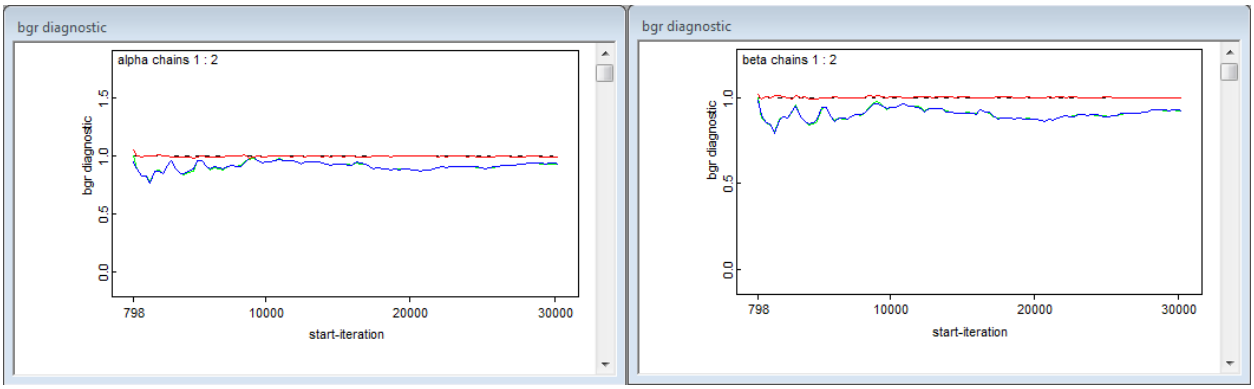


Figure F-8. BGR diagnostic test for convergence of gamma parameters.

To use as samples for the results, 100,000 iterations are run beyond the 60,000 burn-ins. A quick check of the BGR diagnostic for the duration of the sampling, shown in Figure F-9, does not show any significant events to question the validity of the calculations.

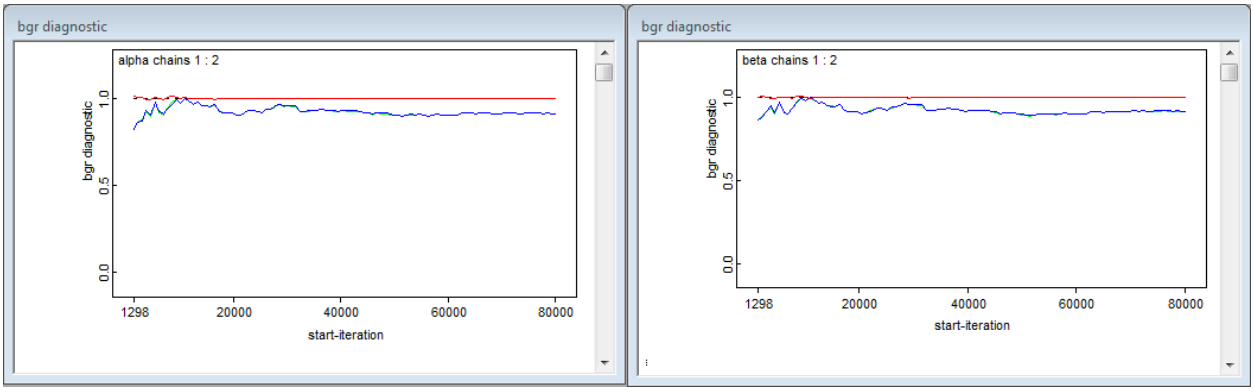


Figure F-9. BGR diagnostic for full sampling of gamma parameters.

F-2.2.2 Results of the Rate-Based Population Variability Analysis

Results of the analysis shown in Figure F-10 provide the following insights:

- The Chi-Square Bayesian P-value goodness-of-fit parameter is at 0.44, which is close to the ideal value of 0.5 and indicates high confidence in the predicted posterior results.
- The predicted posterior distribution that would be used for the PRA failure distribution for this component is a Gamma with alpha = 1.00 and beta = 7.76. Its mean is 1.56E-01 per year with a 5th percentile value of 8.95E-04 and a 95th percentile value of 5.38E-01. The Gamma(1.00, 7.76) distribution would be valid for use to update facility component performance until the next overall population update is performed.
- This prior, used in the Poisson model, provides the estimations of the lambda for the mean, 5th percentile and 95th percentile for each source and a predicted lambda as well.

Node statistics

	mean	sd	MC_error	val5.0pc	median	val95.0pc	start	sample
alpha	1.002	1.046	0.01994	0.2677	0.757	2.408	60001	200000
beta	7.763	8.578	0.1547	1.297	5.72	20.0	60001	200000
lambda[1]	0.1277	0.07714	1.791E-4	0.03198	0.1128	0.2742	60001	200000
lambda[2]	0.07949	0.05952	1.703E-4	0.01102	0.06587	0.1943	60001	200000
lambda[3]	0.07555	0.05653	1.66E-4	0.01041	0.06266	0.1847	60001	200000
lambda[4]	0.07413	0.05547	1.703E-4	0.01016	0.06142	0.1815	60001	200000
lambda[5]	0.1135	0.06835	1.648E-4	0.02852	0.1005	0.2433	60001	200000
lambda[6]	0.03368	0.03908	2.137E-4	1.206E-4	0.0202	0.1131	60001	200000
lambda[7]	0.516	0.166	9.353E-4	0.2771	0.4983	0.8156	60001	200000
lambda[8]	0.2302	0.09963	3.218E-4	0.09697	0.2146	0.4166	60001	200000
lambda[9]	0.034	0.03937	2.141E-4	1.271E-4	0.02043	0.114	60001	200000
lambda[10]	0.1526	0.07933	2.009E-4	0.05026	0.1387	0.3023	60001	200000
lambda[11]	0.03344	0.03879	2.128E-4	1.171E-4	0.01996	0.1123	60001	200000
lambda.pred	0.1564	0.2471	6.791E-4	8.954E-4	0.08433	0.5378	60001	200000
p.value	0.4478	0.4973	0.001448	0.0	0.0	1.0	60001	200000

Figure F-10. Rate-based results with gamma hyperprior.

A comparison of the lambdas and the predicted lambda as presented in Figure F-11 shows where the predicted value lies within the 5th to 95th percentile ranges of the sources.

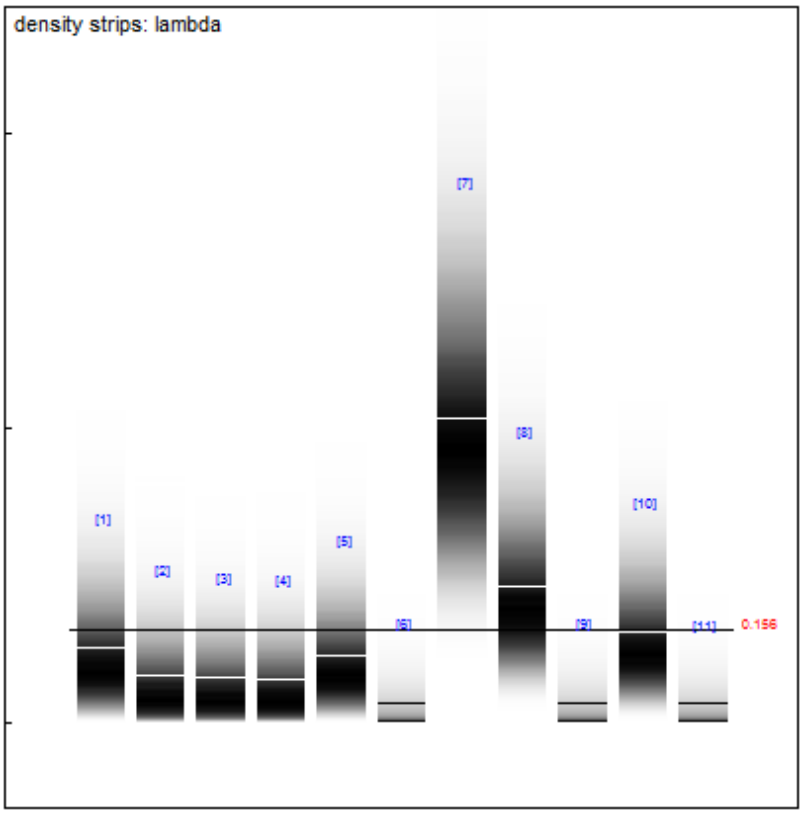


Figure F-11. Comparison of rate-based data source results.

F-2.3 Cluster Analysis Example

Data for a cluster analysis are presented in Table F-2. This failure rate data is from nuclear power plant loss of offsite power (LOOP) records, and deals with a population in which the parameters alpha and beta present difficulty converging to values.

Table F-2. Component failure rate data for a cluster analysis.

Source	Failures	Time (years)
1	1	13.054
2	1	12.77
3	1	7.22
4	1	3.944
5	1	10.548
6	0	10.704
7	0	24.0
8	1	8.76
9	3	11.79
10	2	17.5
11	0	20.03
12	0	13.39
13	5	21.5
14	0	10.075
15	0	26.32
16	1	12.54
17	3	17.5
18	1	14.3
19	3	10.89
20	3	12.5
21	0	21.38
22	2	19.65
23	0	11.34

There are multiple references online for using DBSCAN via R. The first step is to use the k-Nearest Neighbor distance plot to determine the knee in the graph. This is generally the best starting point for the Epsilon (Eps) parameter. It can be seen from Figure F-12 that the knee is approximately at 2.3 NN distance.

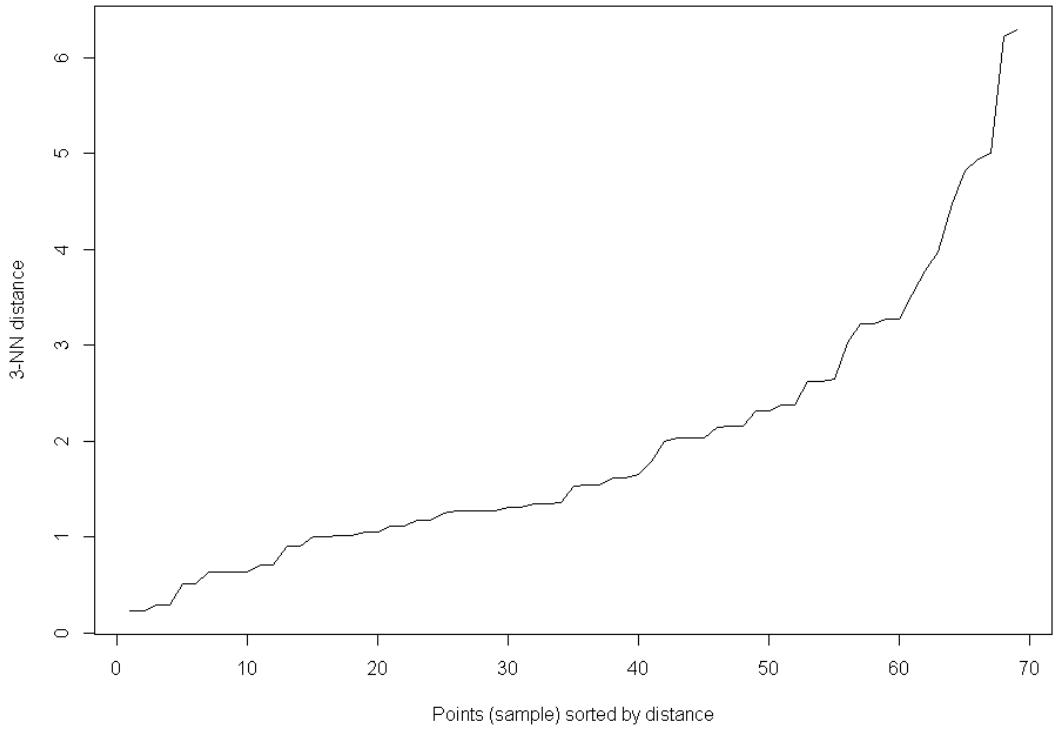


Figure F-12. k-nearest neighbor distance plot.

The next step is to run DBSCAN using the Eps and the Minimum Points (MinPts) parameters. Determining the MinPts parameter is less of a science than the Eps parameter. If one chooses too high of a value for MinPts, the algorithm will not find any clusters; too low of a value and it will find too many clusters. The default for MinPts is 5. For this data set, there are only 23 sources, and using a MinPts of 5 only generates one cluster of 14 and 9 outliers. Using a MinPts value of 3 produces two clusters and 4 outliers: 14 in Cluster 1 and 5 in Cluster 2. This is shown graphically in Figure F-13, with the two clusters and points in the clusters. The x-axis is the failures covariate and the y-axis is the time covariate.

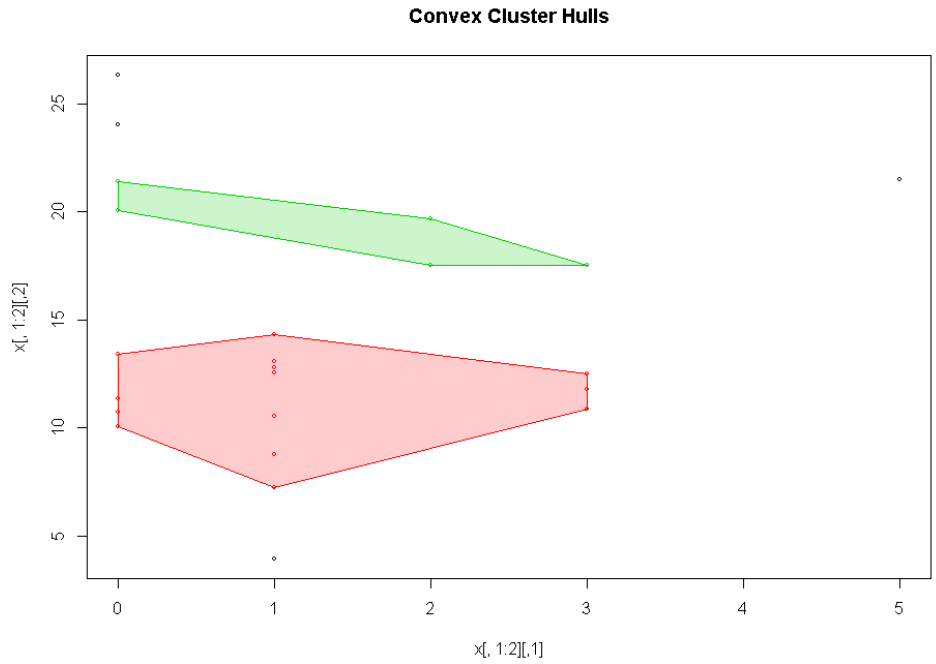


Figure F-13. Convex cluster hulls.

The indices of the data set that belong in each cluster can be extracted in R through the where() command. These are the source numbers in Table F-3.

Table F-3. Clusters with corresponding sources.

Cluster	Data Set Index (Source, from Table G-4)
0 (outliers)	4, 7, 13, 15
1	1, 2, 3, 5, 6, 8, 9, 12, 14, 16, 18, 19, 20, 23
2	10, 11, 17, 21, 22

The clusters can now be run through a PVD analysis to fit individual distributions. If the outliers fail to find a PVD as a group, then they can be re-ran as a cluster analysis as their own data set of four or find an update with a Jeffreys prior to use as individual distributions in the mixture prior.

F-2.4 Use of Mixture Prior Example

This example will use the data presented and grouped by the previous cluster analysis. There are three clusters identified in the cluster analysis, however, one of the clusters is identified as an outlier. Outliers are not related to each other and become their own group when setting up mixture priors. So in essence, there are six clusters to use in setting up the mixture prior.

A weight must be assigned for each cluster. The weight is equivalent to the proportion of the sources in the group to the overall number of sources in the population. Table F-4 summarizes the information.

Table F-4. Cluster weighting for mixture prior example.

Cluster	Weight	Source	Failures	Time (years)
A	0.0435	4	1	3.944
B	0.0435	7	0	24.0

C	0.0435	13	5	21.5
D	0.0435	15	0	26.32
E	0.6090	1	1	13.054
		2	1	12.77
		3	1	7.22
		5	1	10.548
		6	0	10.704
		8	1	8.76
		9	3	11.79
		12	0	13.39
		14	0	10.075
		16	1	12.54
		18	1	14.3
		19	3	10.89
		20	3	12.5
		23	0	11.34
F	0.2170	10	2	17.5
		11	0	20.03
		17	3	17.5
		21	0	21.38
		22	2	19.65

The next step is to see if the individual clusters are poolable. If they are not, then tighter set of parameters are required when using the data mining in order to produce smaller clusters.

The poolability OpenBUGS model for rate-based data is used as previously described.

The results of the p.values for this analysis as shown in Figure F-14 show that it is reasonable to pool the data for each cluster since neither Cluster E nor Cluster F have a p-value that is close to zero or 1 as was the case when testing the entire population in the prior rate example. Ideal fit would be 0.5, however, the range between 0.2 and 0.8 can be considered acceptable.

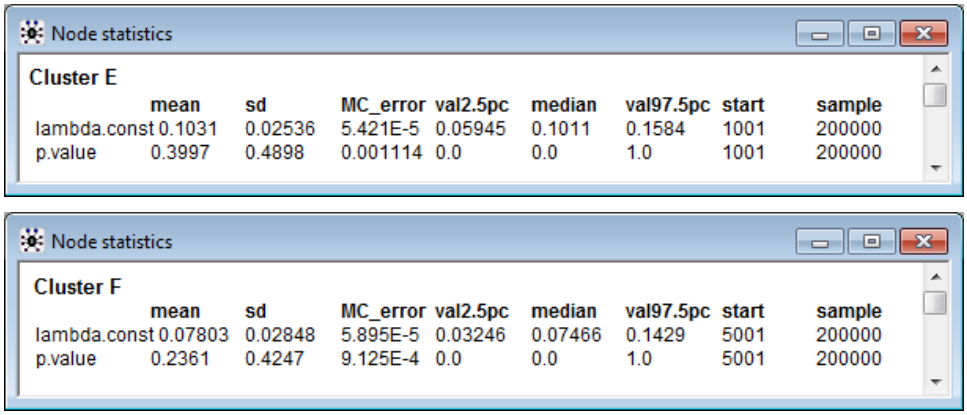


Figure F-14. Pooling result with P values for mixture prior example.

The next step is to pool the data in the clusters with multiple sources by summing the failures and time components. The mixture prior data now consists of that shown in Table F-5.

Table F-5. Clusters with corresponding sources for mixture prior example.

Cluster	Weight	Source	Failures	Time (years)
A	0.0435	4	1	3.944
B	0.0435	7	0	24.0
C	0.0435	13	5	21.5
D	0.0435	15	0	26.32
E	0.6090	1, 2, 3, 5, 6, 8, 9, 12, 14, 16, 18, 19, 20, 23	16	159.881
F	0.2170	10, 11, 17, 21, 22	7	96.06

The model shown in Figure F-15 uses the mixture priors by weighting each cluster's data input through the use of a categorical distribution, of which all components (weights) of the distribution must sum to one.

```
Mixture Prior Example

model{
  for (i in 1:6) {
    lambda[i] ~ dgamma(0.5, 0.0001) #Jeffreys priors for lambda
    x[i] ~ dpois(mean[i])              #Poisson likelihood
    mean[i] <- lambda[i] * time[i]    #Poisson mean
  }
lambda.avg <- lambda[r]   #Weighted use of priors
r ~ dcat(p[]) #Assignment of weights, note "r" is just a variable name
}


Data
list(p=c(0.0435, 0.0435, 0.0435, 0.0435, 0.6090, 0.2170), x=c(1, 0, 5,
0, 16, 7), time=c(3.944, 24.0, 21.5, 26.32, 159.881, 96.06))
```

Figure F-15. Mixture prior example.

Note the lack of a requirement to use initial values to start the MCMC. In the case where each sub-population (cluster) is poolable, the program can usually generate its own initial values and burn-in quickly. For this example, 1,000 samples were used.

The results are shown in Figure F-16. Lambda[1] through lambda[6] are the results for each of the clusters. The node "lambda.avg" is a multi-modal distribution of which its mean, percentiles, and/or standard deviation would be used to fit a traditional distribution for use in industry failure data.

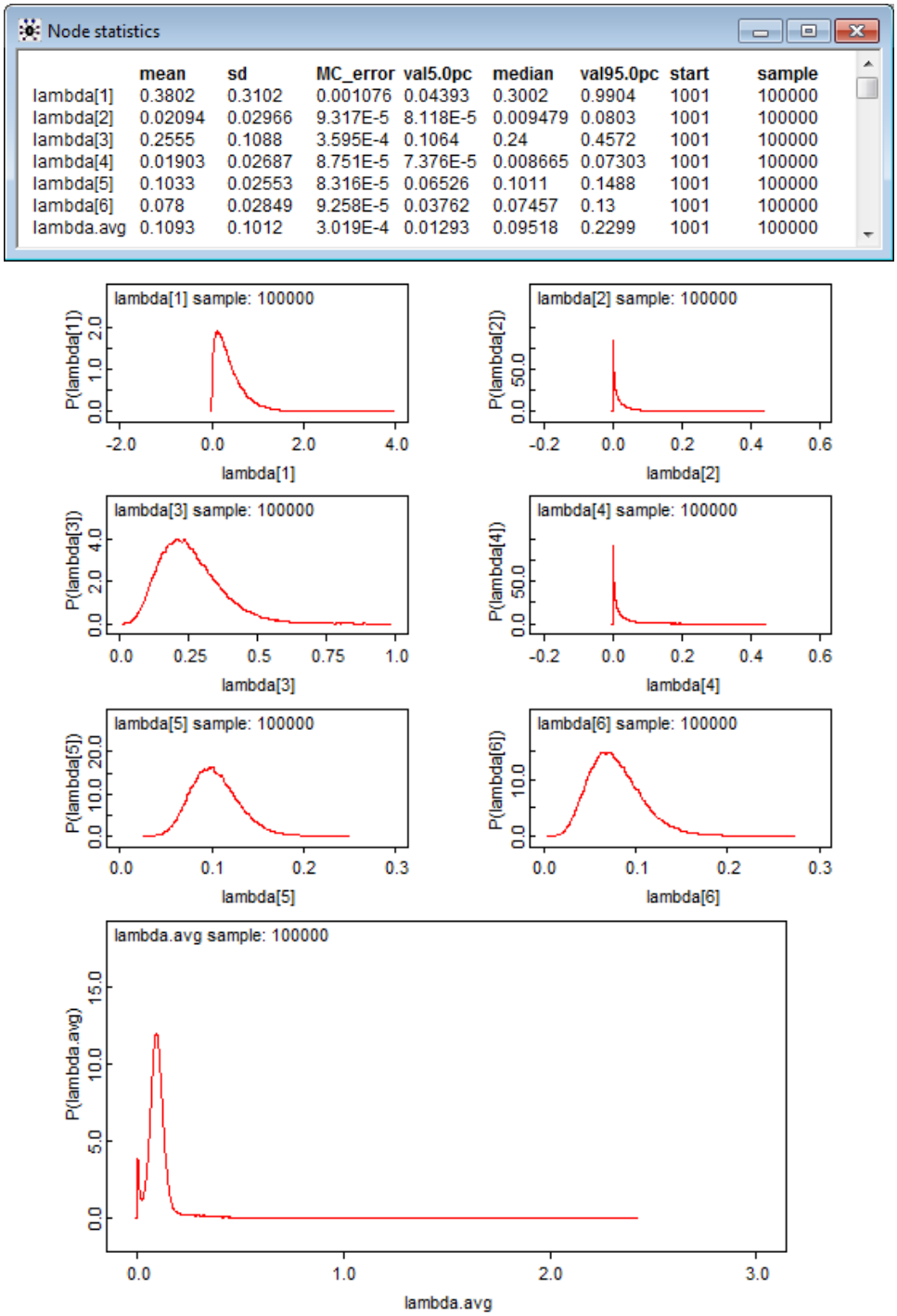


Figure F-16. Convergence test result for mixture prior example.

F-2.5 Demand-Based Component Population Variability Example

Data for a component based on demands and failures experienced are presented in Table F-6. This could be any failure mode for any component and differences in manufacturer or operating conditions should be kept in mind in case the set cannot be pooled and a PVD cannot be fit. This particular data is from a textbook example in [F-5].

Table F-6. Component demand-based failure data.

Source	Failures	Demands
1	0	140
2	0	130
3	0	130
4	1	130
5	2	100
6	3	185
7	3	175
8	4	167
9	5	151
10	10	150

F-2.5.1 Testing for Pool-ability of the Data

A first qualitative look at the data shows what appears to be an outlier in Source 10. This is good to note in case the data cannot be pooled or a PVD cannot be applied.

The first quantitative analysis should be to see if the data can be pooled. An OpenBUGS model to check for pooling applicability is presented in Figure F-17. Note that any MCMC program capable of Bayesian inference will work and that OpenBUGS is used here as an example. The goodness of fit test in this model is a Chi-squared test of the constant probabilities for each source determined by the model versus the replicated results using the posterior predictive distribution which in this case is a Jeffreys prior which adds minimal influence on the data.

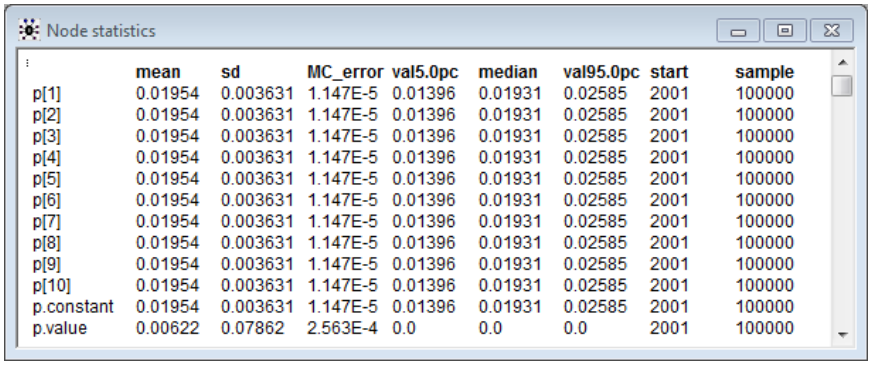
Key parameters in this model are:

- x: Failures for each source listed in the data
- n: Number of demands for each source listed in the data
- N: Number of sources
- p: The probability of failure per demand
- x.rep: The replicated Binomial result for x from the posterior predictive distribution.

```
model {
for (i in 1 : N) {
   x[i] ~ dbin(p[i], n[i])                                    #Binomial dist. for failures
   p[i] <- p.constant                                         #Pooling test Jeffreys prior
   x.rep[i] ~ dbin(p[i], n[i])          #Replicate from posterior predictive distribution

#Generate inputs for Bayesian p-value calculation
   diff.obs[i] <- x[i] - p[i]*n[i]              #Diff between observed and expected x
   chisq.obs[i] <- pow(diff.obs[i],2)/(n[i]*p[i]*(1-p[i]))        #Observed Chi-square
   diff.rep[i] <- x.rep[i] - p[i]*n[i]          #Diff between replicated and expected x
   chisq.rep[i] <- pow(diff.rep[i],2)/(n[i]*p[i]*(1-p[i]))        #Replicated Chi-square
   }
p.constant ~ dbeta(0.5,0.5)                     #Jeffreys prior to test for poolability
chisquare.obs<-sum(chisq.obs[])
chisquare.rep<-sum(chisq.rep[])
p.value <- step(chisquare.rep - chisquare.obs)          #Mean should be near 0.5
}

data
x[] n[]
0  140
0  130
0  130
1  130
2  100
3  185
3  175
4  167
5  151
10  150
END

list(N = 10)
```

Figure F-17. OpenBUGS model to check for poolability of demand-based data.

The results of running the model with 2,000 samples to "burn-in" and converge, with 100,000 samples taken for results are shown in Figure F-18.



	mean	sd	MC_error	val5.0pc	median	val95.0pc	start	sample
p[1]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p[2]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p[3]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p[4]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p[5]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p[6]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p[7]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p[8]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p[9]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p[10]	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p.constant	0.01954	0.003631	1.147E-5	0.01396	0.01931	0.02585	2001	100000
p.value	0.00622	0.07862	2.563E-4	0.0	0.0	0.0	2001	100000

Figure F-18. Pooling result with P values for demand-based example.

Note that the individual sources are using the constant probability which has defined values for its Beta distribution based on the Jeffreys prior. The p.constant does not require an initial set value in this particular model because OpenBUGS is able to generate the initial value on its own. The p.constant converges to the 1.95E-02 failures/demand based on the inference from the data set. The replicated

Binomial result from each source is compared to the pooled result to determine the P-value (p.value) as with the previous rate-based example. The P-value of 6.22E-03 indicates that the pooled model is not a good fit to the data.

F-2.5.2 Demand-Based Heirarchical Bayes for a Population Variability Distribution Estimation

Now that it is determined that the data set cannot be pooled, the next step is to use a hyperprior to represent the variability of the data from source to source and check to see if this model can fit the data.

A hyperprior is selected to attempt to fit the data source variability. The most common hyperprior distributions used for a Binomial model are the Beta, but other distributions such as the Lognormal can be used. A Beta distribution is used in this example. The hyperprior should not influence the model; rather the model should drive the parameters of the hyperprior distribution to the values that fit the data. For this reason, the parameters of the Beta distribution are in turn represented by diffuse values, given an initial starting point and then the MCMC uses Bayesian inference to drive the Beta parameters to converged values.

The model in Figure F-19 uses the Beta prior to infer upon the probability in the Binomial distribution in a similar manner that the fixed distribution of the Jeffreys Beta prior was used to test for poolability of the data. The Gamma hyperprior representing the Beta parameters are the PVD that helps to fit the data. Predicted performance from the population is given by the posterior Beta distribution (p.pred) with the parameters alpha and beta. Note that the prior being used here allows the parameters to vary based upon the distributions they are defined by so the MCMC will have to run enough samples to converge both the alpha and the beta parameters.

```
model {
for (i in 1 : N) {
   x[i] ~ dbin(p[i], n[i])                                  #Binomial dist. for failures
   p[i] ~ dbeta(alpha, beta)                                #First stage prior
   x.rep[i] ~ dbin(p[i], n[i])          #Replicate from posterior predictive distribution

#Generate inputs for Bayesian p-value calculation
   diff.obs[i] <- x[i] - p[i]*n[i]                #Diff between observed and expected x
   chisq.obs[i] <- pow(diff.obs[i],2)/(n[i]*p[i]*(1-p[i]))         #Observed Chi-square
   diff.rep[i] <- x.rep[i] - p[i]*n[i]            #Diff between replicated and expected x
   chisq.rep[i] <- pow(diff.rep[i],2)/(n[i]*p[i]*(1-p[i]))         #Replicated Chi-square
   }

p.pred ~ dbeta(alpha,beta)             #PVC and posterior predictive distribution
alpha ~ dgamma(0.0001, 0.0001)    #Diffuse hyperprior for alpha
beta ~ dgamma(0.0001, 0.0001)     #Diffuse hyperprior for beta

chisquare.obs<-sum(chisq.obs[])
chisquare.rep<-sum(chisq.rep[])
p.value <- step(chisquare.rep - chisquare.obs)          #Mean should be near 0.5
}

inits
list(alpha = 1, beta = 30)
list(alpha = 0.5, beta = 70)

data
x[] n[]
0  140
0  130
0  130
1  130
2  100
3  185
3  175
4  167
5  151
10  150
END

list(N = 10)
```

Figure F-19. Hierarchical Bayes model for source population variability in demand-based data.

F-2.5.3 Convergence of the Hyperprior Parameters

The BGR diagnostic for the alpha and beta parameters in Figure F-20 shows some wild fluctuations prior to approximately 40,000 iterations (80,000 samples) at which point the R-value settles along the 1.0 line and the other two measures track with each other. Another BGR is performed to make sure that nothing out of the ordinary happened during the sampling for analysis. Figure F-21 shows slight bumps away from exactly 1.0, as viewed around 60,000 iterations of each parameter. This is not out of the ordinary and does not indicate divergence. An example of divergence would be continued behavior such as noted prior to 35,000 iterations.

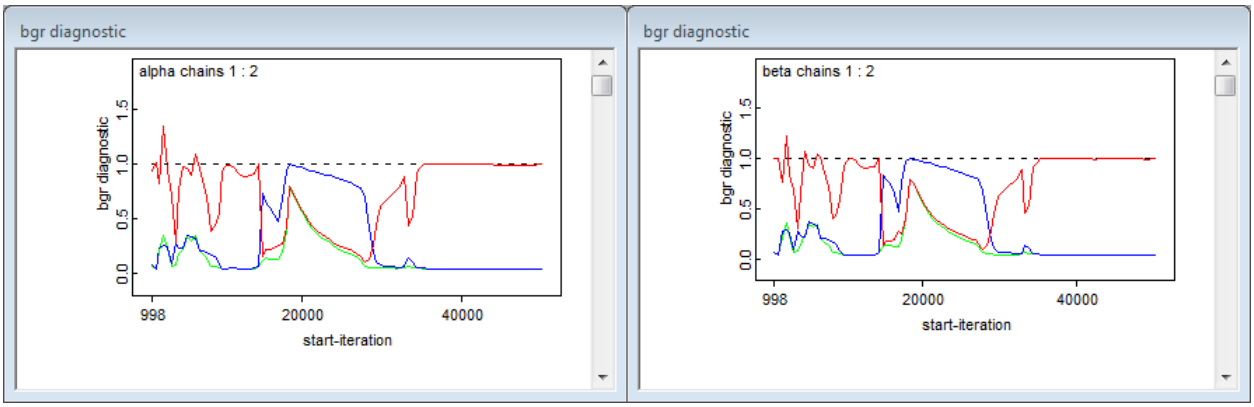


Figure F-20. BGR diagnostic for convergence of beta parameters.

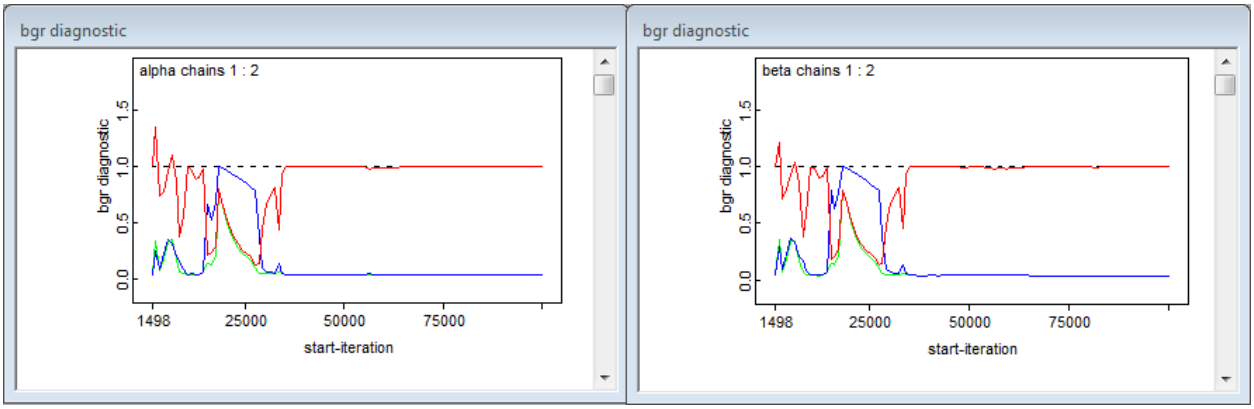


Figure F-21. BGR diagnostic for full sampling of beta parameters.

F-2.5.4 Results of the Demand-based Population Variability Analysis

Results of the analysis are shown in Figure F-22 and provide the following insights:

- The predicted posterior distribution that would be used for PRA failure data for this component is a Beta with alpha = 2.225 and beta = 118.6. Its mean is 2.11E-02 per demand, with a 5th percentile value of 3.68E-04 and a 95th percentile value of 6.38E-02.
- This prior, used in the Binomial model, provides the estimations of the probability of failure per demand for the mean, 5th percentile and 95th percentile for each source and a predicted probability.
- The P-value goodness-of-fit parameter is at 0.43, which is close to the ideal value of 0.5 and indicates high confidence in the predicted posterior distribution results.

Node statistics

	mean	sd	MC_error	val5.0pc	median	val95.0pc	start	sample
alpha	2.225	6.021	0.195	0.3137	1.098	6.15	80001	200000
beta	118.6	315.1	10.24	11.92	58.65	335.8	80001	200000
p[1]	0.00628	0.006197	6.685E-5	5.82E-5	0.004428	0.01869	80001	200000
p[2]	0.006555	0.006446	6.735E-5	5.971E-5	0.004664	0.01945	80001	200000
p[3]	0.006532	0.006417	6.633E-5	5.952E-5	0.004654	0.01931	80001	200000
p[4]	0.01161	0.007862	4.143E-5	0.00187	0.01011	0.02639	80001	200000
p[5]	0.01959	0.01112	2.736E-5	0.005537	0.01757	0.04062	80001	200000
p[6]	0.01694	0.0082	2.092E-5	0.005924	0.01573	0.03213	80001	200000
p[7]	0.0176	0.008529	2.164E-5	0.006167	0.0163	0.03343	80001	200000
p[8]	0.02239	0.009914	3.161E-5	0.009096	0.02086	0.0409	80001	200000
p[9]	0.02851	0.01185	5.798E-5	0.01264	0.02665	0.05071	80001	200000
p[10]	0.05133	0.01807	1.775E-4	0.02534	0.04934	0.08417	80001	200000
p.pred	0.02107	0.0291	7.455E-5	3.684E-4	0.01402	0.06376	80001	200000
p.value	0.4254	0.4944	0.002484	0.0	0.0	1.0	80001	200000

Figure F-22. Demand-based results with beta prior.

A comparison of the probabilities (p) and the predicted probability presented in Figure F-23 shows where the predicted value lies within the 5th to 95th percentile ranges of the sources.

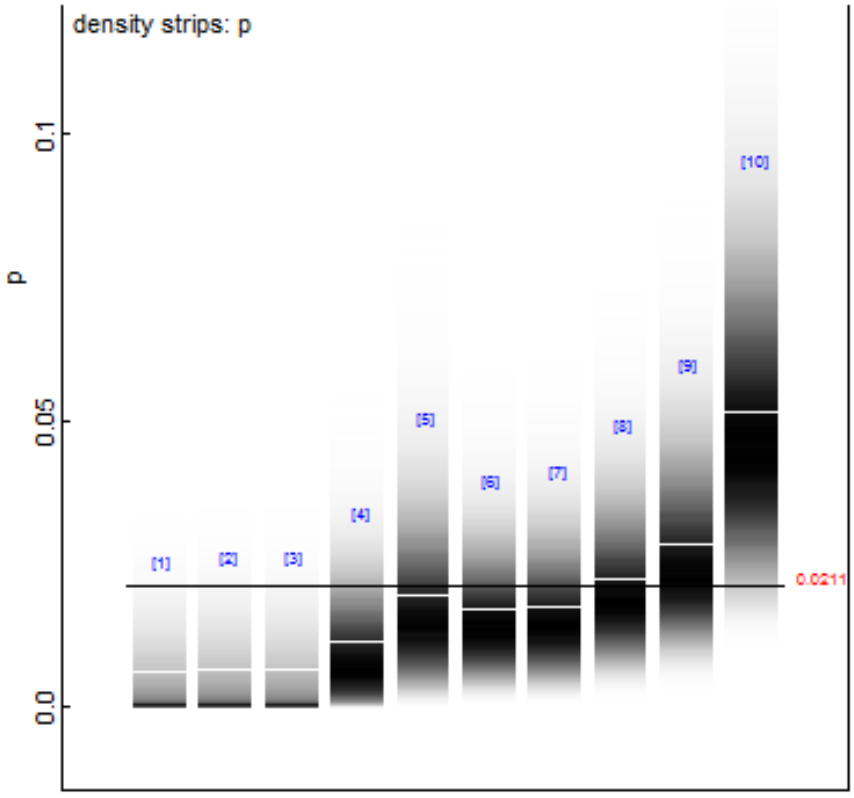


Figure F-23. Comparison of demand-based source results.

F-2.5.5 Pitfalls of MCMC and Selection of Hyperpriors

Convergence Issues

In the detailed demand-based data example used above, the alpha and beta variables in the Beta distribution were slow to converge. Generally, it is best to run as many samples as required to attain converged samples for measure. However, if the variables continue to refuse to converge it sometimes helps to reparameterize the distribution in terms such as mean and variance. If secondary parameters are

used, then take the samples for measure after they converge and use the secondary parameters to attain the primary ones for use in the PVD. For more information on this topic, see [F-6]. If running more samples and reparameterization does not work, then group analysis must be performed on the population using DBSCAN or another cluster algorithm, individual PVD analysis performed for each group, and a mixture prior set up for use as the current state of knowledge.

Choosing Adequate Hyperpriors

MCMC programs use random "picks" of a simulation across the breadth of the posterior distribution. For distributions with long tails this can present problems where the mean can be in the tail, sometimes even beyond the 95th percentile. Care must be taken by the analyst to choose a prior that will not only cause the posterior to fit the data, but will also produce logical results. An example using the rate-based data set follows.

Another prior that is popular to use with Poisson data is the lognormal. An OpenBUGS script using the lognormal as the hyperprior along with the sample data from Table F-1 is shown in Figure F-24.

```
model {
for (i in 1 : N) {
   x[i] ~ dpois(mean[i])                  #Poisson dist. for events
   mean[i] <- lambda[i] * time[i]         #Poisson parameter for each unit
   x.rep[i] ~ dpois(mean[i])       #Replicate value from posterior predictive distribution
   lambda[i] ~ dlnorm(mu, tau) #Model variability in rate

#Generate inputs for Bayesian p-value calculation
   diff.obs[i] <- pow(x[i] - mean[i], 2)/mean[i]
   diff.rep[i] <- pow(x.rep[i] - mean[i], 2)/mean[i]

   }

# Lognormal Hyperprior
lambda.pred ~ dlnorm(mu, tau)
mu ~ dflat()
sigma ~ dunif(0, 10)
tau <- pow(sigma, -2)

# Calculate Bayesian p-value
chisq.obs <- sum(diff.obs[])
chisq.rep <- sum(diff.rep[])
p.value <- step(chisq.rep - chisq.obs) # Mean of this node should be near 0.5
}

inits
list(mu=-5, sigma=5)
list(mu=-7, sigma=0.5)

data
x[]    time[]
2      15.986
1      16.878
1      18.146
1      18.636
2      18.792
0      18.976
12     18.522
5      19.04
0      18.784
3      18.868
0      19.232
END

list(N=11)
```

Figure F-24. Poisson model with lognormal prior.

This model's parameters for the lognormal PVD converge much more quickly. By 10,000 iterations the BGR R-value is solidly at 1.0 and the other two are tracking each other. Running the model for an additional 100,000 iterations gives us 200,000 samples, the same number as was used in the Gamma distribution hyperprior. This produces the results shown in Figure F-25.

Node statistics

	mean	sd	MC_error	val5.0pc	median	val95.0pc	start	sample
lambda[1]	0.1155	0.07585	2.446E-4	0.02678	0.09863	0.2613	10001	200000
lambda[2]	0.06978	0.05479	2.278E-4	0.01016	0.05614	0.177	10001	200000
lambda[3]	0.06571	0.05123	2.068E-4	0.009704	0.05292	0.1655	10001	200000
lambda[4]	0.06486	0.05052	2.045E-4	0.009588	0.05231	0.1634	10001	200000
lambda[5]	0.102	0.06574	2.213E-4	0.0242	0.0877	0.2279	10001	200000
lambda[6]	0.03387	0.03488	1.927E-4	0.001222	0.02315	0.103	10001	200000
lambda[7]	0.5769	0.178	5.453E-4	0.318	0.5591	0.8973	10001	200000
lambda[8]	0.2289	0.1053	3.07E-4	0.09004	0.2118	0.4261	10001	200000
lambda[9]	0.03396	0.03493	2.003E-4	0.001253	0.02323	0.1032	10001	200000
lambda[10]	0.1423	0.07976	2.486E-4	0.04299	0.1268	0.2943	10001	200000
lambda[11]	0.03369	0.03474	2.001E-4	0.001213	0.02314	0.1023	10001	200000
lambda.pred	88.36	26660.0	59.43	0.003114	0.06976	0.9319	10001	200000
mu	-2.76	0.6488	0.005352	-3.918	-2.685	-1.872	10001	200000
p.value	0.4504	0.4975	0.001223	0.0	0.0	1.0	10001	200000
sigma	1.548	0.7168	0.005968	0.7365	1.393	2.874	10001	200000
tau	0.7061	1.323	0.009733	0.1211	0.5156	1.844	10001	200000

Figure F-25. Rate-based results with lognormal hyperprior.

The Bayesian P-value of 0.45 shows that this model replicates the data as well as the Gamma hyperprior model did. A review of the lambdas for the data set shows that the means are close to the ones calculated for the Gamma model. However, a study of the PVD (lambda.pred) shows that the result for the mean (88.36 failures/year) is extreme and well beyond the 95th percentile of 0.93 failures/year. The mean lies in the heavy tail due to the MCMC picking some extreme values in the tail.

This is an anomaly where the goodness-of-fit measure says that the lognormal model replicates the data just as well as the Gamma model, yet the predictive posterior distribution's mean is not logical based on the most extreme case in the data set (lambda[7]) having a 95th percentile result of 0.90 failures/year. This is also intuitively a "wrong" answer because a qualitative look at the data tells us that there is very little chance of 88 failures in a year. Figure F-26 displays the full probability density function and the section near zero using just the predicted μ and σ. This illustrates a sharp peak very close to zero and a long tail.

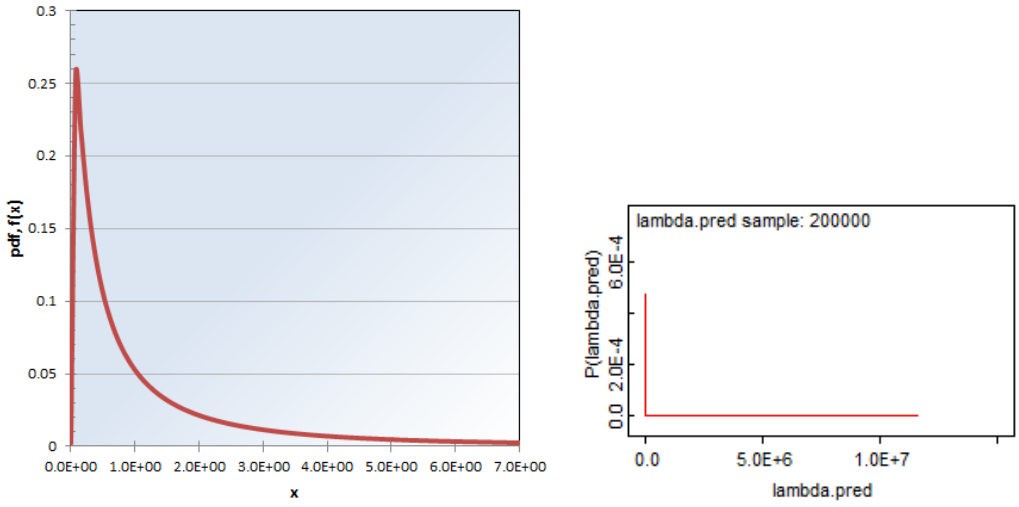


Figure F-26. Probability density function with lognormal hyperprior example.

Further analysis of the two priors can be performed by truncating the Gamma and lognormal priors using the OpenBUGS interval command of "I(x,y)" where x is the lower number and y is the highest number in the results to consider. The OpenBUGS script used for comparison of the two hyperpriors is shown in Figure F-27. Note that the interval command is placed inline and behind the distribution text. The use of "#" comments out the hyperprior not currently in use.

```
model {
for (i in 1 : N) {
   lambda[i] ~ dgamma(alpha, beta)I(0,5) #Model variability in rate
#   lambda[i] ~ dlnorm(mu, tau)I(0,5) #Model variability in rate
   mean[i] <- lambda[i] * time[i]        #Poisson parameter for each unit
   x[i] ~ dpois(mean[i])                 #Poisson dist. for events
   x.rep[i] ~ dpois(mean[i])      #Replicate value from posterior predictive distribution

#Generate inputs for Bayesian p-value calculation
   diff.obs[i] <- pow(x[i] - mean[i], 2)/mean[i]
   diff.rep[i] <- pow(x.rep[i] - mean[i], 2)/mean[i]

   }

#######Hyperpriors########

lambda.pred ~ dgamma(alpha, beta)I(0,5)
alpha ~ dgamma(0.0001, 0.0001)  #Vague hyperprior for alpha
beta ~ dgamma(0.0001, 0.0001)   #Vague hyperprior for beta

#lambda.pred ~ dlnorm(mu, tau)I(0,5)
#mu ~ dflat()
#sigma ~ dunif(0, 10)
#tau <- pow(sigma, -2)

# Calculate Bayesian p-value
chisq.obs <- sum(diff.obs[])
chisq.rep <- sum(diff.rep[])
p.value <- step(chisq.rep - chisq.obs) # Mean of this node should be near 0.5

}

inits
list(alpha = 1, beta = 1)
list(alpha = 0.5, beta = 0.5)

list(mu=-5, sigma=5)
list(mu=-7, sigma=0.5)

data
x[]   time[]
2     15.986
1     16.878
1     18.146
1     18.636
2     18.792
0     18.976
12    18.522
5     19.04
0     18.784
3     18.868
0     19.232
END

list(N=11)
```

Figure F-27. Hyperprior comparison model.

This script was run for truncations from (0,1) to (0,5) to discover the behavior of each of these hyperpriors as more of their complete distribution is used in predicting the rate of the varied population. A chart of the results is presented in Figure F-28.

A few insights from the results:

- The 95th percentile diverges, which shows the effect of the much larger tail of the lognormal.
- The 50th percentile (median) is flat for both, with a small difference between the two. This explains the equally good replication of the data in the Bayesian Chi-squared test.
- The lognormal mean starts correlated with the median in relationship to the Gamma at truncation (0,1), but then it is affected by the tail as the truncation increases, eventually reaching the 88.36 prediction at full use of the distribution.
- The Gamma mean has very little movement between the (0,5) truncation and the use of the full distribution.

The take-away is that even though both hyperpriors "fit" the data in the Bayesian P-value replication of the model, this sort of analysis points out the better of the two priors to use for this particular set of data when used as a PVD for predicting future performance since its mean converges to its full distribution value within an intuitively reasonable numbers of failures per year truncation.

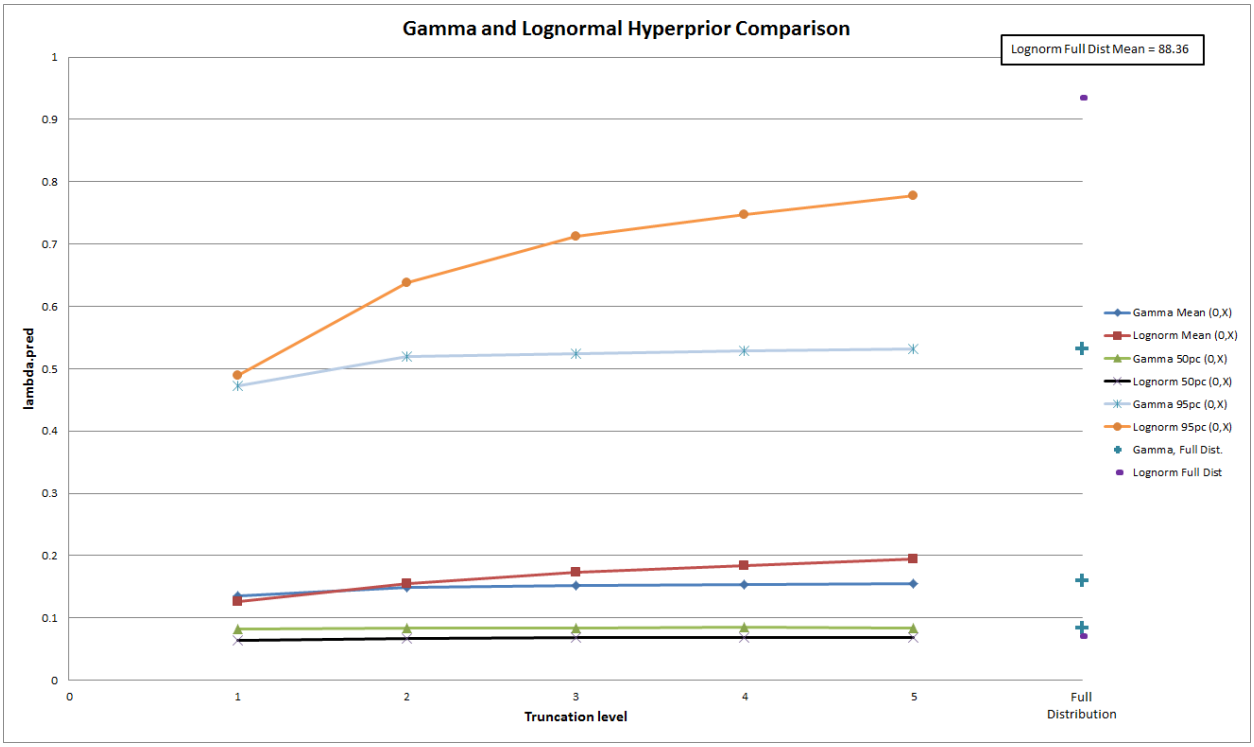


Figure F-28. Hyperpriors comparison.

F-3. PRECISION LIMITATIONS OF NUMERICAL PROGRAMS

Returning to the rate-based example, let us assume that a prediction for hourly failure rates is required, rather than the yearly one. One could simply divide the results by 8760 since that is how many hours there are in a year; however, another analyst might want to perform the MCMC calculations using the yearly data first converted to hourly data. However, beware the precision limits of the MCMC program when performing this sort of analysis (note this limitation holds for many software programs that have to treat small numbers).

The issue here comes from how the mean is attained through the Poisson model:

$$P(x) \sim mean^{x*e(-mean)}$$

$$mean = lambda * t$$

$$P(x) \sim (lambda * t)^{x} e^{(-lambda*t)} \quad \text{(F-11)}$$

As parameter "t" becomes larger it is multiplied by the diffuse Gamma in the model and the numerical precision of the MCMC program comes into play.

A comparative analysis was performed using the Gamma prior with yearly data versus using hourly data. The results of these are presented in Figure F-29. The difference (delta) between the two shows up most significantly in the lower tails of zero failure data sources, where the extremely low values for the 2.5^{th} percentile in the E-08 range indicate an issue with the analysis. However, there is approximately a 100% delta across the board between using yearly data versus hourly data. A look at the hourly lambda.pred of 3.904E-05/h (not in the figure) multiplied by 8760 h/y results in a mean of 0.342/y rate, which is 219% greater than the rate determined by using yearly data.

In cases where yearly data are given and an hourly rate is desired, it is best to use the yearly data and convert the results to hourly for use in PRA. If hourly data are given, be aware of limitations of the MCMC program in use and possibly convert the hourly data to yearly for the analysis.

	Mean	Sdev	2.5th	50th	97.5th
lambda.yr[1]	0.1344	0.08748	0.02053	0.1161	0.3519
lambda.yr[2]	0.07431	0.06359	0.004127	0.05714	0.2402
lambda.yr[3]	0.06973	0.05975	0.00385	0.05367	0.2256
lambda.yr[4]	0.06791	0.05804	0.003755	0.0524	0.2193
lambda.yr[5]	0.1162	0.07555	0.01777	0.1004	0.304
lambda.yr[6]	0.01844	0.03077	2.06E-08	0.00555	0.1079
lambda.yr[7]	0.6123	0.1769	0.3173	0.595	1.006
lambda.yr[8]	0.2594	0.1125	0.08827	0.2432	0.522
lambda.yr[9]	0.01863	0.03106	2.09E-08	0.005649	0.1087
lambda.yr[10]	0.1643	0.08961	0.03811	0.1483	0.3807
lambda.yr[11]	0.01827	0.03052	2.00E-08	0.005531	0.1069
lambda[1]	0.1277	0.07711	0.02379	0.1125	0.3181
lambda[2]	0.07972	0.05963	0.006915	0.06604	0.2309
lambda[3]	0.07552	0.05657	0.006465	0.06256	0.2186
lambda[4]	0.0741	0.05554	0.006303	0.06132	0.2142
lambda[5]	0.1133	0.06814	0.0209	0.1003	0.2811
lambda[6]	0.03369	0.03913	1.62E-05	0.02023	0.1399
lambda[7]	0.5165	0.1663	0.2443	0.4986	0.8912
lambda[8]	0.2303	0.0995	0.08144	0.2149	0.4653
lambda[9]	0.03387	0.03928	1.62E-05	0.02027	0.1406
lambda[10]	0.1526	0.07968	0.03928	0.1387	0.3452
lambda[11]	0.03335	0.03858	1.61E-05	0.02006	0.1375
Delta 1	95%	88%	116%	97%	90%
Delta 2	107%	94%	168%	116%	96%
Delta 3	108%	95%	168%	117%	97%
Delta 4	109%	96%	168%	117%	98%
Delta 5	98%	90%	118%	100%	92%
Delta 6	183%	127%	78485%	365%	130%
Delta 7	84%	94%	77%	84%	89%
Delta 8	89%	88%	92%	88%	89%
Delta 9	182%	126%	77544%	359%	129%
Delta 10	93%	89%	103%	94%	91%
Delta 11	183%	126%	80600%	363%	129%

Figure F-29. Comparison of using yearly data versus hourly data in Poisson model.

F-4. REFERENCES

F-1. Kaplan, S., 1983, On a "two-stage" *Bayesian Procedure for Determining Failure rates From Experimental Data*, IEEE Transactions on Power Apparatus and Systems, PAS-102, 1983.

F-2. Dezfuli, H., D. Kelly, C. Smith, K. Vedros, and W. Galyean, 2009, *Bayesian Inference for Probabilistic Risk and Reliability Analysis*, NASA/SP-2009-569, June 2009.

F-3. Modarres, C., E. Droguett, and M. Fuge, 2016, *A Novel Clustering Based Methodology for Overcoming Heterogeneous Populations for Reliability Prediction*, Wiley-Manuscripts, 2016.

F-4. Ester, M., H. P Kriegel, J. Sander, and X. Xu, 1996, *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, Proceedings of Second International Conference on Knowledge Discovery and Data Mining (KDD-96), AAAI Press, 1996.

F-5. Kelly, D. and C. Smith, 2011, *Bayesian Inference for Probabilistic Risk Assessment- A Practitioners Guidebook*, Springer, 2011.

F-6. Kelly, D. and C. Atwood, 2008, *Bayesian Modeling of Population Variability – Practical Guidance and Pitfalls*, PSAM-9, INL/CON-08-14208, May 2008.