Right-Censoring Model

We employ a very simple right-censoring model, where the right-censoring time τ is fixed at some known value, e.g., an experiment is run for a fixed amount of time τ, and all systems that have not failed by the end of the experiment are right-censored. The censoring time S_i of the i^{th} system is thus given by

$$S_i = \min\{T_i, \tau\}.$$

So, after we generate the system failure time T_i, we generate the censoring time S_i by taking the minimum of T_i and τ.

Masking Model for Component Cause of Failure

We must generate data that satisfies the masking conditions described in Section 3.1. There are many ways to satisfying the masking conditions. We choose the simplest method, which we call the *Bernoulli candidate set model*. In this model, each non-failed component is included in the candidate set with a fixed probability p, independently of all other components and independently of $\boldsymbol{\theta}$, and the failed component is always included in the candidate set.

7.3 Issues with Convergence to the MLE

The surface of the log-likelihood function can be quite complex with many local maxima and ridges. This makes it difficult to find the MLE using local search methods like Newton-Raphson. In this section, we discuss some of the issues we encountered when estimating the MLE.

One issue we did not necessarily encounter in our simulation study, but is worth mentioning, is that the log-likelihood function may not be

Since this is a simulation study, we knew the true parameter value $\boldsymbol{\theta}$ and so started the optimization routine at the true parameter value. This is not the case in real-world scenarios, where the true parameter value is unknown, and so the optimization routine must start at some initial guess. We would have encountered even more issues if we had started the optimization routine at a poor initial guess.

Identifiability

When estimating the parameters, at may sometimes be the case that the likelihood function is not maximized at a unique point. This is known as the *identifiability* problem. If the likelihood function is not maximized at a unique point, then a lot of the theory we have developed so far breaks down (McLachlan and Krishnan, 2007).

In our case, since we are estimating the parameters of latent components, identifiability is not guaranteed. We consider two examples where this might occur, but there are many more.

1. The candidate sets could have been constructed in a way that prevents us from distinguishing between some of the components. For example, if in the candidate sets in a sample have the characteristic that component 1 is in a candidate set if and only if component 2 is in a candidate set, then we do not have enough information to estimate the parameters of component 1 and component 2 separately. This could happen, for instance, if the failure analysis is done by a human, and he or she is only able to identity that a larger component failed, but not the smaller components inside it. In this case, we may want to combine the two components into one component, and estimate the parameters of the combined component.

   In our Bernoulli candidate set model, this is something that can arise only by chance, and is unlikely to occur in practice, particularly for reasonably large samples.

2. The series system has a component that is the least reliable by a significant margin and is most likely the component cause of failure and is in every candidate set. In this case, our data may not be informative enough to estimate the parameters of the other components.

We constructed a quick experiment to demonstrate (2) above. In this experiment, we performed the following steps:

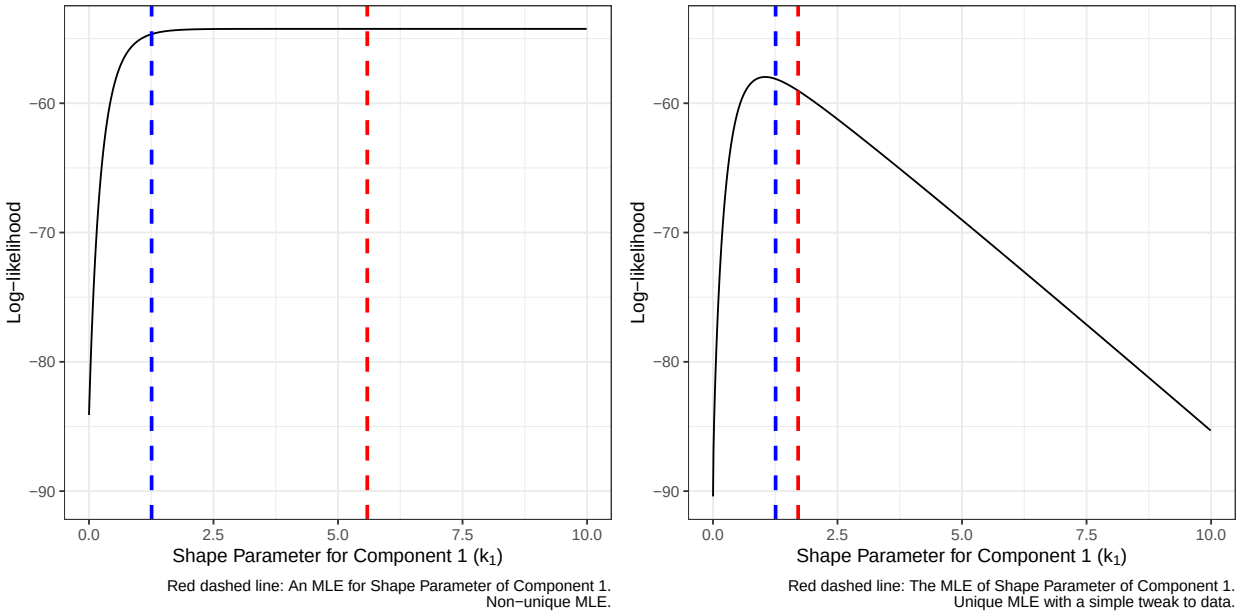

Figure 2: Log-likelihood Profile vs Shape Parameter for Component 1 (k_1): Non-unique MLE vs Unique MLE. Red Dashed Line is the True Shape Value.

1. We use the the series system from (Guo et al., 2013) as a base, but tweak it slightly to design the MTTF of the last component (component 3) be be two orders of magnitude smaller than the others. We did this by changing its scale parameter to $\lambda_3 = 4.1141$.

2. Generated a data set of size $n = 30$ from this system with a right-censoring time of $\tau = 6.706782$, corresponding to the 82.5% quantile of the system's lifetime, and with a masking probability $p = 0.215$ using the Bernoulli candidate set model.

3. We found an MLE by maximizing the log-likelihood function with the data set generated in step 2. As shown in the left plot in Figure 2, the log-likelihood function is flat, and therefore there there is no unique MLE. It appears any value of $\hat{k}_1$ larger than 3 will maximize the log-likelihood function.

   For a possible reason, we see that *every* candidate set in the data set contains component 3 because it was the component cause of failure in every system failure due to its significantly shorter MTTF.

4. We tweaked the data set in step 2 by removing component 3 from the candidate set in the first observation, leaving only component 1, which also happened to be in the candidate set (but if it was not, we could have simply inserted it). This is a very small change to the data set, but as shown in the right plot in Figure 2, the log-likelihood function is no longer flat, and there is a unique MLE. We also see that it is a much better estimate of the true parameter value for k_1, although we did not do the analysis to assess whether this generally holds.

According to this experiment, one could potentially justify either excluding these data sets from the analysis, or tweak them slightly as we had done, since otherwise they do not provide a unique MLE. In our simulation study, we mitgated these issues by choosing parameter values that are representative of real-world systems where there is no single component that is much less reliable than the others. We also use the Bernoulli candidate set model, which is unlikely to produce candidate sets that are not informative enough to estimate the parameters of the components, unless of course the masking probability p is very large.

After taking these precautions, we largely ignored identifiability issues in our simulation study, with the exception that we discarded any data sets that did not converge to a solution after 150 iterations.[1] A

[1]The choice of 150 iterations was driven by the computational demands of the simulation study combined with the subsequent bootstrapping of the confidence intervals.

log-likelihood function that is flat can cause our convergence criteria to take a long time to reach a solution. Therefore, a failure to converge within 150 iterations could be seen as evidence of potential identifiability issues.

Nonetheless, such scenarios occurred infrequently. During the bootstrapping of confidence intervals, we included all MLEs, even those that did not converge. This worst-case analysis approach was adopted because our main objective was to assess the performance of the BCa confidence intervals. We were concerned that if we took any additional steps, we may unintentionally bias the results in favor of producing narrow BCa confidence intervals with good coverage probabilities.

Parameter rescaling

When the parameters under investigation span different orders of magnitude, parameter rescaling can significantly improve the performance and reliability of optimization algorithms. Parameter rescaling gives an optimizer a sense of the typical size of each parameter, enabling it to adjust its steps accordingly. This is crucial in scenarios like ours, where shape and scale parametes are a few orders of magnitude apart. Without rescaling, the optimization routine may struggle, taking numerous small steps for larger parameters and overshooting for smaller ones. For more information, see (Nocedal and Wright, 2006).

Speed of convergence was particularly important in our case, since in our simulation study, we employ the bootstrap method to estimate the sampling distribution of the MLE, which requires us to estimate the MLE for many data sets. We found that parameter rescaling significantly improved the speed of convergence, which allowed us to run our simulation study in a tractable amount of time.

7.4 Assessing the Bootstrapped Confidence Intervals

Our primary interest is in assessing the performance of the BCa confidence intervals for the MLE. We will assess the performance of the BCa confidence intervals by computing the coverage probability of the confidence intervals. Under a variety of scenarios, we will bootstrap a 95%-confidence interval for $\boldsymbol{\theta}$ using the BCa method, and we will evaluate its calibration by computing the coverage probability and its precision by assessing the width of the confidence interval.

The coverage probability is defined as the proportion of times that the true value of $\boldsymbol{\theta}$ falls within the confidence interval. We will compute the coverage probability by generating R datasets from the Data Generating Process (DGP) and computing the coverage probability for each dataset. We will then aggregate this information across all R datasets to estimate the coverage probability.

7.5 Simulation Scenarios

We parameterize τ by quantiles of the series system, e.g., if $q = 0.8$, then $\tau(0.8)$ is the 80% quantile of the series system such that 80% of the sytems are expected to fail before time $\tau(0.8)$ and 20% of the series systems are expected to be right-censored.

We define a simulation scenario to be some comination of n and p. We are interested in choosing a small number of scenarios that are representative of real-world scenarios and that are interesting to analyze.

Here is an outline of the simulation study analysis:

1. Choose a scenario (sample size n, masking probability p (τ fixed).
2. Generate R datasets from the Data Generating Process (DGP). The DGP should be compatible with the assumptions in our likelihood model. In our case, we use:
   - Right-censored series lifetimes with $m = 5$ Weibull components.
   - Mmasking component cause of failure using Bernoulli candidate set model.
3. For each of these R datasets, calculate the Maximum Likelihood Estimator (MLE).
4. For each of these R datasets, perform bootstrap resampling B times to create a set of bootstrap samples.

5. Calculate the MLE for each of these bootstrap samples. This generates an empirical distribution of the MLE, which is used to construct a confidence interval for the MLE.

6. Repeat steps 4 and 5 for each of the R datasets.

7. For each dataset, determine whether the true parameter value falls within the computed CI. Aggregate this information across all R datasets to estimate the coverage probability of the CI.

8. Interpret the results and discuss the performance of the MLE estimator under various scenarios.

For how we generate a scenario, see Appendix A.

7.6 Effect of Right-Censoring on the MLE

In all of our simulation studies, we use a fixed right-censoring time $\tau = 377.71$, which is the 82.5% quantile of the series system. This means that 82.5% of the series systems are expected to fail before time τ and 17.5% of the series.

This represents a situation in which an experiment is run for a fixed amount of time τ, and all systems that have not failed by the end of the experiment are right-censored.

Right-censoring introduces a source of bias in the MLE. Right-censoring has the effect of pushing the MLE to estimate a lower value for the scale parameters and a higher value for the shape parameters. This is because when we observe a right-censoring event, we know that the system failed after the censoring time, but we do not know precisely when it will fail.

7.7 Effect of Masking the Component Cause of Failure on the MLE

The mean time to failure (MTTF) for the j^{th} component in a Weibull distribution is given by:

$$\text{MTTF} = \lambda_j \Gamma(1 + 1/k_j).$$

Scale Paramater: Positive Bias

As the scale parameter λ_j increases (keeping the shape parameter constant), the MTTF increases. When we increase the component masking, which in our simulation study is determined by the masking probability p, then there is less certainty about which component caused the system to fail, but we know it was one of the components in the candidate set. Therefore, to make it more likely that the component cause of failure is in the candidate set, we increase the scale parameters of the components in the candidate set, which increases their respective MTTFs.

We will demonstrate this effect in Section 7.9.

Shape Parameter: Negative Bias

As the shape parameter increases (keeping the scale parameter constant), the quantity $1/k_j$ decreases. Given that the gamma function Γ is a monotonically increasing function, a decrease in its argument (which is the case here since $1 + 1/k_j$ decreases with an increase in shape) results in a decrease in the MTTF.

Hence, when the shape parameter increases, the MTTF decreases, assuming the scale parameter is kept constant. Now, if the MLE for the shape parameter is positively biased (i.e., $E(\hat{k}_j) > k_j$), then the estimated shape parameter is larger than the true shape parameter. When we use this larger estimated shape parameter to compute the MTTF, the resulting MTTF will be smaller than the actual MTTF (assuming the scale parameter is kept constant or accurately estimated).

In other words, a positive bias in the estimated shape parameter would lead to an underestimation of the MTTF, assuming that the scale parameter is either known or accurately estimated.