Finite Sample Properties of Minimum Kolmogorov-Smirnov Estimator and Maximum Likelihood Estimator for Right-Censored Data

by

Jerzy Wieczorek

Master of Science
in
Statistics

Adviser: Dr. Jong Sung Kim
Second Reader: Dr. Mara Tableman

Portland State University

June 5, 2009

1. Summary

I am grateful to Dr. Kim for advising me on this Master's Project and suggesting the MKSFitter algorithm of Weber, Leemis, and Kincaid (2006) as a topic of study.

MKSFitter computes minimum Kolmogorov-Smirnov estimators (MKSEs) for several different continuous univariate distributions, using an evolutionary optimization algorithm, and recommends the distribution and parameter estimates that best minimize the Kolmogorov-Smirnov (K-S) test statistic. We modify this tool by extending it to use the Kaplan-Meier estimate of the cumulative distribution function (CDF) for right-censored data. Using simulated data from the most commonly-used survival distributions, we demonstrate the tool's **inability** to consistently select the correct distribution type with right-censored data, even for large sample sizes and low censoring rates. We also compare this tool's estimates with the right-censored maximum likelihood estimator (MLE). While the two estimation techniques have comparable accuracy at low censoring rates, the MKSE **significantly underperforms** the MLE at moderate and severe censoring rates.

2. Introduction

A common problem in analyzing biological or other survival data is to model the distribution function of a population. It is hoped that one's dataset is representative of the population being studied, and one attempts to find a model that shows a good fit to this data. Usually one must assume a certain distribution type and then estimate the model parameters that best fit the available data. In some cases prior knowledge of the system strongly suggests that a particular distribution is more appropriate than any other. However, when several models are possible, it can be tedious to fit and compare them all, and thus it is possible to overlook distributions that fit the data better than whatever initial distribution has been chosen.

Furthermore, some of the most commonly used distributions in survival analysis have quite a complicated form. Thus, standard estimators such as maximum likelihood estimators (MLEs) can be challenging or impossible to compute analytically, and often one is forced to resort to numerical optimization. These difficulties are exacerbated with censored data, which arises frequently in survival analyses.

In light of some of these challenges, Weber, Leemis, and Kincaid (2006) propose, implement, and test a software tool called MKSFitter that is designed to aid in distribution selection and parameter estimation in the case of uncensored data from a univariate continuous distribution. It implements minimum Kolmogorov-Smirnov estimation (MKSE), in which the parameter estimates for a given distribution are chosen so as to minimize the one-sample Kolmogorov-Smirnov (K-S) test statistic. Part of the appeal of MKSE stems from the fact that the K-S test is commonly used to select a distribution; thus, it makes sense to provide an estimator derived from the criterion function of the test. With no implementation of MKSE available, the K-S test has commonly been used with MLEs instead. Although MKSFitter does not actually perform the K-S test, it does report the value of the K-S statistic at the MKSEs for each distribution, which provides a descriptive measure of goodness-of-fit that can be used informally to select a distribution.

Gyorfi, Vajda, and Van Der Meulen (1996a, 1996b) explore the idea of MKSE from a theoretical standpoint and show that parameter estimates are consistent for most common distribution models. However, MKSFitter appears to be the first published software tool to implement the MKSE approach. The estimates are found by an evolutionary optimization algorithm, presented in Sobieszczanski-Sobieski, Laba, and Kincaid (1999), whose objective function is minimization of the K-S statistic. An overview of the algorithm, called bell-curve based (BCB) evolutionary optimization, is briefly presented in Weber et al. (2006) and is restated in section 4 of this paper. Further details of implementation and performance are available in Sobieszczanski-Sobieski et al. (1999) and Kincaid, Weber, and Sobieszczanski-Sobieski (2000). Other optimization tools (such

as a simple grid search) may be easier to use and understand, but we continue to use the BCB algorithm in this paper for the sake of consistency with the existing MKSFitter software.

The results of simulation studies by Weber et al. (2006) show that MKSFitter selects the correct distribution fairly well for large sample sizes on uncensored data: depending on the true distribution, the correct selection was made 40 to 70 percent of the time for samples of size $n = 100$. They also run an experiment which suggests that MKSE performance is comparable to MLE, in terms of the Euclidean distance (in the parameter space) of the respective parameter estimates from their true values. The mean distance from estimates to true values, standard deviation of these distances, and min and max distances are all found to be similar for MLEs and MKSEs.

However, MKSFitter is only designed to work on uncensored data. Weber et al. (2006) suggest that users can extend this tool to right-censored data by modifying the software to minimize the vertical distance between the fitted cumulative distribution function (CDF) and the Kaplan-Meier estimate of the CDF. (The Kaplan-Meier product-limit estimator is a commonly-used empirical estimate of the CDF for right-censored data. In the case where no observations are censored, it is equivalent to the usual empirical CDF.) Since this suggestion has not been followed up with any test results, it seemed to us to be worthwhile to carry out the idea and report its performance.

In other words, if $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ are iid from a population with unknown parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_k)$, the original MKSFitter finds estimates $\hat{\boldsymbol{\theta}}$ that minimize the K-S test statistic

$$D_n = \sup_x |F(x|\hat{\boldsymbol{\theta}}) - \hat{F}_n(x)| \tag{1}$$

where n is sample size, $F(x|\hat{\boldsymbol{\theta}})$ is the fitted CDF based on the given set of parameter estimates, and $\hat{F}_n(x)$ is the empirical CDF based directly on the data. For uncensored data, $\hat{F}_n(x)$ is a step function increasing by $1/n$ at every observation. For a right-censored dataset, however, we use the Kaplan-Meier product-limit estimator as $\hat{F}_n(x)$ and otherwise continue exactly as in Weber et al.:

For each proposed set of parameter estimates tested by MKSFitter, it calculates the K-S statistic to evaluate the fitness of that set of parameter estimates. (Technically this is no longer exactly the true K-S test, since the empirical CDF estimate is different. Nonetheless, the same kind of test statistic is calculated the usual way, by finding the absolute vertical difference between the theoretical CDF value at each step point and the empirical CDF values just before and just after that step, and then recording the maximum of these distances). Finally the BCB algorithm generates new 'children' parameter estimates near the best-fitting 'parents,' continuing for a pre-specified number of generations and ideally finding the globally-optimal estimates in that time.

3. **Implementation**

The original MKSFitter source code (in the C language) is available for download on the web at `www.math.wm.edu/~leemis`. In order to allow the use of censored data, we modified the C source code to accept both the observations and their corresponding empirical CDF values as input. Thus, a more user-friendly statistical software package such as R can compute the Kaplan-Meier estimator (or even some other type of empirical CDF estimate, for analyzing other kinds of censored data) and that estimated CDF can be passed in to MKSFitter. For ease of use, and to make efficient use of code already existing in R for computing the Kaplan-Meier estimator, we created an R function that runs MKSFitter and saves the results. In this way we performed several tests of the censored MKSFitter's performance.

In our tests, we used R to generate censored random samples from several distributions commonly used in survival analysis. Weber et al. report that the distributions used in MKSFitter are parameterized as in Leemis, but in fact the exponential distribution parameter is inverted from Leemis' definition. For clarity, we define the relevant distributions and parameters in the following table. Parameter estimates in the output of MKSFitter are identified simply as the estimated 'scale'

and 'shape' parameters for each distribution. (This may not be the standard notation for some of these parameters, but we would like to be consistent with the output of MKSFitter. For example, if we generate random data from the distribution we are calling 'Weibull(1, 1.5),' MKSFitter's output should report a 'Scale Param' estimate close to 1 and 'Shape Param' estimate close to 1.5.)

Exponential (scale λ)	Weibull (scale λ, shape κ)	LogNormal (scale λ, shape κ)
$f(t) = \frac{1}{\lambda} e^{-t/\lambda}$	$f(t) = \kappa \lambda^{\kappa} t^{\kappa-1} e^{-(\lambda t)^{\kappa}}$	$f(t) = \frac{1}{\kappa t \sqrt{2\pi}} e^{\frac{-(log(t)-\lambda)^2}{2\kappa^2}}$

R uses a different parameterization of the Exponential and Weibull densities. The respective R function calls to generate n data points from these distributions would be

Exp(λ):	`rexp(n,rate=1/lambda)`
Weibull(λ, κ):	`rweibull(n,shape=kappa,scale=1/lambda)`
LogNormal(λ, κ):	`rlnorm(n,meanlog=lambda,sdlog=kappa)`

Using MKSFitter's notation, the four models we used were Exponential(1), Weibull(1, 1.5), Weibull(1, 0.5), and LogNormal(2, 0.5). Their failure rates are constant, increasing (IFR), decreasing (DFR), and upside-down bathtub (UBT), respectively, which correspond to most of the common hazard function shapes.

For example, in the first round of tests, each of these distributions was used to generate 100 random samples (of size $n = 100$ each) to represent the true (uncensored) survival times, and the same distribution with slightly different parameters would be used to generate censoring times. An observation was uncensored if its survival time was lower than its censoring time, in which case the true survival time was placed in the dataset; otherwise, the observation was treated as censored and its censoring time was placed in the dataset. In this way, we formed datasets of $n = 100$ observations each but with variable censoring rates. For instance, if the survival times were generated from Exp(1) and the censoring times were generated from Exp(3), then on average about 25% of the observations were censored. With censoring times from Exp(1/3), about 75% of the observations were censored. For each distribution type, we used trial-and-error to find censoring distribution parameters that caused 25% and 75% mean censoring rates. 50% censoring rates resulted from using the true time distribution as the censoring distribution. Thus, we were able to test the influence of censoring rate on MKSFitter performance.

Hazard function	True times	25% censoring times	75% censoring times
Constant	Exp(1)	Exp(3)	Exp(1/3)
IFR	Weibull(1, 1.5)	Weibull(0.5, 1.5)	Weibull(2, 1.5)
DFR	Weibull(1, 0.5)	Weibull(1/9, 0.5)	Weibull(9, 0.5)
UBT	LogNorm(2, 0.5)	LogNorm(2.5, 0.5)	LogNorm(1.5, 0.5)

4. **Optimization algorithm**

The BCB (bell-curve based) evolutionary optimization algorithm of Sobieszczanski-Sobieski et al. (1999) is a general tool that can be used to optimize any mathematical problem with an objective function and constraints. Generally speaking, the objective function is evaluated on points in a k-dimensional parameter space. For most of the distributions considered here, $k = 2$ (except for Exponential where $k = 1$), so we will explain the BCB method for $k = 2$ and generalize from there.

For a given dataset, this optimization is performed separately for each distribution type. Consider a 2-dimensional parameter space, letting the x-axis represent the scale parameter and the y-axis represent the shape parameter (or vice versa). In the first step of optimization for a given distribution type, the BCB algorithm generates an intial population of μ random points throughout the parameter space, using the uniform distribution within boundaries set by the user. (For instance, we specified that each parameter should lie between 0.001 and 100, so both the scale and

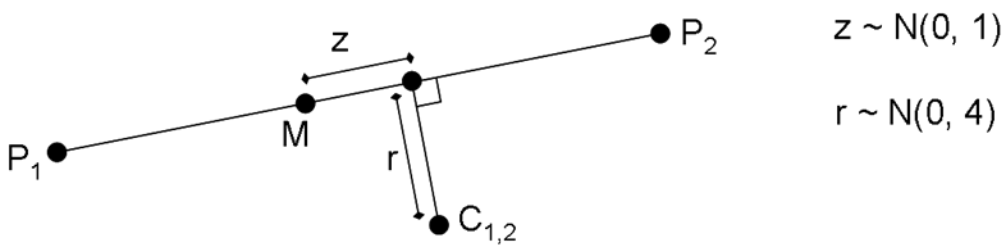

FIGURE 1. Example of generating a child (C) from two parents (P_1, P_2) using BCB algorithm

shape parameters for each initial point are i.i.d. Uniform(0.001, 100). The default value of μ is 50, so 50 such bivariate points are generated.) These μ points are the first set of 'parents.'

In the next stage, each of these μ points is evaluated by the objective function. In our case this means that the K-S statistic is calculated for the dataset against the given distribution, once for each parent in the initial population. The parents are ranked according to their 'fitness,' i.e. the fittest parents are the parameter values with the lowest K-S statistic value. μ pairs of parents are selected for mating, such that the fittest parents are most likely to be selected to mate most often. (*Intuitively*, each parent in a mating pair is selected by the equivalent of spinning a roulette wheel whose slots are spaced in proportion to fitness, so that the fittest parent (with the lowest K-S value) has the largest chance of being selected each time. *Technically*, this is implemented with an algorithm called Baker's stochastic universal sampling, presented in Appendix A, which runs more efficiently than simulating 2μ roulette spins directly.) Each mating pair of parents generates a 'child,' as described in the next paragraph. Finally the children are evaluated too, and the μ best points are selected (out of all 2μ parents and children together) to constitute the parent population in the next generation. After a prespecified number of generations, the algorithm terminates, with the hope that it will have converged to an optimal solution by then.

The previous paragraph describes standard evolutionary algorithms in general, such as are discussed in Back (1996). The unique feature of the BCB algorithm is the particular way in which it generates children from each pair of parents P_1 and P_2. See Figure 1 for an example. First, the program finds the weighted mean M of the two parents. (*Intuitively*, if both parents are equally fit, M is the midpoint between them; otherwise, it will be closer to the fitter parent (with the lower K-S statistic value). *Technically*, if K_i is the value of the K-S statistic for parent i, the weighted mean is $\frac{K_{P1}}{K_{P1}+K_{P2}}$ of the way along the line from P1 towards P2.) Next, a normal random variate z is generated from N(0, σ_m^2), where σ_m is a prespecified parameter of this algorithm (default value 1.0). The point M is mutated by shifting z units (arbitrarily in either direction) along the line $\overline{P_1P_2}$ between the parents. Since z may turn out to be larger than the distance to the appropriate parent, $M + z$ is not necessarily restricted to be between the parents.

In the 1-dimensional case, as for the Exponential distribution, the algorithm stops here. For the 2-dimensional case, there is an additional mutation, which generates a random distance r from N(0, σ_r^2), and places the child perpendicular to $\overline{P_1P_2}$ at r units away from $M + z$. Again, σ_r is another prespecified parameter of the algorithm (default value 4.0).

In the k-dimensional case, P_1 and P_2 are points in k-space, as are their weighted midpoint M, the shifted midpoint $M + z$, and the actual child C. z still shifts the midpoint along the line $\overline{P_1P_2}$, and C is still perpendicular to that line at $M + z$. However, with the added dimensions, the set of all points r units away from $M + z$ and perpendicular to $\overline{P_1P_2}$ becomes not two points but a hypersphere. For $k = 3$, it is a circle around the line $\overline{P_1P_2}$; in four dimensions, it is a sphere 'around' that line; etc. In these situations there is one additional necessary step, which is to select C by sampling uniformly from this hypersphere around $\overline{P_1P_2}$ at radius r from $M + z$.

In the MKSFitter implementation, the optimization algorithm's four parameters (σ_m, σ_r, population size μ, and number of generations N) are settings that can be changed by the user. It is difficult to gauge the effect of changing σ_m and σ_r, so they were left at their default values for this research. However, the number of randomly-generated starting points and the number of generations did have to be edited for this research. We had to change the default values (from 50 to 200 for initial population size, and from 100 also to 200 for number of generations) in order to avoid some very bad fits. In practice, users evaluating a single dataset at a time can use the lower values for speedier processing, since they will notice any poor fits (marked by K-S statistic values near 0.5, usually meaning the fitted CDF is almost a horizontal line with value 0.5 over the range of the data). A user noticing such problems, in applying MKSFitter to a single dataset, can tweak these parameters and process the dataset again. However, when generating and analyzing a hundred datasets in a row, as we did, it is impractical to stop after each one and re-tweak it, so we had to use 'safer' settings that unfortunately led to longer run-times (approximately 30 minutes for 100 datasets of size 100 each, and approximately 3 hours for 500 datasets of size 200 each). In future work, it would be much easier on the user (even if slightly less computationally efficient) to simply perform a grid search instead of such complicated BCB optimization.

Also worth noting is that the BCB algorithm's random number generator appears to generate the same random starting values each time. When we would re-analyze a given dataset that had obviously-bad results, it kept reproducing the same exact results until we increased the initial population size (at which point the fit would improve dramatically).

5. **Results**

5.1. **Test 1: Distribution identification.** In our first set of tests, presented in tables 1 through 4, we evaluated how often the MKSFitter identified the correct distribution for moderate sample sizes. (For each true distribution and censoring rate, we randomly generated 100 datasets of size $n = 100$ each). As these tables show, the frequency of selecting the correct distribution decreased monotonically as censoring rate increased. Naturally, more censoring means more uncertainty about the data, a worse Kaplan-Meier estimate of the CDF, and fewer uncensored points at which to fit the distribution. However, even for the lowest censoring rate, the highest correct identification rates were around 50% at most, and only for Weibull data at that. Thus, it does not seem safe to rely on MKSE to select the correct distribution. However, if you start out with a particular distribution in mind, presumably MKSFitter could at least be used to warn you that this choice of distribution is unreasonable if its K-S statistic was much higher than that of other distributions.

5.2. **Test 2: Parameter estimation.** In our second set of tests, shown in tables 5 through 8, we evaluated how well the MKSEs matched the correct parameters of the true generating distribution against how the MLEs performed. Censored MLEs were calculated by the 'survreg' function in R. Again, for each distribution type and censoring rate, we generated 100 datasets of size $n = 100$ each.

As censoring rate increases, the fit get worse for both MLEs and MKSEs. This is to be expected, since there is less and less information about the data. However, of particular importance is the fact that MKSEs get worse more severely than the MLEs. Thus, the censored MLEs appear to be much better parameter estimates than MKSEs, except at the lowest censoring rates where both perform similarly. Overall, both tests indicated the worst performance for LogNormal data, which was also the case for uncensored data tests presented in Weber et al.

5.3. **Test 3: Large-sample properties.** The third set of tests is essentially a repetition of the previous tests, but performed for many more iterations and larger samples (500 datasets of size $n = 200$ each). These test results are presented in Tables 9 through 16.

Tables 9 through 12 mirror Tables 1 through 4, except with an additional feature: for this test, we also record how much the 'winning' distribution outperformed the correct distribution's

K-S statistic on average. For example, in Table 9 we see that at censoring rate 0.25, the Weibull distribution's ΔKS value is 0.010. This means that in the instances where the Weibull distribution had the lowest K-S statistic value, its K-S statistic was only 0.010 units lower than the Exponential distribution's K-S statistic (on average). For all four distribution types and all three censoring rates, except perhaps the highly-censored LogNormal data, the ΔKS value tends to be quite small compared to the AvgKS value. This suggests that when the correct distribution fails to 'win,' at least it is not usually losing by much.

If the true distribution's K-S value tended to be considerably higher than the 'winning' distribution's value, then MKSFitter would be worthless as a 'sanity check' tool. However, as it is, a sensible approach may be to begin with an a priori idea of which distribution to use, and then change your mind only if your distribution's K-S value is much higher than other values. If your distribution were correct, it would probably have a lower K-S value, but not necessarily the lowest.

Tables 13 through 16 mirror Tables 5 through 8. The same pattern is evident in both sets of tables: as censoring rate increases, both estimators perform worse, but the MKSEs get worse much more severely than the MLEs do. In short, even in the large-sample case, we cannot say that MKSE performs as well as MLE for heavily-censored data.

6. **Conclusions**

Clearly distribution selection performance degrades at high censoring rates. This fact in itself would not condemn the use of MKSFitter for censored data, since any estimation technique is necessarily going to suffer as more and more of the data is censored. However, since MLE consistently performs at least as well as MKSE (and often much better) in terms of distance from the estimated to the true parameters, MKSE does not appear to have an advantage over the better-studied MLE approach. This is particularly clear for highly-censored data.

The censored MLE's superior performance is probably due to the fact that it uses the entire likelihood function and thus manages to incorporate partial information on every observation. The MKSE does this only in a roundabout way: it uses the Kaplan-Meier estimated CDF, which is based on all the data, but it calculates the difference in empirical and parametric estimated CDFs only at the uncensored observations. Thus, MKSFitter essentially does the same kind of fitting as before, except that now it has only $n*$*(censoring rate)* effective observations, compared to the n observations the MLE incorporates. Hence it is unsurprising that the MLE should perform better.

Furthermore, the MKSE approach does not generate any standard errors of the estimates. Using bootstrapping to add this functionality could make MKSFitter a more useful tool and allow for some hypothesis testing or confidence interval construction for the parameter estimates. However, the distance standard deviations in Tables 5 through 8 and 13 through 16 suggest that MKSEs are more variable than MLEs, again suggesting that MLEs are the better tool.

Donoho and Liu (1988) have shown that for some true distributions arbitrarily close to your assumed model distribution, MKSE can have arbitrarily high variance. However, an estimator based on minimizing the Cramer-VonMises statistic does not have the problem of such inconsistency. Thus, it is conceivable that such an estimator may perform better than the K-S statistic, including on censored data. Furthermore, the Cramer-VonMises statistic involves an integral rather than a supremum, so it is more sensitive to the full shape of the estimated empirical CDF – another reason why it might conceivably find better fits than the MKSE. This is a potential area for future study.

In any case, one strong point of the MKSFitter software is that it prints out a K-S statistic for all of the tested distribution types. This can be useful as a 'sanity check' to make sure there is not another distribution with a much better fit than the one proposed *a priori*. Another possible use for MKSE is in simulation studies in which one wishes to replicate the properties of the sample data, rather than to make inferences about the true parameters of the original population. In such cases, one might consider 'smoothing' the data by generating new datasets from the distribution and

parameter estimates whose CDF matches the original data most closely, even if that distribution is not really the 'correct' one.

Finally, despite all of the above concerns, MKSFitter may potentially be more useful if expanded to analyze multivariate data. It can be quite challenging to apply maximum likelihood estimation on bivariate and higher-dimension distributions in censored situations. Thus in some cases it may be easier to estimate the parameters numerically with a MKSE approach, although it remains to be seen whether the resulting estimates would be good enough to be useful.

Nonetheless, the overall sense is that MKSE is not a useful approach for right-censored data when MLEs are available (even when finding the MLE involves computational optimization rather than an analytical solution). A more promising course of future study is to return to uncensored data and try implementing a minimum Cramer-VonMises estimation (MCVME) tool. Another question is how this approach performs against MLE in corrupted datasets: is MKSE (or MCVME) any more robust than MLE? It could also be worthwhile to investigate how the power of the K-S test would change if MKSEs were used instead of MLEs, since it seems sensible to use an estimator based on the actual test statistic. Finally, all of the distributions considered in this paper are regular families, which are precisely the distributions for which MLEs naturally have desirable properties and good performance. Perhaps in non-regular distributions the MKSE and similar approaches would be able to outperform MLE.

References

[1] Weber, M., Leemis, L., and Kincaid, R. (2006). Minimum Kolmogorov-Smirnov test statistic parameter estimates. *Journal of Statistical Computation and Simulation* **76**, 195-206.

[2] Gyorfi, L., Vajda, I., and Van Der Meulen, E. (1996a). Minimum Kolmogorov distance estimates of parameters and parametrized distributions. *Metrika* **43**, 237-255.

[3] Gyorfi, L., Vajda, I., and Van Der Meulen, E. (1996b). Minimum Kolmogorov distance estimates for multivariate parametrized families. *American Journal of Mathematical and Management Sciences* **16**, 167-191.

[4] Sobieszczanski-Sobieski, J., Laba, K., and Kincaid, R. (1999). Bell-curve based evolutionary optimization algorithm. *Structural Optimization* **18**, 264-276.

[5] Kincaid, R., Weber, M., and Sobieszczanski-Sobieski, J. (2000). Performance of a bell-curve based evolutionary optimization algorithm. Proceedings of the 41st AIAA Structures, Structural Dynamics, and Materials Conference, Atlanta, GA, 3-6 April, AIAA Paper 2000-1388.

[6] Leemis, L. (1995). *Reliability: Probabilistic Models and Statistical Methods.* Englewood Cliffs, NJ: Prentice-Hall.

[7] Back, T. (1996). *Evolutionary Algorithms in Theory and Practice.* Oxford: Oxford University Press.

[8] Donoho, D. and Liu, R. (1988). Pathologies of some minimum distance estimators. *The Annals of Statistics* **16**(2), 587-608.

Appendix A. **Baker's stochastic universal sampling**

This algorithm, which originated in a paper by Baker in 1987, can be found in Back (1996) on page 120. It is intended to be a more computationally-efficient way to simulate many spins of a roulette wheel with uneven slot spacings. However, the specific implementation described here is the one used in the C code for MKSFitter.

We have μ individuals in the current population (i.e. μ sets of parameter estimates in the current generation of parents). Let K_i be the value of the K-S statistic for the i^{th} individual, and let $\bar{K}$ be the average of all μ individuals' K-S values. Then the i^{th} individual's fitness is defined as $\Phi_i = \bar{K}/K_i$. The i^{th} individual's selection probability is defined as $p_i = \frac{\Phi_i}{\sum_{j=1}^{\mu} \Phi_j}$. Finally, the expected number of occurrences of individual i in the population is $\eta_i = \mu \cdot p_i$.

However, in our case, we need to generate μ pairs of parents. To do this, we need to temporarily create an ordered set of 2μ individuals. Let us call this set T. We will expect to see individual i occurring $2\eta_i$ times in this temporary set T. (Once T is generated, we pair up the 1^{st} member with the $(\mu+1)^{th}$, then the 2^{nd} member with the $(\mu+2)^{th}$, etc., to form the pairs that will be mated to generate children.)

Baker's algorithm ensures that each individual appears exactly either $\lfloor 2\eta_i \rfloor$ or $\lceil 2\eta_i \rceil$ times in the set T; that the bias (expected difference between $2\eta_i$ and the actual number of appearances) is zero; and computational complexity is low.

Here is a potential realization of the algorithm using R:

```
C <- 0
S <- runif(n=1,min=0,max=1)
for(i in 1:mu){
  S <- S + nu[i]
  while(S > C){
    append(T,individual[i])
    C <- C + 1
  }
}
```

The algorithm begins by generating a random number S uniformly from the interval $[0, 1)$ and keeping a counter C starting at 1. For each individual i in the old population, for $i \in (1, \mu)$, S increases by $2\eta_i$. Then a copy of the i^{th} individual is added to T as many times as the counter C can be incremented upwards by 1 without going over the current value of S. (This counter ensures that we add exactly either $\lfloor 2\eta_i \rfloor$ or $\lceil 2\eta_i \rceil$ copies of individual i to T. The intial value of S and the order in which the observations are processes determines whether it is $\lfloor 2\eta_i \rfloor$ or $\lceil 2\eta_i \rceil$ for a particular i.) Once C is too high to let us add any more copies of individual i to T, the algorithm proceeds to the $(i + 1)^{th}$ individual.

TABLE 1. Best distributions identified: Exp(1)
100 datasets of size n=100

	Censoring rate = .25		Censoring rate = .50		Censoring rate = .75	
	Frequency	AvgKS	Frequency	AvgKS	Frequency	AvgKS
Exponential	0.35	0.062	0.32	0.095	0.16	0.181
Normal	0.01	0.094	0.10	0.113	0.04	0.158
Weibull	0.42	0.057	0.29	0.084	0.08	0.117
Log normal						
Log logistic	0.21	0.059	0.26	0.088	0.22	0.084
Gompertz	0.01	0.054	0.03	0.084	0.25	0.078
Gamma						
Exponential power					0.25	0.100

TABLE 2. Best distributions identified: Weibull(1, 1.5)
100 datasets of size n=100

	Censoring rate = .25		Censoring rate = .50		Censoring rate = .75	
	Frequency	AvgKS	Frequency	AvgKS	Frequency	AvgKS
Exponential			0.01	0.135	0.12	0.255
Normal			0.06	0.086	0.04	0.194
Weibull	0.48	0.049	0.25	0.064	0.11	0.105
Log normal			0.01	0.139		
Log logistic	0.12	0.057	0.21	0.068	0.27	0.100
Gompertz	0.17	0.050	0.27	0.083	0.32	0.092
Gamma			0.04	0.053		
Exponential power	0.23	0.051	0.15	0.066	0.15	0.097

TABLE 3. Best distributions identified: Weibull(1, 0.5)
100 datasets of size n=100

	Censoring rate = .25		Censoring rate = .50		Censoring rate = .75	
	Frequency	AvgKS	Frequency	AvgKS	Frequency	AvgKS
Exponential					0.09	0.270
Normal					0.04	0.191
Weibull	0.52	0.049	0.29	0.067	0.18	0.123
Log normal			0.01	0.139		
Log logistic	0.14	0.054	0.24	0.072	0.20	0.100
Gompertz					0.04	0.083
Gamma	0.04	0.044	0.02	0.057	0.02	0.101
Exponential power	0.30	0.053	0.44	0.078	0.43	0.088

TABLE 4. Best distributions identified: LogNorm(2, 0.5)
100 datasets of size n=100

	Censoring rate = .25		Censoring rate = .50		Censoring rate = .75	
	Frequency	AvgKS	Frequency	AvgKS	Frequency	AvgKS
Exponential						
Normal			0.07	0.078	0.26	0.113
Weibull	0.06	0.051	0.16	0.076	0.10	0.100
Log normal	0.37	0.051	0.27	0.079	0.07	0.125
Log logistic	0.44	0.053	0.29	0.075	0.26	0.090
Gompertz			0.06	0.102	0.23	0.132
Gamma	0.13	0.051	0.13	0.066	0.01	0.095
Exponential power			0.02	0.074	0.07	0.188

TABLE 5. Distances from estimates ($\hat{\lambda}$) to true values: Exp(1)
100 datasets of size n=100

	Censoring rate = .25				Censoring rate = .50				Censoring rate = .75			
	Mean	StDev	Min	Max	Mean	StDev	Min	Max	Mean	StDev	Min	Max
MKSE	0.10	0.08	0.00	0.36	0.12	0.08	0.00	0.32	0.21	0.14	0.00	0.71
MLE	0.10	0.08	0.00	0.38	0.11	0.08	0.01	0.37	0.17	0.14	0.00	0.83

TABLE 6. Distances from estimates ($\hat{\lambda}, \hat{\kappa}$) to true values: Weibull(1, 1.5)
100 datasets of size n=100

	Censoring rate = .25				Censoring rate = .50				Censoring rate = .75			
	Mean	StDev	Min	Max	Mean	StDev	Min	Max	Mean	StDev	Min	Max
MKSE	0.16	0.11	0.01	0.76	0.27	0.21	0.01	1.49	0.51	0.41	0.08	2.64
MLE	0.14	0.08	0.01	0.40	0.17	0.10	0.02	0.50	0.25	0.14	0.03	0.68

TABLE 7. Distances from estimates ($\hat{\lambda}, \hat{\kappa}$) to true values: Weibull(1, 0.5)
100 datasets of size n=100

	Censoring rate = .25				Censoring rate = .50				Censoring rate = .75			
	Mean	StDev	Min	Max	Mean	StDev	Min	Max	Mean	StDev	Min	Max
MKSE	0.23	0.20	0.01	1.25	0.30	0.24	0.01	1.16	0.52	0.40	0.02	1.86
MLE	0.21	0.17	0.00	0.90	0.28	0.23	0.03	1.34	0.43	0.32	0.02	1.79

TABLE 8. Distances from estimates ($\hat{\lambda}, \hat{\kappa}$) to true values: LogNorm(2, 0.5)
100 datasets of size n=100

	Censoring rate = .25				Censoring rate = .50				Censoring rate = .75			
	Mean	StDev	Min	Max	Mean	StDev	Min	Max	Mean	StDev	Min	Max
MKSE	0.07	0.04	0.01	0.19	0.19	0.62	0.00	6.26	1.94	7.69	0.04	56.8
MLE	0.06	0.03	0.01	0.19	0.09	0.04	0.01	0.19	0.10	0.07	0.00	0.56

TABLE 9. Best distributions identified: Exp(1)
500 datasets of size n=200
(ΔKS is amount by which 'winning' distribution outperformed Exp)

	Censoring rate: .25			Censoring rate: .5			Censoring rate: .75		
	Freq	AvgKS	ΔKS	Freq	AvgKS	ΔKS	Freq	AvgKS	ΔKS
Exp				0.01	0.073		0.14	0.154	
Norm							0.06	0.149	0.032
Weib	0.42	0.037	0.010	0.24	0.054	0.015	0.12	0.088	0.018
LogNorm							0.01	0.055	0.026
LogLog	0.04	0.042	0.016	0.08	0.053	0.026	0.14	0.071	0.019
Gomp	0.35	0.036	0.009	0.21	0.051	0.014	0.25	0.087	0.029
Gamma	0.04	0.040	0.010	0.02	0.057	0.017	0.01	0.072	0.019
ExpPwr	0.15	0.037	0.017	0.44	0.059	0.018	0.28	0.080	0.016

TABLE 10. Best distributions identified: Weibull(1, 1.5)
500 datasets of size n=200
(ΔKS is amount by which 'winning' distribution outperformed Weibull)

	Censoring rate: .25			Censoring rate: .5			Censoring rate: .75		
	Freq	AvgKS	ΔKS	Freq	AvgKS	ΔKS	Freq	AvgKS	ΔKS
Exp				0.01	0.112	0.000	0.07	0.167	0.000
Norm				0.01	0.046	0.005	0.05	0.149	0.001
Weib	0.71	0.037		0.37	0.052		0.15	0.088	
LogNorm							0.01	0.200	0.000
LogLog	0.06	0.043	0.007	0.12	0.062	0.008	0.22	0.077	0.004
Gomp	0.04	0.038	0.013	0.26	0.063	0.009	0.38	0.085	0.007
Gamma	0.03	0.038	0.003	0.06	0.048	0.003			
ExpPwr	0.16	0.038	0.005	0.18	0.052	0.005	0.13	0.091	0.003

TABLE 11. Best distributions identified: Weibull(1, 0.5)
500 datasets of size n=200
(ΔKS is amount by which 'winning' distribution outperformed Weibull)

	Censoring rate: .25			Censoring rate: .5			Censoring rate: .75		
	Freq	AvgKS	ΔKS	Freq	AvgKS	ΔKS	Freq	AvgKS	ΔKS
Exp							0.06	0.226	0.000
Norm							0.01	0.177	0.010
Weib	0.79	0.035		0.32	0.059		0.09	0.043	
LogNorm							0.05	0.040	0.002
LogLog	0.03	0.042	0.008	0.18	0.051	0.006	0.22	0.044	0.001
Gomp				0.01	0.116	0.000	0.04	0.099	0.001
Gamma	0.02	0.035	0.008	0.02	0.048	0.003	0.01	0.037	0.001
ExpPwr	0.16	0.035	0.006	0.49	0.064	0.005	0.52	0.053	0.002

TABLE 12. Best distributions identified: LogNorm(2, 0.5)
500 datasets of size n=200
(ΔKS is amount by which 'winning' distribution outperformed LogNormal)

	Censoring rate: .25			Censoring rate: .5			Censoring rate: .75		
	Freq	AvgKS	ΔKS	Freq	AvgKS	ΔKS	Freq	AvgKS	ΔKS
Exp									
Norm				0.03	0.068	0.011	0.16	0.095	0.051
Weib	0.04	0.045	0.012	0.16	0.058	0.012	0.12	0.084	0.051
LogNorm	0.40	0.038		0.26	0.054		0.07	0.099	
LogLog	0.36	0.038	0.004	0.31	0.055	0.004	0.31	0.079	0.076
Gomp				0.01	0.073	0.020	0.24	0.101	0.056
Gamma	0.20	0.037	0.005	0.22	0.054	0.005	0.04	0.068	0.055
ExpPwr				0.01	0.070	0.022	0.05	0.116	0.023

TABLE 13. Distances from estimates ($\hat{\lambda}$) to true values: Exp(1)
500 datasets of size n=200

	Censoring rate: .25				Censoring rate: .5				Censoring rate: .75			
	Mean	StDev	Min	Max	Mean	StDev	Min	Max	Mean	StDev	Min	Max
MKSE	0.07	0.05	0.00	0.29	0.10	0.08	0.00	0.50	0.16	0.12	0.00	0.96
MLE	0.07	0.05	0.00	0.25	0.08	0.06	0.00	0.31	0.11	0.09	0.00	0.59

TABLE 14. Distances from estimates ($\hat{\lambda}, \hat{\kappa}$) to true values: Weibull(1, 1.5)
500 datasets of size n=200

	Censoring rate: .25				Censoring rate: .5				Censoring rate: .75			
	Mean	StDev	Min	Max	Mean	StDev	Min	Max	Mean	StDev	Min	Max
MKSE	0.12	0.07	0.00	0.42	0.18	0.12	0.01	0.80	0.44	0.34	0.01	2.31
MLE	0.10	0.06	0.01	0.32	0.12	0.07	0.01	0.42	0.18	0.11	0.01	0.58

TABLE 15. Distances from estimates ($\hat{\lambda}, \hat{\kappa}$) to true values: Weibull(1, 0.5)
500 datasets of size n=200

	Censoring rate: .25				Censoring rate: .5				Censoring rate: .75			
	Mean	StDev	Min	Max	Mean	StDev	Min	Max	Mean	StDev	Min	Max
MKSE	0.14	0.10	0.01	0.64	0.20	0.12	0.01	0.69	0.65	0.55	0.01	2.81
MLE	0.13	0.09	0.00	0.60	0.17	0.12	0.00	0.64	0.34	0.26	0.01	1.82

TABLE 16. Distances from estimates ($\hat{\lambda}, \hat{\kappa}$) to true values: LogNorm(2, 0.5)
500 datasets of size n=200

	Censoring rate: .25				Censoring rate: .5				Censoring rate: .75			
	Mean	StDev	Min	Max	Mean	StDev	Min	Max	Mean	StDev	Min	Max
MKSE	0.05	0.03	0.00	0.16	0.07	0.04	0.00	0.26	0.65	1.40	0.01	11.97
MLE	0.04	0.02	0.00	0.12	0.05	0.03	0.00	0.19	0.07	0.05	0.00	0.29