Bayesian Nonparametric System Reliability Estimation using the Beta-Stacy Process

Jackson T. Curtis

A selected Project submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Dr. Richard Warr, Chair
Dr. Scott Grimshaw
Dr. John Lawson

Department of Statistics

Brigham Young University

April 2019

ABSTRACT

Bayesian Nonparametric System Reliability Estimation using the Beta-Stacy Process

Jackson T. Curtis
Department of Statistics, BYU
Master of Science

This project explores a method for fast, flexible estimation of reliability for systems of many components. The beta-Stacy process is used as a conjugate nonparametric method to model an unknown CDF. Methods for combining multiple sources of data from reliability tests are explained.

An R package is introduced that implements the calculation needed to fit this model. Several simulation studies are performed to validate assumptions and assess strengths and weaknesses of the method.

Keywords: Dirichlet process, hierarchical

ACKNOWLEDGMENTS

This research builds extensively off previous research by Dr. Richard Warr and associates. Dr. Warr and I have worked closely over the last year and a half on the theory presented here, and his guidance has been invaluable in completing this project.

Additionally, Dr. Berrett, Dr. Dahl, and Dr. Reese all taught classes that contributed greatly to my understanding of the Bayesian and reliability concepts used in this project.

Finally, my education has been greatly enhanced by all the great students and faculty in the BYU statistics program.

CONTENTS

CHAPTER 1

BACKGROUND

1.1 Introduction

This project provides an estimation technique for system reliability of large systems. As systems become increasingly complex and test data on individual components within the system become more available, traditional methods such as choosing a parametric model, validating each assumption of that model, and, in the case of Bayesian models, performing MCMC to get parameter estimates become unwieldy as the number of components making up the whole system grows. In addition, the realities of real-world testing usually make it relatively cheap to test individual components but extremely costly to test entire systems. Thus, our method will utilize as much information from the small tests as possible. Our method provides a computationally efficient way to combine these types of tests and provide accurate predictions and uncertainty estimates of reliability, while making minimal assumptions about the model form.

1.2 System Reliability

The primary goal of system reliability is to estimate the probability that the system will be working as designed as a function of time. A system can be made up of many smaller subsystems and individual components. For example, an airplane relies on its fuel system, electrical system, and landing gear, all of which are made up of even smaller components such as intake valves, circuit boards, and tires. Our method will combine information from tests on these small components, and use subsystem and whole system tests to update reliability estimates at each level of the hierarchy.

Two main relationships will be used to combine reliability information. The first is a redundant relationship, where the system will continue to function as long as one of two components continues to function. This is referred to as components in parallel. The second is a dependent relationship, where if either component fails the system will fail. This is referred to as components in series. The CDF for random variable describing a system's time to failure when it is made up of two independent components in parallel (X_1 and X_2) is given by:

$$F_S(t) = P(X_1 \leq t \text{ and } X_2 \leq t) = P(X_1 \leq t)P(X_2 \leq t) = F_{X_1}(t)F_{X_2}(t) \tag{1.1}$$

Similarly, a system made up of components in series can be calculated as:

$$\begin{gathered} F_S(t) = P(X_1 \leq t \text{ or } X_2 \leq t) = 1 - P(X_1 > t \text{ and } X_2 > t) = \\ 1 - P(X_1 > t)P(X_2 > t) = 1 - R_{X_1}(t)R_{X_2}(t) \end{gathered} \tag{1.2}$$

where $R_x(t)$ is the reliability function $(1 - F_x(t))$.

We will demonstrate how individual component reliabilities can be combined using these equations to produce system estimates of reliability, which can be supplemented with data from system tests, while properly accounting for uncertainty in our estimates at each point.

1.3 The Beta-Stacy Process

Our method for estimating reliability utilizes a Bayesian nonparametric process called the beta-Stacy process. We aim to estimate a failure time CDF, F(t). Instead of choosing a parametric family and estimating the parameters (which may have undesirable consequences when we know little about the data generating process), the nonparametric approach puts a probability distribution over all valid CDFs. Frequentist nonparametric approaches typically use the empirical distribution, and the Kaplan-Meier estimator is a common estimator when

dealing with failure times and right censored data (Kaplan and Meier 1958). To approach the problem from a Bayesian perspective, we can set a prior distribution over all possible CDFs and get a posterior distribution over CDFs after data is observed. Benefits of the Bayesian approach include the ability to incorporate expert prior knowledge in the system, proper accounting of uncertainty estimates, and quick computation through the use of conjugate models.

Previous work by Warr and Collins (2014) established a model using the Dirichlet process to model these unknown CDFs. The Dirichlet process (DP) model can be written as:

$$F(t) \sim DP(\alpha, G(t)) \tag{1.3}$$

The DP is controlled by two parameters, the precision parameter α which determines the certainty of our estimate, and the centering measure, G(t), which is a valid CDF and controls the mean of our estimate as a function of time. At any time point, the probability distribution for the probability that the component has failed by that time is distributed Beta$(\alpha G(t), \alpha(1 - G(t))$. A key benefit of the Dirichlet process is conjugacy. That is, if we set a prior distribution over CDFs using the DP, after updating our model with observed data the posterior is also a DP.

Our current research will extend this model to address some key issues. The main issue is that the DP model loses its conjugate properties in the presence of censored data. Censored data occurs in reliability whenever the item did not fail by the time the measurement period ended and is extremely common. By replacing the Dirichlet process with the beta-Stacy process (Walker and Muliere 1997), of which the DP is a special case, we can address the problem of right censored data. The main practical difference is that the α parameter is now a function of time, similar to the centering measure G(t). This is necessary because as points are censored, we become more uncertain of the rate of failure, so our precision will drop after each censored point.

Because a DP is a beta-Stacy process (BSP) with constant precision, the Dirichlet process will be used as a prior in our models. Although this assumption could be relaxed and a BSP with non-constant precision could be used as a prior, using DP's often makes sense unless your uncertainty in the value of F(t) changes with t. If a DP prior is used and no censoring occurs in the observed data, the BSP posterior will have constant precision and will reduce to the DP.

In order to simplify the calculations used in the model, one constraint that will be imposed is that the centering measure, G(t), must be the CDF of a discrete, positive-valued random variable. This will allow us to perform the calculations using only the finite number of jumps in the centering measure. A user can still imagine a continuous CDF to represent his prior belief, but will discretize the continuous distribution. As the number of jumps in the discretization increases, the difference between the discrete approximation and the continuous curve will be negligible, but will provide rapid improvements in computational efficiency. The precision function, $\alpha(t)$, will also be limited to have a finite number of discrete jumps.

1.4 Hierarchical Model

In order to make inferences about the system as a whole, we need to combine data from the component tests with data from the system tests. Our general strategy is to estimate the CDF of components in series or parallel, and then use the posterior estimates of the reliability as our prior when analyzing the data from the system-wide tests. This allows us to use either non-informative or informative priors at the component level and to have well-informed priors at the higher levels that are informed by the data from the component level.

One problem that will be addressed is how to maintain conjugacy when assessing the reliability of components in series or parallel. Although the CDF of each component individually will have a beta-Stacy process posterior, taking the min or max of the components'

CDFs as described in (1.1) and (1.2) results in a posterior distribution that is not a BSP. However, it seems reasonable that a BSP would provide a good approximation to the CDF of the min or max. We can calculate the first and second moments of the true distribution at each time t and find a BSP to use as an approximation that has those same moments. A key piece of the simulation will be evaluating the effectiveness of this approximation.

1.5 Posterior Computation

Here we provide the equations necessary to compute the posterior beta-Stacy process given a beta-Stacy prior and (possibly right-censored) data. Let

$$M(t) = \sum_{i=1}^{m} I(T_i \geq t), \text{ (number of units still functioning just before time } t)$$

$$J(t) = \sum_{i=1}^{m} C_i I(T_i = t), \text{ (number of failures that occurred at time } t).$$

where C_i is an indicator variable with the value of 0 if T_i was censored.

Then, given the prior

$$F(t) \sim BSP(\alpha(t), G(t))$$

the posterior (where T represents our observed data) is

$$F(t)|T \sim BSP(\alpha^*(t), G^*(t))$$

where the new centering measure is

$$G^*(t) = 1 - \prod_{i=1}^{m} 1 - \frac{\alpha(t_i)(G(t_i) - G(t_i-)) + J(t_i)}{\alpha(t_i)(1 - G(t_i-)) + M(t_i)} \tag{1.4}$$

and new precision parameter

$$\alpha^*(t) = \frac{\alpha(t)(1 - G(t)) + M(t) - J(t)}{1 - G^*(t)}. \tag{1.5}$$

where $t_1 < t_2 < ... < t_n$ is each point where G(t) or $\alpha(t)$ is discontinuous and m is the integer such that $t_m \leq t < t_{m+1}$. $G(t_i)$ is the prior base measure evaluated at t_i, $G(t_i-)$ is the prior base measure evaluated in the limit as G approaches t_i from the left.

Careful examination shows that in the absence of prior information ($\alpha(t) = 0$), the base measure is equivalent to the Kaplan-Meier estimate. Likewise, with constant prior precision and no censoring, the precision will be constant so that the BSP can be written as a Dirichlet process.

1.6 Sampling CDFs from a BSP

A draw from a beta-Stacy process, as we've defined it, will be a valid, discrete CDF with jumps wherever a jump in the BSP centering measure occurs. Each jump, conditional on all previous jumps, has a beta-Stacy distribution (Walker and Muliere 1997). Practically, we can use the following steps to obtain a CDF draw:

1. For t_1 (the first jump in the centering measure), set F(t_1) to a draw from a Beta($\alpha(t_1)G(t_1)$, $\alpha(t_1)(1 - G(t_1))$) distribution

2. For each t_i, i = 2, 3,...n (number of jumps in BSP)

    a) Set F(t_i) = F(t_{i-1}) + x * (1-F(t_{i-1})) where x is a draw from a Beta($\alpha(t_i)(G(t_i) - G(t_{i-1}))$, $\alpha(t_i)(1 - G(t_i))$)

    The resulting F(t) is a discrete CDF sampled from the BSP.

CHAPTER 2

OBJECTIVES

The main goals of this project will be to create an R package that makes implementing these models easy and fast and use this package to validate the theory regarding the method and assess its strengths and weaknesses.

2.1 R Package

We desire to make our model easy to implement through the use of a package in R. One of the main advantages of our model is that it replaces complex MCMC with conjugate models, however, performing the algebraic operations to obtain the posterior requires tedious and careful coding. We will make the model more accessible by providing functions that do not require the user to implement the formulas themselves.

The main core of our package will be a beta-Stacy process object and associated methods. Users will be able to specify their prior by creating one of these objects. Another method will be implemented to take in a set of (possibly censored) data and a prior, and return a new posterior BSP object. Additional methods will be implemented to calculate moments, graph, print, and evaluate the precision at different time points. Finally, methods will be implemented to sample a BSP. Each sample from a BSP object returns a valid CDF and draws from the CDF can be used to simulate and visualize the process.

The success of the R package will be judged on its ease of use, speed, and ability to model components with many subsystems and components. Testing should ensure that a reasonable time limit for the computations is achieved as both the amount of data is increased and the number of components is increased. In addition, thorough documentation should allow the user to easily understand how to provide their data in the required format

and use the functions provided. The code will be made available on Github to be installed locally by the user.

2.2 Simulation

We have identified several main questions we would like to explore through simulation. First, can we provide valid credible intervals for the probability of failure at given times? Second, are the methods unbiased at estimating the probability of failure? Third, when is the approximation of a parallel or series system valid and when does it fail? We will describe these in more detail below.

Asymptotic Consistency

The first question is a basic demonstration that our method does what we claim it does. That is, given a complex system with many components, if we gather data at all levels and merge them into an estimate of system reliability, will we be estimating the system's true reliability? We can test this by designing a system and then estimating the bias between the actual probability of failure and the predicted probability of failure from the beta-Stacy posterior. We can also look at the coverage of our confidence intervals to see if we are accurately quantifying our uncertainty in the probability of failure.

BSP Approximation

The goal of the second simulation will be to justify the use of approximating a series or parallel subsystem with a beta-Stacy process. As noted earlier, if two components both have BSP posteriors, when they are merged in series or parallel, the merged process is not a BSP. Instead, we will approximate it with a BSP by finding the first and second moments at each time point of the true distribution, and then find a BSP with those same moments. The simulation will compare these true and approximated processes, looking for times when the approximation breaks down. Specifically, we will look at differences between series and

parallel, cases where the two distributions are at their extremes ($F_1(t) \approx 1$ and $F_2(t) \approx 0$), and cases where the posterior precision is radically different for the two components. From this simulation, we will create recommendations for when this approximation can be used and when it is likely to produce poor results.

Simulation Follow-Up Questions

After these initial two simulations, several problems presented themselves that required further exploration and refining. In the first follow-up, we investigate why coverage of our credible interval does not meet nominal coverage, even in the most simple cases by comparing it to simpler Bayesian models.

We also investigate several modifications of our original experiment. We explore why some of the problems were seen and how to avoid them in practice, as well as what we can expect when the number of components gets much larger than the small system we used initially.

CHAPTER 3

R PACKAGE

We created the R package, BnpSysRel (Bayesian Nonparametric System Reliability) to implement our methods. The package is publicly available on Github. The main contributions of this package are (1) the creation of a class representing a beta-Stacy process and associated generic functions for that class, (2) functions to combine prior information and data to compute posterior estimates of a beta-Stacy process, and (3) a simple way for users to describe and obtain reliability estimates of complex systems. Each is briefly discussed below.

Beta-Stacy Class

A large part of the package is devoted to creating a class that enables easy manipulation of the beta-Stacy process. At its core, the process, as defined in Warr and Greenwell (2014), is a collection of discrete jumps in the centering measure as a function of time, along with precision at each point in time. Thus, we represented our process as a list with three components: a vector for the support (a list of ordered time points where each jump occurred), a vector of centering measures, and a vector of precision values. Care had to be taken to properly represent the centering measure and support, as the centering measure is a step-wise function of time where each point is right-continuous but the precision is a step-wise function of time that is left-continuous when censoring occurs. Functions were implemented for evaluating the centering measure and precision at arbitrary points to ensure these bounds were respected.

To enable users to use this class, a creation method was implemented, which returns a betaStaceyProcess object. This will typically be used to create priors to represent their prior knowledge. Once these objects are created, generic methods can be used to examine

them. The function print() will transform the data to a dataframe for easy inspection. Plot() will plot the centering measure against the support from the betaStaceyProcess, as well as approximate credible intervals for the value of the unknown CDF which the BSP is modeling. Quantile() will provide quantiles for the expected time to failure.

Bayesian Computation

The intended use of the package and classes is to allow the end user to directly model an unknown CDF in a nonparametric Bayesian way. The framework previously explained in the background section allows a BSP prior object to be supplemented with fully-observed and right-censored data and result in a posterior process that is also a BSP. Thus, our package includes a function to perform these computations and returns the resulting BSP object. The inputs to the function are a BSP representing the prior information, created with the function described in the last section, and a matrix of data, the first column specifying the failure times and the second column indicating whether the observation was censored or not.

Since one of the major benefits of our modeling choices is the ability to get exact posteriors without the need for MCMC, an important aspect of this function is to keep the computational complexity low enough to be applicable to large data sets. Testing shows that our method can compute posteriors for data sets of 10,000 measurements in under three seconds.

Reliability Systems

One of the important innovations for our package is a syntax to easily specify the possible relationships in a reliability diagram. For systems with a hundred or more components, a huge amount of code would be required for the user to find the posterior for each component, merge the component posteriors to get priors for subsystems, and repeat the process until system-wide reliability could be estimated. Instead, we replace this with a syntax to specify these relationships and a function that will take the diagram, priors, and data and calculate

```
S(valve1, valve2, valve3):plumbing
P(generator1, generator2):power
S(plumbing, power):system
```

Figure 3.1: Users will produce a text file indicating how all the components and subsystems relate to one another.

the posteriors. Figure 3.1 shows the simple syntax which users can use to indicate the relationships within the system. Users specify which components are in parallel and which are in series, and provide names that are used as locations to find the associated prior and data. Complete rules for how to specify the diagram can be found in the documentation.

Using this syntax, we implemented a function that will compute the posterior reliability for all the named pieces of the system. The inputs to this function are (1) a file location where the reliability diagram is specified in a text file, (2) a named list of BSP objects to be used as priors for each component in the first level of the hierarchy, and (3) a named list of data matrices for each component on which data was collected. Given these inputs, the function will do all the heavy lifting of merging and calculating all the posteriors necessary which will greatly reduce the opportunity for users to introduce errors while coding. The return of the function will be a named list of BSP objects similar to the provided prior list, but each element representing the posterior BSP.

Documentation

An important part of any package is its documentation. All functions of the package are documented using the roxygen2 package in R. This documentation includes explanations, inputs, outputs, and examples. Additionally, a vignette is provided that walks through how a basic analysis would proceed for a system made up of bicycle parts. The functions to do the analysis and examine the results are demonstrated. This vignette is included as an appendix.

CHAPTER 4

SIMULATION STUDY

The goals of the simulation study will be to assess the performance of our method of estimation in several realistic scenarios.

Initial Simulation

The first simulation will examine the properties of our method of estimation under normal conditions. Questions we will answer with this simulation include: (1) Does our method provide unbiased estimates of reliability and accurate confidence intervals with nominal coverage for individual components? (2) Does the method of finding an approximate BSP with the correct moments to use as a subsystem prior still provide unbiased estimates and proper confidence intervals?

Figure 4.1 demonstrates the system constructed for the simulation. The true reliability of the valve will follow a Weibull(4, 20) distribution while the generator will follow a chi-squared(3) distribution. The distributions for plumbing, power, and the whole system

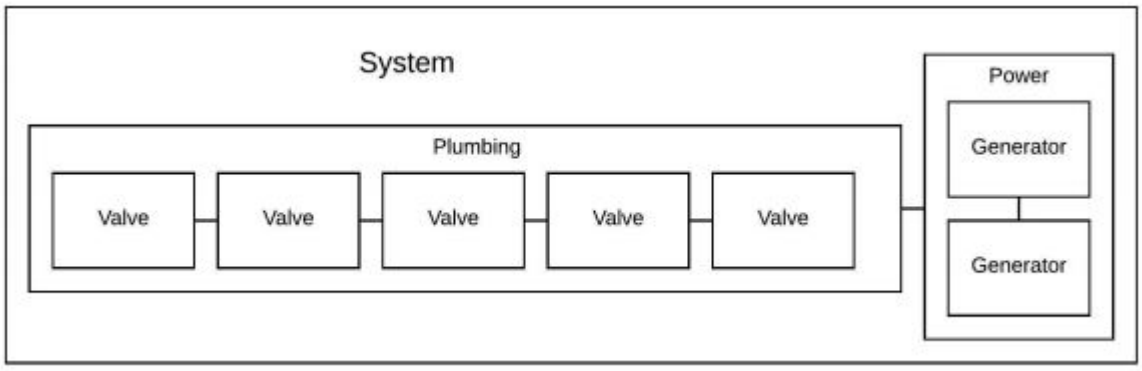


Figure 4.1: Reliability diagram for the system used in Simulation 1. Data was gathered at each of the valve, generator, plumbing, power, and system levels

are implied by taking the minimums and maximums of the valve and generator. For the priors we will set weakly informative but unbiased priors by setting the prior centering measures to the true values but the prior precision to 0.2, equivalent to 1/5th of one observation.

We will vary several factors during the simulation to see how these factors change the results. First, we will vary the amount of reliability measurements we've gathered for each of the pieces of the system. We will experiment with n=10, 50, and 500. When n=10 we will gather 10 failure times for each of the valve, generator, plumbing system, power supply, and system. Next, we will vary whether censoring was present in the data. In cases with censoring, we will generate n random (uniform) times between the 70th and 99th quantile of the distribution (we cut off at 99 because most of these distribution have infinite support). We will compare this randomly generated time to the generated failure time and if it is smaller we will mark that observation as censored and record the censoring time instead of the actual failure time. This results in approximately 10% of our data being censored, with censoring times occurring only in the upper tail of the distribution. Finally, we will measure performance in the right tails, middle, and left tails (20th, 50th, and 80th percentile of each distribution) to see if accuracy is maintained, especially with the upper tail in the presence of censoring.

The first metric we will look at is coverage of the 95% credible intervals for the true probability of failure at specific times.

Each scenario was run for 5,000 iterations and the coverage results are shown in Figure 4.2. Several problems are immediately noticeable. The first thing that is troubling is that even in our simple components (valves, generators), which should be a straight-forward, conjugate calculation, the coverage is too low. For n=10 coverage is less than 90% and when n=50 coverage is still not quite 95%. Additionally, for the more complex subsystems the coverage is significantly lower than expected. The differences between cases with and without censoring look slight however, suggesting that the beta-Stacy model is appropriately compensating for the censoring.

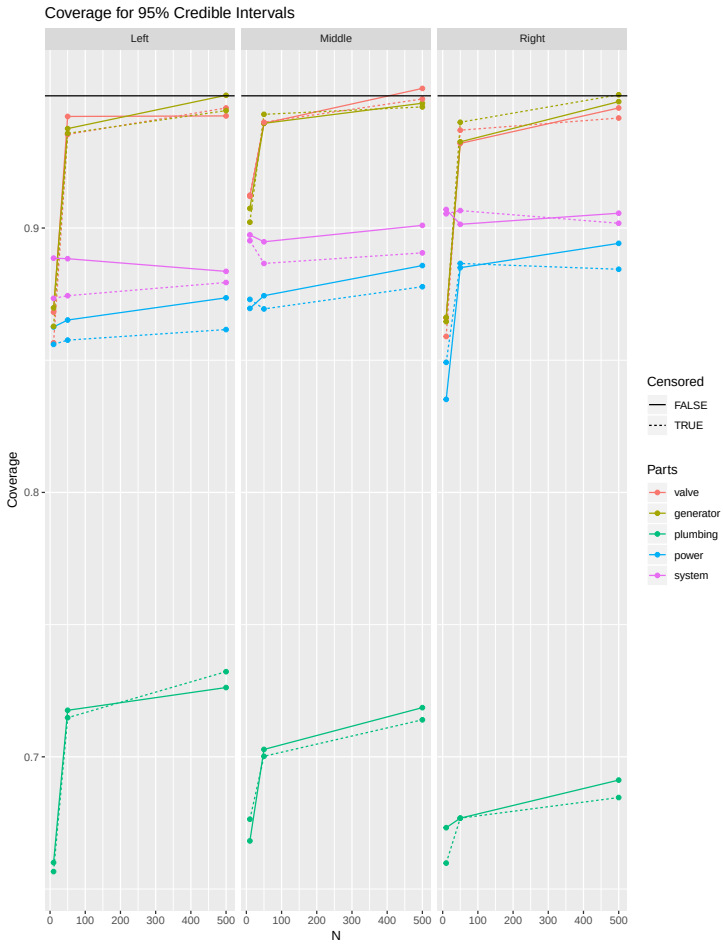


Figure 4.2: 5,000 simulated confidence intervals were produced and coverage was measured for each combination of the parameters. The three graphs represent results from the left, center, and right tails of the true distribution.

4.1 Credible Interval Coverage Problems

Our first simulation indicated that even in the most simple scenarios (single component, no censoring) we were not achieving nominal coverage of our credible intervals. When n=10, the intervals had less than 90% coverage and when n=50 the coverage was only between 93% and 94%. For a single component with no censoring, our posterior is a Dirichlet process. The Dirichlet process has a simple, closed-form distribution for the probability that a component has failed at a given time:

$$F(t)|\hat{F}(t), t \sim \text{Beta}(\alpha F_0(t) + n\hat{F}(t), \alpha(1 - F_0(t)) + n(1 - \hat{F}(t))) \tag{4.1}$$

where α is the prior precision, F_0 is the prior centering measure, and $\hat{F}$ is the empirical CDF of the data. You can obtain this exact same posterior distribution by considering an experiment done at time t, analyzed with a beta prior on the probability and a binomial likelihood, where items that have failed before t are counted as successes and items that have not failed at t are failures. In this scenario, α represents our prior sample size, where $100 * F_0(t)\%$ of the prior sample were "successes" (items failed before t) and $100 * (1 - F_0(t))\%$ were prior sample "failures" (items that lasted beyond t). Because of this relationship, any coverage problems we have in this most simple (Dirichlet) case should also be present for simulations of the coverage of a credible interval on the success probability in the binomial experiment case. Therefore, studying the beta-binomial conjugate model should highlight when and why we are likely to have problems.

We investigated the true coverage of the beta-binomial model for various values of the true success probability, the sample size, and the prior used. Two major issues were identified, which are demonstrated in Figures 4.3 and 4.4. The first is that when a Beta(0,0) improper prior is used, the coverage is highly variable with the sample size. The coverage is frequently under, but occasionally over the nominal amount, although this effect lessens as sample size increases. Priors tend to smooth out coverage when varying the sample size, but result in artificially inflating or deflating the coverage based on how close the prior is to the true probability.

Figure 4.4 is a dramatic representation of the other problem identified. We set a semi-informative, highly accurate prior, Beta(0.4, 4) to estimate the true success probability of 9%. Small sample sizes result in significant over-coverage, which is not surprising due to our highly accurate prior, but we see a sharp drop in coverage at N=20. When N=19, and 0 successes are observed (which happens 16.7% of the time), the 95% credible interval is (0.000, 0.093), which includes the true value of 0.09. However, when N=20 and 0 successes are observed (which happens 15.2% of the time) the credible interval is (0.0000, 0.0896), which no longer includes the truth. This corresponds with the 15% drop in coverage that

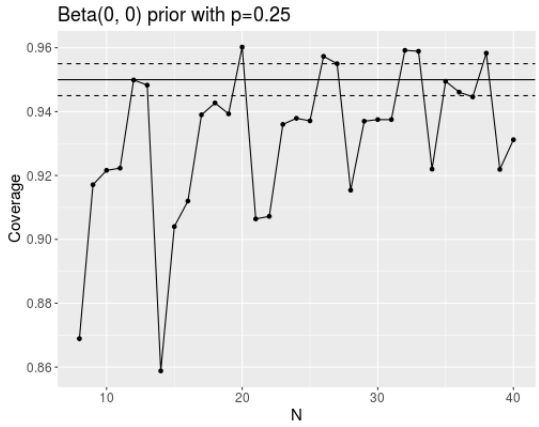


Figure 4.3: Coverage for various sample sizes when Beta(0,0) is used and p=25%

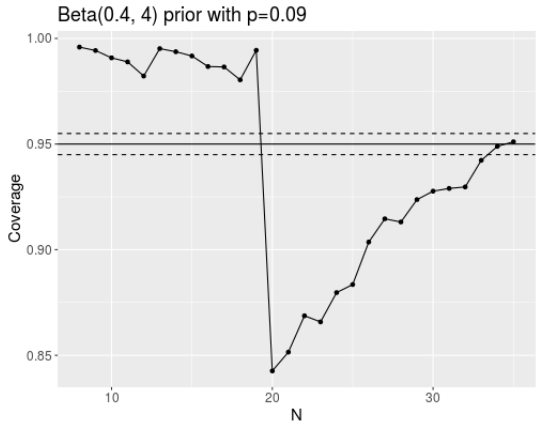


Figure 4.4: Coverage for various sample sizes if the true probability of success is 9% and a Beta(0.4, 4) prior is used.

we witness in the plot. While not always this dramatic, the discrete nature of the binomial distribution creates jumps in the coverage that are particularly dramatic for small N but lessen as the sample size increases.

4.2 Multi-component Subsystem Coverage Problems

The second problem clearly seen from Figure 4.2 is that the coverage is radically worse for systems made of multiple components. This problem is more easily explained as a flaw in reasoning. For the case when we have two i.i.d. components (such as the generators in Figure 4.1), in our nonparameteric setup we assume $X \sim F_1$ and $Y \sim F_1$ and that

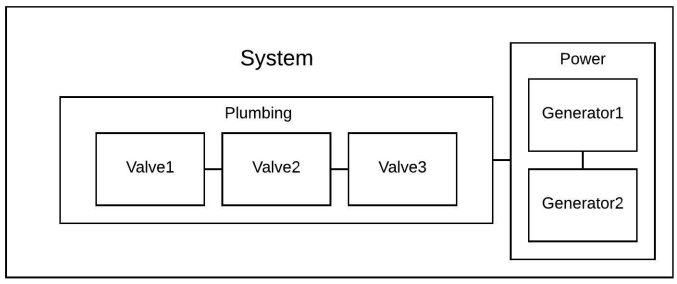


Figure 4.5: Modified reliability diagram for Simulation 2.

F_1 is random with a distribution. When we combine components in series or parallel, we assume each component is independent of each other. However, after we condition on the data, while X and Y are independent of each other, our posterior distributions of X and Y, calculated as $F_1|T = t$ are no longer independent because they condition on the same data. Essentially this amounts to a double counting of the data, which makes the credible intervals too narrow to capture the true uncertainty. We verify this theory by doing a modified version of the first simulation, one in which all components are unique and have their own data collected independent of the others. Figure 4.5 shows the new setup. Because we are adding complexity by generating data for each part independently, we limited the computation time by reducing the number of valves, changing the largest n to be 250 instead of 500, and running for 3,000 simulations instead of 5,000.

The results shown in Figure 4.6 show a marked improvement over the first simulation. The performance of the subsystems of multiple components is now almost indistinguishable from the individual components. Components approach nominal coverage much better overall. In addition, places where the coverage falls short lines up almost exactly with what we would expect from the beta-binomial coverage problems demonstrated in the last section. For example, if we use the set up from the last problem to examine what our beta-binomial coverage would be if n=10, the true probability of success was 20%, and the prior on the proportion was Beta(0.1, 0.1) (similar to the 0.2 precision we used in Simulation 2), then the coverage we would expect to see is 86% ($\pm$0.5%). This agrees almost perfectly with the

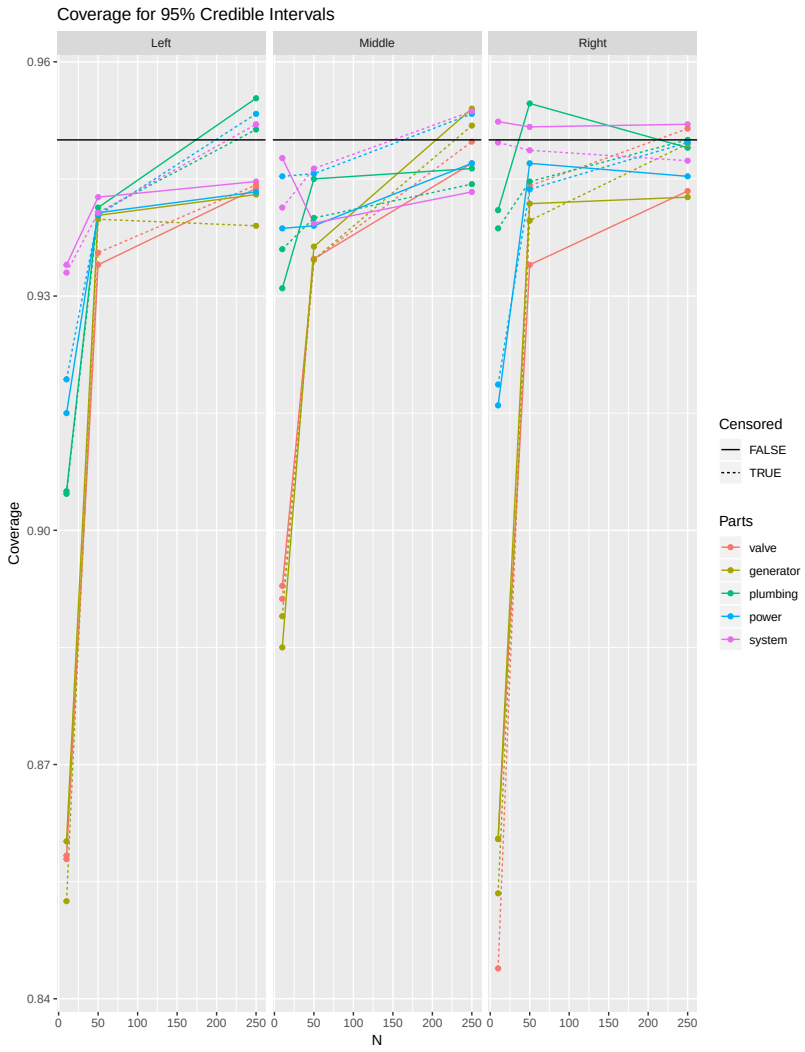


Figure 4.6: Coverage results for Simulation 2.

coverage we see in the simulation for the two individual components. Likewise if we raise the probability of success to 50%, the coverage jumps 89%, which again matches what we see in the middle section of the graph. In cases where strong prior information is provided, naturally the coverage will vary with the accuracy of the prior, but it is reassuring to note that in cases with very limited prior information, we can provide an accurate estimate of the coverage by analyzing a comparable beta-binomial Bayesian experiment.

Figure 4.7 shows bias as another measure of accuracy for our estimation method. To calculate these we took the expected value of the BSP at each of the three time points

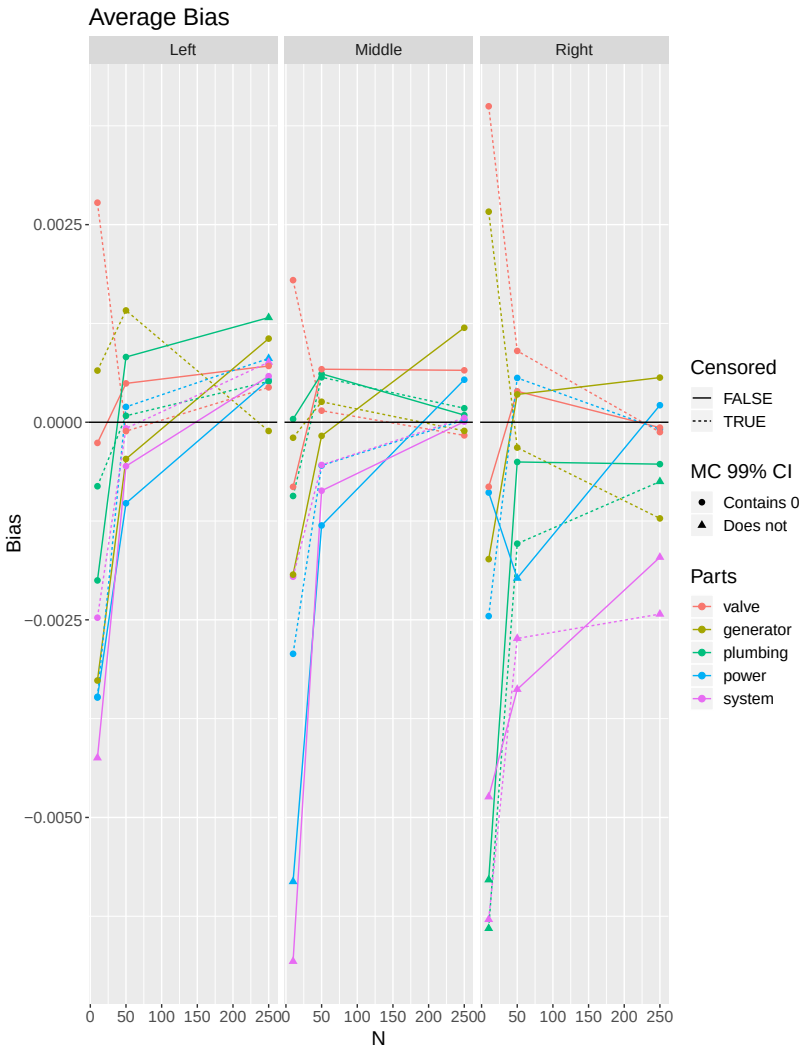


Figure 4.7: Plot showing average bias for probability estimates, with indicator of whether it is within Monte Carlo error of 0

and compared them to the true failure rate of the component and averaged over all the simulations. The first things to notice is that the y-axis is quite small, and many of the estimates of bias contain 0 within the Monte Carlo confidence interval of the error. The more complex components (power, system, plumbing) tended to be more biased for small n. For large n (n=250) the left tail measurements tended to be overestimating the probability of failure while the right tails generally underestimated the probability of failure. Inferences in the middle seemed largely unbiased.

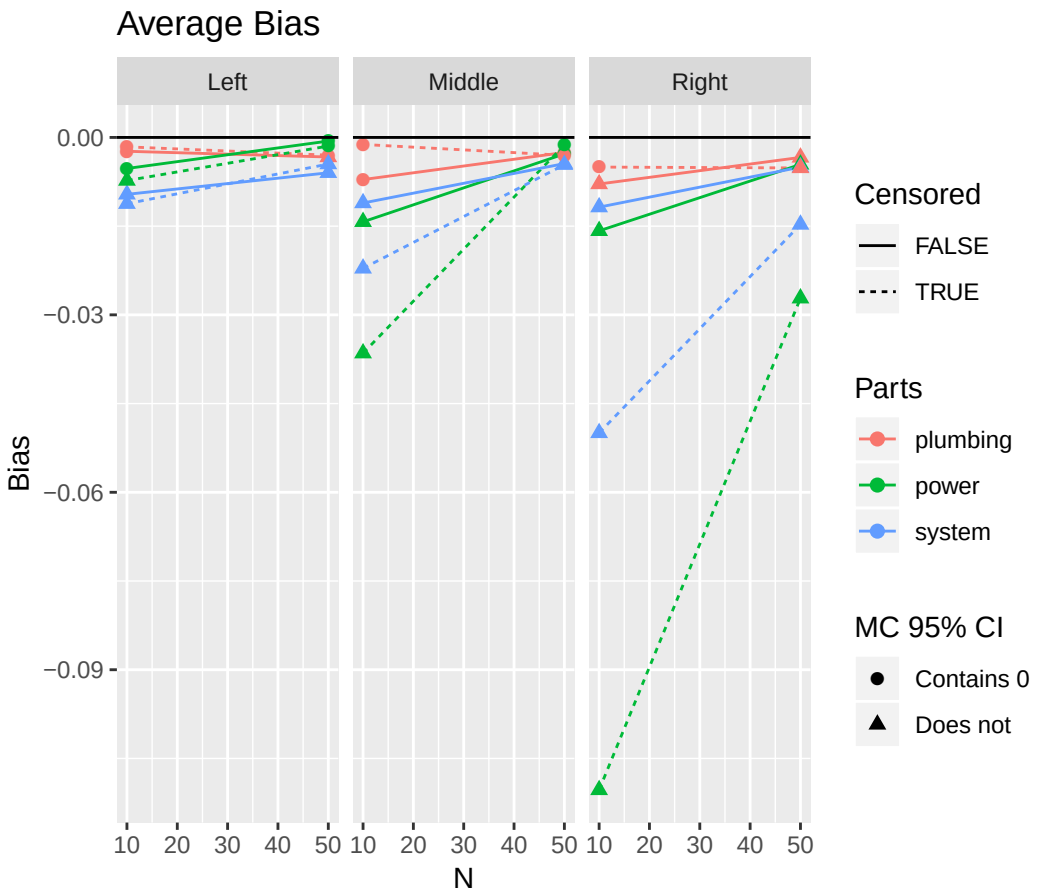


Figure 4.8: Bias results for the plumbing, power, and system shown.

4.3 Many Component Simulation

The results in Figure 4.7 seem suggestive that the bias is worse for the more complex pieces (e.g. the whole system). We wanted to do one more test to see if dramatically increasing the number of components would dramatically increase the bias. This led to a few interesting results.

To set up this simulation we simply extended the original simulation study. In this study, the plumbing subsystem is now made up of ten valves in series instead of three, and the power subsystem is now ten generators in parallel instead of two. Since the worst behavior seems to be for small n, we considered only n=10 and n=50.

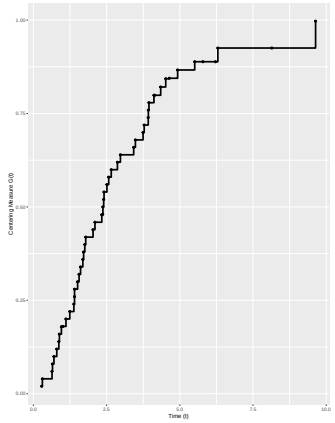


Figure 4.9: A BSP with censoring in right tail

The result, as shown in Figure 4.8, that the censored, parallel system does much worse than the others is surprising, but can be explained by some of the phenomena that has been observed with the BSP theory. The first thing to note is that there is no jump in the centering measure when censoring occurs, just a change in the precision (as opposed to a fully observed observation, which does increase the centering measure). The second thing to note is that if we evaluate a BSP at a time where a jump does not occur, its centering measure at that time is assumed to be the same as the jump time immediately before it. Figure 4.9 gives an example where because the test components with the longest survival are censored, the centering measure does not continue the smooth curve upward that you would expect.

Because censoring creates these long flat tails in areas with little fully-observed data and little prior information, there is a tendency to underestimate the failure rate. In a parallel system, the system lasts longer than any of the individual components, so its distribution is right-shifted compared to the component. This means if there are many little, flat portions among the components, they are likely to effect the parallel system in the mid-to-right tail substantially. This is much less problematic for series systems which left shift their results, meaning far-right censoring would be shifted into the extreme right tails.

4.4 Approximation Accuracy

Another goal of our simulations was to assess the accuracy of approximating two components' posteriors in series or parallel as a single BSP, to be used as the prior for the subsystem. As noted earlier, merging two BSP posteriors in series or parallel does not produce a BSP. In order to maintain conjugacy we fit a BSP by finding the first and second moments of the merged posteriors and finding a BSP with those same moments (this is not always possible, see Limitations and Future Work). This is a source of potential bias, but results from the second simulation show that the effects appear to be small, and our credible intervals maintain expected coverage. In this section, we visually assess the differences between the BSP approximation and the distribution we get at select points from sampling from the two posterior BSPs and creating the true distribution.

A sample from a BSP (which we can obtain using the sampling function in the package, bspSampling()) is a single, discrete CDF which we can evaluate at a certain time, t, to look at the marginal distribution at time t. In order to determine what the real distribution is, we will sample from from the component posteriors at a given time, and merge them using the formulas in Equations 1.1 and 1.2 for parallel and series systems, respectively. We can compare these merged draws with the draws obtained by the approximate BSP with the correct moments to look for any serious problems with the approximation.

We demonstrate this by building on our basic setup of the last simulation. We will create two posteriors based on the valve components, each with n=10 measurements and identical priors. We can then calculate the BSP approximation and the actual distribution at a given point for both a parallel and series system (for the series system the point is t=13.7 and for parallel, t=18.2). Figures 4.10 and 4.11 show the marginal distributions for the approximate and exact distributions. While not identical, the approximation is clearly catching the major features that we would like.

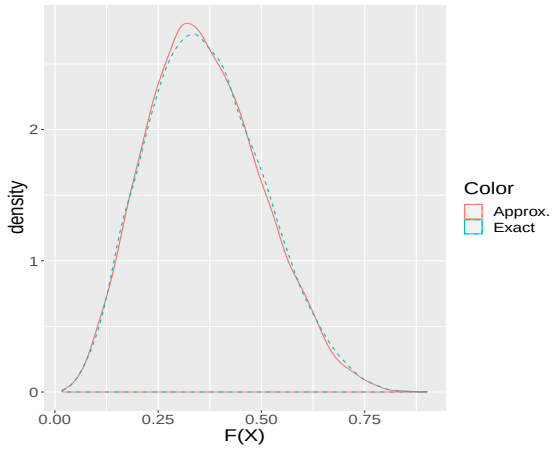


Figure 4.10: Marginal distribution of F(T) at t=13.7 when 2 valves are in series

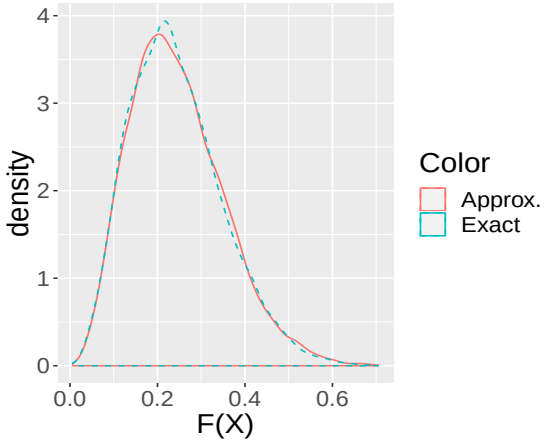


Figure 4.11: 2 valves in parallel, t=18.2

A more extreme edge case that we can look at is when all of the observations for one of the components are censored. This results of this is that marginal draws at a certain time are almost all very close to one or zero, as in Figure 4.12. When combining this with another component, this results in a surprising, bimodal distribution, where the first mode is where the zeroes were sampled and the second mode is where the ones were sampled. Even in this extreme case, Figure 4.13 shows that the moment-matching BSP does a good job capturing the features of even this extreme distribution.

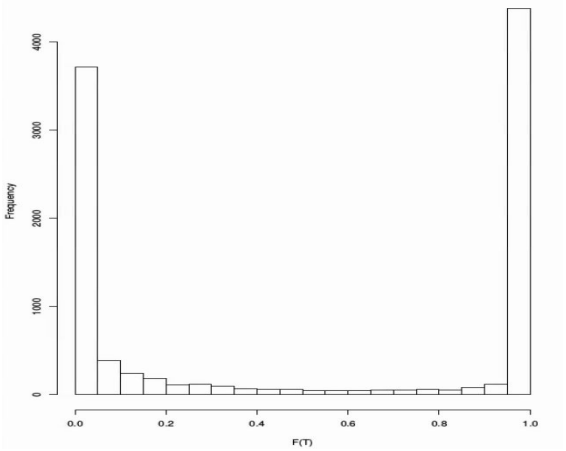


Figure 4.12: Marginal distribution of F(X) when t=31.5 and all observations are censored

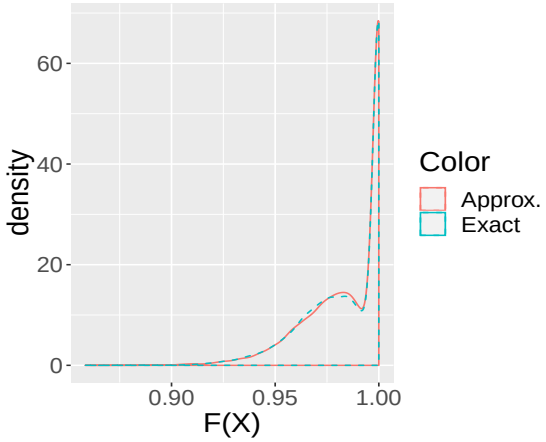


Figure 4.13: Exact and approximate distribution when merged with the draws in 4.12

Previous theories had suggested two scenarios when the approximation might be less than ideal: when the two components have very dissimilar precision and when the centering measure is extremely different (one is close to 0 and the other is close to 1). However, the improvements we have made in calculating precision and simulating from the distributions seem to have mitigated these issues, and all approximations seem close to ideal. Figure 4.14 combines 100 observations from a Weibull(4, 20) with 100 observations from a Weibull(8, 20) in series and evaluates it at t=25 (where $F(t) > 0.91$ for the first and $F(t) < 0.03$ for the second). The second combines data from two Weibull(4, 20) in parallel, but one has 1000 observations and one has only 10. In both cases the approximation is nearly indistinguishable from the actual distribution.

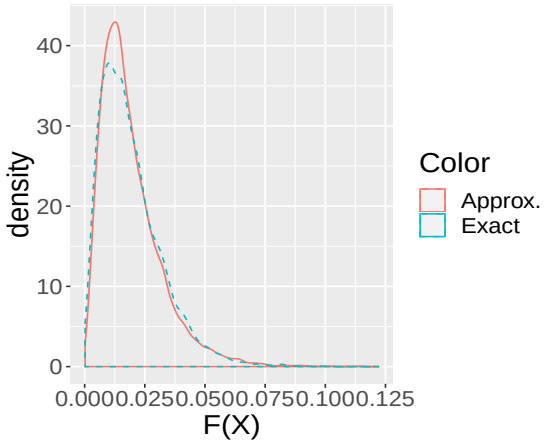


Figure 4.14: Combining two BSPs in series with highly different centering measures

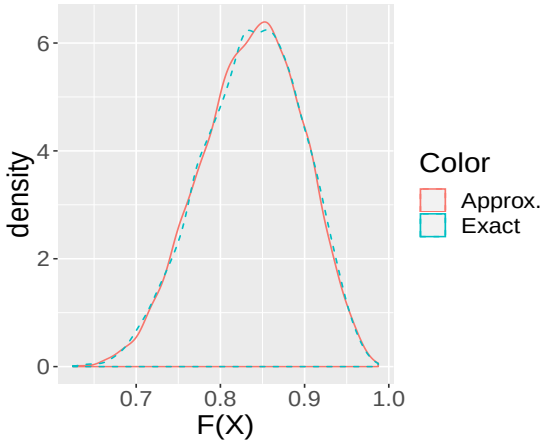


Figure 4.15: Combining two BSPs in parallel with highly different precisions

CHAPTER 5

CONCLUSION

In this project, we took a theoretical approach to reliability modeling and created the tools necessary to implement analyze the theory in practice. The R package created will be of great benefit to people who research and apply the beta-Stacy process to future work. The functions provided make creating, visualizing, and manipulating this data model much easier. In addition, the tools provided to perform a Bayesian nonparametric analysis on data will make it much easier to use on practical, applied reliability problems.

While creating the software and running the simulations, we found and addressed several problems that had gone unnoticed in previous work. We improved how precision was calculated. We improved the way we could simulate credible intervals. We were able to validate theory and ensure that by-hand calculations match what the code does.

The simulation studies were able to provide guidance on how the beta-Stacy process can be expected to perform on real data. We were able to show that we can quantify uncertainty in the failure rate at given times with credible intervals. While these intervals did not always produce nominal coverage, they aligned with the coverage we would expect from an equivalent beta-binomial model on the same data. Additionally we show that the estimates of the failure rate are consistent and the bias, in most situations, is quite small.

5.1 Limitations and Future Work

A key limitation of this approach lies in being able to find a BSP approximation to the true distribution of two BSPs in series or parallel. Because we are calculating the first and second moments at each time point on the true distribution and then fixing our parameters to match those moments, our method suffers similar problems to other method of moment techniques, namely, that the parameter estimate does not necessarily live in the range of the parameter

(Casella and Berger 2001, 312). It was frequently seen that the equations to calculate the BSP parameters would result in negative values for the α parameter. This was frequently a problem in the far tails of complex, multi-component systems. By how the equations are defined, the valid BSPs they produce will have the correct first and second moments. However, when invalid parameters are obtained using the equations, it seems indicative that no solution exists to provide those exact moments. One theoretical solution would be to find the nearest valid BSP with approximately the same moments, while loosening the restriction that the moments have to be exact, but this is not something we have addressed.

Several practical issues remain to be done before we could endorse widespread adoption of these methods. The code will need polishing and testing before it would meet the expectations of CRAN. Specifically, the code needs to deal clearly and rationally with cases when the precision is negative and undefined. In situations where negative precision values appeared, the inferences for the well-defined parts of the model generally seemed to be correct, but clear warnings and error messages will be necessary to communicate with the user.

Future work could also address some practical problems that applied statisticians would likely face. First, we would want to address how we can make inferences about two i.i.d. components from one set of test data, without double counting the data. Second, we could explore intelligent ways to address situations when two components do not fail completely independently of each other. Finally, engineers are often interested in estimating reliability after a fix is introduced (reliability growth), and we would want to provide guidance on approaches to that problem in the beta-Stacy context.

BIBLIOGRAPHY

Casella, G., and Berger, R. (2001), *Statistical Inference*, Duxbury Resource Center.

Kaplan, E. L., and Meier, P. (1958), "Nonparametric estimation from incomplete observations," *Journal of the American statistical association*, 53, 457–481.

Walker, S., and Muliere, P. (1997), "Beta-Stacy processes and a generalization of the Pólya-urn scheme," *The Annals of Statistics*, 1762–1780.

Warr, R. L., and Collins, D. H. (2014), "Bayesian nonparametric models for combining heterogeneous reliability data," *Journal of Risk and Reliability*, 228, 166–175.

Warr, R. L., and Greenwell, B. M. (2014), "A Hierarchical Nonparametric Bayesian Model that Integrates Multiple Sources of Lifetime Information to Model Large-Scale System Reliability," .

APPENDICES

BnpSysRel Vignette

Jackson Curtis

2019-02-25

Introduction

The goal the the BnpSysRel is to enable the estimation of system reliability for complex systems using the beta-Stacy process. In this vignette we walk through the basics of how an analysis would proceed, but for further theoretical details, see the Masters Project PDF attached to this repository.

In this vignette we will walk through estimating reliability for a simple system and how to organize your data in order to use the functions appropriately.

Our system

The function `estimateSystemReliability()` is going to do most of our heavy lifting here. This function takes three arguments. Once you know how to set up those three arguments you'll be ready to analyze your own data. The first argument `file` is looking for a string that will be the location of a text file to define how your system's components relate to each other. For this example, we will consider a simplistic version of a bike as our system of interest. Only a couple things can go wrong on this bike: either the front or the back tire can pop, or the front and the back brake could brake. If either tire breaks the bike becomes unusable, but because we like to live on the edge we will assume we only need one brake for the bike to function. In reliability, this is known as having the tires in series and the brakes in parallel (a redundant relationship). The tires and brakes are subsystems made of several components, and they are also in series.

The File

Let's look at how this text file should be written to tell R how our system works. We will use R to create the file, but you could just use your favorite text editor.

```
require(BnpSysRel)
file="MyBikeSystem.txt"
write.table("S(BackTire, FrontTire):Tires
P(BackBrake, FrontBrake):Brakes
S(Tires, Brakes):Bike", file=file, quote=F, row.names=F, col.names=F)
```

So what did we just do? The text file contains three lines. Each line will begin either with an S or a P, indicating whether it is specifying a parallel or series relationship. After, the names inside the parenthesis indicate what parts make up the component after the colon, e.g. Tires is made up of BackTire and FrontTire. The syntax is meant to be pretty simple, but the complete grammar rules can be found by reading through `?estimateSystemReliability`.

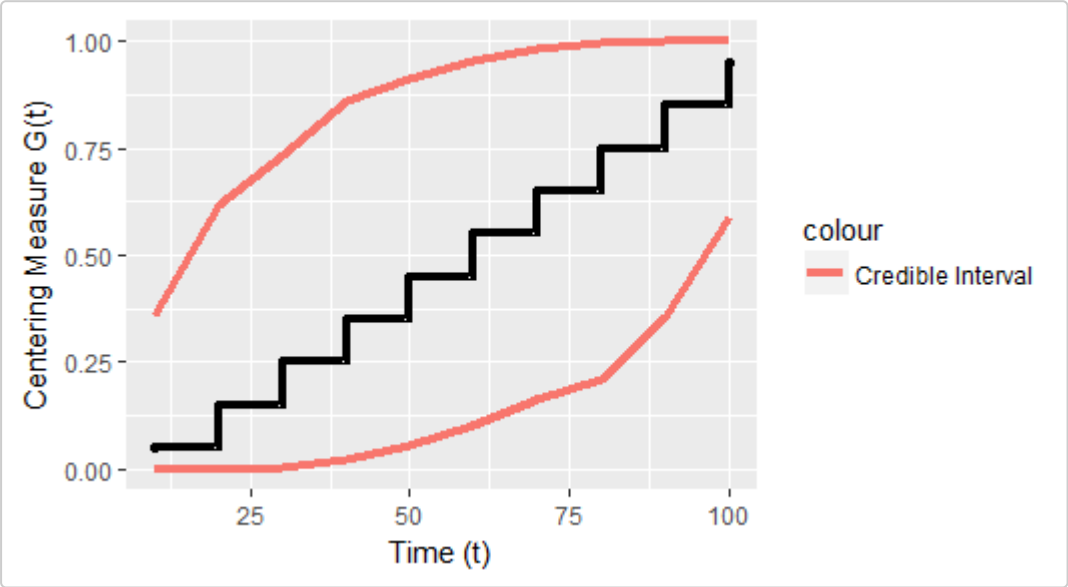
The priorList

So now that we've defined our system, we're done with the first argument to the function. Because our method is Bayesian in nature, the next argument, `priorList` specifies the priors we want to use on the components. Each prior in our model is a beta-Stacy Process, which is how we maintain conjugacy and get beta-Stacy posteriors, and the priorList is just a named list with a bunch of betaStacyProcess objects. So let's look at how to create a betaStacyProcess object:

```
tirePrior<-bsp(support = seq(10, 100, length.out = 10),
               centeringMeasure = seq(.05, .95, length.out = 10),
               precision = 3)
```

The `bsp()` function creates betaStacyProcess objects. Again, you'll have to read the paper for the more technical details, but we are basically defining our best guess as to what the CDF of the component should look like, in a discrete way. `support` is the list of jumps we want to see in our prior, `centeringMeasure` is how large those jumps should be, and `precision` is a measure of how certain we are in that CDF. We can look at our prior belief in what the CDF looks like:

```
plot(tirePrior, withConfInt = T)
```



We expect the CDF to be like the black line, but because we set a low precision (equivalent to measuring three failure times), our confidence is a very wide interval. Essentially, we're saying that at time t=50 (months maybe) we think somewhere between 5% and 90% of tires will have failed.

Now we do the same thing for the other components. We don't need priors for brakes, tires, or bike because we use the posterior of the subcomponents as the prior for the subsystem.

```
brakePrior<-bsp(support=c(22,44,66), centeringMeasure=c(.25,.5,.75), precision = 1)
priorList=list(FrontTire = tirePrior, BackTire=tirePrior,
               FrontBrake = brakePrior, BackBrake = brakePrior)
```

We're all done! Of course we don't have to assign the back tire and front tire the same prior, but for simplicity we will here.

The dataList

The dataList is the exact same format as the priorList, but instead of betaStacyPrior objects as elements of the list, we provide it data matrices. The data we collect is a bunch of failure times indicating how long it took the system to fail, which can be right-censored (meaning we ended the test before failure). Let's say we tested five back tires, the last two of which were censored:

```
backTireData<-matrix(c(14, 1,
                       29, 1,
                       67, 1,
                       75, 0,
                       75, 0), byrow = T, nrow=5)
```

The first column indicates the time recorded for each experiment, the second column is a 0/1 indicator for whether the observation was censored or not (1 if it was fully observed, 0 if it was censored). We can do the same for any component we have measurements on! If we don't have measurements, we will pass the computed prior through as the posterior. Let's finish measurements (with no more censoring).

```
set.seed(20)
frontTireData<-cbind(rnorm(50, 50, 10), 1)
backBrake<-cbind(rnorm(30, 20, 5), 1)
frontBrake<-cbind(rchisq(40, 20), 1)
bike<-cbind(c(40,47,42,46), 1)
dataList<-list(BackTire=backTireData,
               FrontTire=frontTireData,
                BackBrake=backBrake,
                FrontBrake=frontBrake,
                Bike=bike)
```

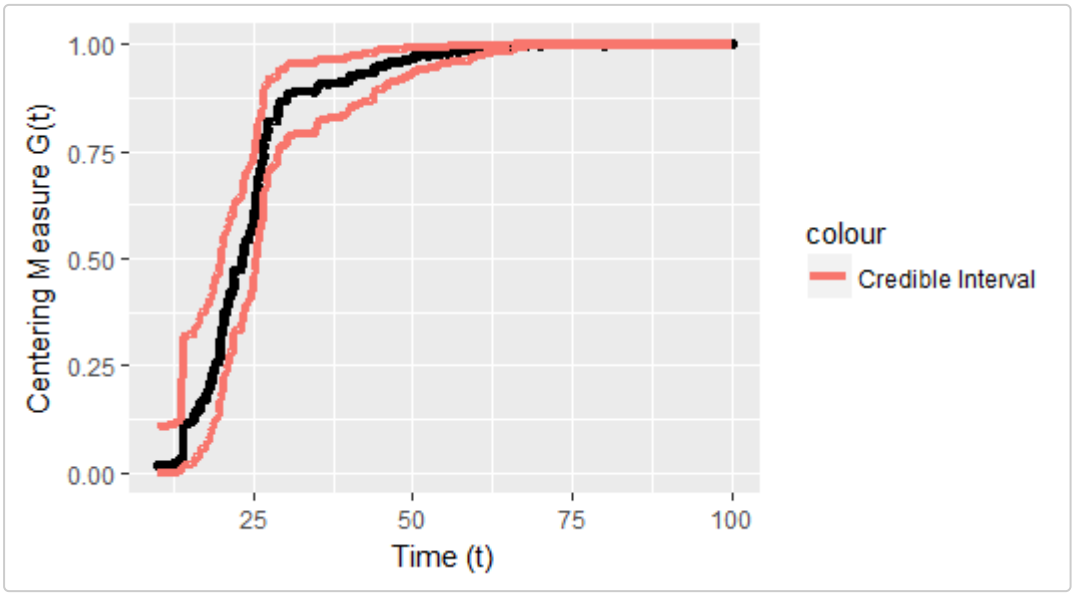
Now we have a list of our data matrices, and we are ready to use the function.

```
posteriors = estimateSystemReliability(file=file, priorList=priorList, dataList=dataList)
```

Analyzing the results

The function returns a posterior for every component in the system. We could explore and make inference on any of these components, but we will focus on the bike. Let's plot it:

```
plot(posteriors$Bike, withConfInt=TRUE)
```



Now we can get a point estimate and credible interval on the percentage of bicycles that will have failed by t=20 and t=50 months using the `evaluate_centering_measures()` and `bspConfint()` functions:

```
evaluate_centering_measures(posteriors$Bike, times=c(20,50))
```

```
## [1] 0.3283582 0.9701220
```

```
bspConfint(posteriors$Bike, times=c(20,50), conf.level = .05)
```

```
##              20        50
## 2.5%  0.1809604 0.9332901
## 97.5% 0.5167226 0.9931072
```

At t=20 our point estimate for the proportion of bikes that will have failed is 32.84% with a 95% credible interval of (18.1%, 51.7%).

Similarly we can calculate when we would expect the first 10% and 90% of bikes to have failed by using the quantile function:

```
quantile(posteriors$Bike, probs=c(.1,.9))
```

```
##      10%      90%
## 13.59176 34.78649
```

So 14 months in we expect 10% of the bikes to have failed and 35 months in we expect 90% of the bikes to have failed.

One last function that might be useful is that if you don't have an entire system, but just a single component, you can use the `bspPosterior()` function to calculate one posterior at a time. Its arguments are the same–one prior and one data matrix!

```
BackTirePosterior=bspPosterior(priorList$BackTire, dataList$BackTire)
```

We can see that they're the same whether regardless of which function we use:

```
print(posteriors$BackTire)
```

```
##    support centeringMeasure precision
## 1        0        0.0000000  8.000000
## 2       10        0.0187500  8.000000
## 3       14        0.1437500  8.000000
## 4       20        0.1812500  8.000000
## 5       29        0.3062500  8.000000
## 6       30        0.3437500  8.000000
## 7       40        0.3812500  8.000000
## 8       50        0.4187500  8.000000
## 9       60        0.4562500  8.000000
## 10      67        0.5812500  8.000000
## 11      70        0.6187500  8.000000
## 12      75        0.6187500  2.754098
## 13      80        0.7276786  2.754098
## 14      90        0.8366071  2.754098
## 15     100        0.9455357  2.754098
```

```
print(BackTirePosterior)
```

```
##    support centeringMeasure precision
## 1        0        0.0000000  8.000000
## 2       10        0.0187500  8.000000
## 3       14        0.1437500  8.000000
## 4       20        0.1812500  8.000000
## 5       29        0.3062500  8.000000
## 6       30        0.3437500  8.000000
## 7       40        0.3812500  8.000000
## 8       50        0.4187500  8.000000
## 9       60        0.4562500  8.000000
## 10      67        0.5812500  8.000000
## 11      70        0.6187500  8.000000
## 12      75        0.6187500  2.754098
## 13      80        0.7276786  2.754098
## 14      90        0.8366071  2.754098
## 15     100        0.9455357  2.754098
```