NPS-OR-04-004

NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

Computational Support for the Study of Lifetime Distribution Characteristics

by

Robert R. Read

April 2004

Prepared for: Operations Research Department
Naval Postgraduate School
Monterey, CA 93943-5219

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** April 2004	**3. REPORT TYPE AND DATES COVERED** Technical Report

4. TITLE AND SUBTITLE: Computational Support for the Study of Lifetime Distribution Characteristics	**5. FUNDING NUMBERS**
6. AUTHOR(S) Robert R. Read	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER** NPS-OR-04-004
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** N/A

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

The report develops and makes available programs that treat the support functions of a set of survivor distributions: Weibull, Gamma, and Lognormal. The issues of model characterization functions, maximum likelihood estimation, bias reduction, and censored samples are treated generally. The general material is made explicit for the distributions named. It features open code, allowing the user to pursue plans of his own.

The paper also contains some items of more general interest. First, a technique is developed that offers substantial reduction in the dependence of the initialization values for the success of the Newton-Raphson iteration technique. Second, high-precision, numerical analysis techniques are developed for the parallel computation of several derivatives of the Incomplete Gamma function.

14. SUBJECT TERMS Reliability, Censored data, Lifetime Distributions, maximum likelihood estimation, confidence regions	**15. NUMBER OF PAGES** 92
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

This report was prepared for and funded by the Operations Research Department, Naval Postgraduate School, Monterey, CA 93943-5219.

This report was prepared by:

ROBERT R. READ
Professor of Operations Research

Reviewed by:

LYN R. WHITAKER
Associate Chairman for Research
Department of Operations Research

Released by:

JAMES N. EAGLE
Chairman
Department of Operations Research

LEONARD A. FERRARI, Ph.D.
Associate Provost and Dean of Research

Computational Support for the Study of Lifetime Distribution Characteristics

R. R. Read

ABSTRACT

The report develops and makes available programs that treat the support functions of a set of survivor distributions: Weibull, Gamma, and Lognormal. The issues of model characterization functions, maximum likelihood estimation, bias reduction, and censored samples are treated generally. The general material is made explicit for the distributions named. It features open code, allowing the user to pursue plans of his own.

The paper also contains some items of broader interest. First, a technique is developed that offers substantial reduction in the dependence of the initialization values for the success of the Newton-Raphson iteration technique. Second, high-precision, numerical analysis techniques are developed for the parallel computation of several derivatives of the Incomplete Gamma function.

1. Introduction

Much has been written on life length distribution applications—especially those in the field of reliability. The first course textbooks usually cover the ground well, but the examples often are shallow because there are so few distributions that can readily be managed, supported only by the calculus. The computational aspects are extremely easy when dealing with the exponential distribution, and they become quite difficult when using other common distributions. This situation creates difficulties in the teaching of first courses, the writing of textbooks, and, of course, in the execution of research. The use of the exponential distribution becomes overworked and dull, and its sole use masks the dangers that can be encountered.

The present report offers useful relief to these problems. Algorithms and S-Plus code are developed so that the user can explore on the computer the consequences of using the Weibull, Gamma, and Lognormal distributions. Also, a general mathematical structure is presented so that a programmer can extend and deal with other distribution families.

At the same time, we are beginning to see products of this type from other software writers. The Splida system by Meeker and Alvarez has completed beta testing and has appeared on the market. This system appears to be very comprehensive. It is written in S-Plus and features a large number of specialized drop-down dialog menus. A product named Reliasoft is already on the market. It too requires that the user fill out dialog boxes. The present work is far less ambitious, but contains open S-Plus code that allows the user to deal directly with issues of his own choosing and to check the precision of his solutions. It contains a number of command line functions that a user can integrate into his own specialized problem-solving package. (Meeker offers to supply computer code upon request; it is not known whether this includes the supporting algorithmic analysis.)

Attention is restricted to the continuous time life length variables. General formulae are developed for these. A first goal is to compute and graph five basic support functions, namely,

f(t)	=	probability density functions (pdf),
S(t)	=	survivor function,
h(t)	=	hazard function,
H(t)	=	cumulative hazard function,
LL(t)	=	expected residual life.

The first two often are included in the standard statistical software packages. The others usually require the practitioner to exercise some calculus and programming. The user should be familiar with the shapes of these functions and how they vary from model to model. In fact, model selection is often a judgment call rather than a statistical exercise. In terms of these support functions, some model comparative graphing can be quite valuable.

Complete samples are those for which the testing process is not finished until all objects have realized their entire lifetimes. In these cases, the stock statistical methods are utilized. One usually calls upon maximum likelihood to serve for parameter estimation. These have well studied properties. They are not easy to find and often iterative methods are created for this purpose. Seldom are these available in the standard statistical software packages. The report contains examples of how one can generate such methods. The large sample theory for maximum likelihood estimators is well established and it is used to create confidence regions for the parameters.

Generally, maximum likelihood estimators are known to be biased. This issue has not received great attention. The methods for dealing with it are difficult and the effects are not believed to be great, except possibly for small sample sizes. But the report does consider a first order bias reduction method and applies it to the models treated. At least the reader is allowed to view the nature of the problem.

Recent times have seen the development of methods to manage censored life length data, i.e., data collection plans that do not collect full life length information on all subjects. We consider right-censored data only, i.e., each subject is either observed to expire or has survived beyond a known point. Plans that await a complete data set are often too expensive. The maximum likelihood point estimation schemes are available in concept, but the implementation is often intricate and requires special methods. The iteration functions utilized are advertised as requiring high quality initialization points. A method is proposed that appears to give substantial relief to this problem and shows great promise at this point in time.

When dealing with the gamma distribution in the censored case, one must face the computation of the derivatives of the incomplete gamma function. Methods for accomplishing this are developed.

The report is organized as follows. Section 2 contains the general mathematical structure for the support functions, for complete samples, for a bias reduction method and for censored samples. This is followed, in Section 3, by the presentation of some computations and graphs typical of the capabilities that are being supported. This part does much to illustrate the use of the programs and the kinds of issues that may be addressed. The reader who is well versed in the issues will find this chapter most useful as it identifies the programs and illustrates their use. Sections 4, 5, and 6 treat the mathematical details for the three popular distributions: Weibull, Gamma, and Lognormal, respectively. Explicit formulae are developed for the characterizing support functions, the treatment of complete samples with maximum likelihood, bias reduction, and methods for censored samples.

Two data sets have been enlisted to test the programs in Section 3. The Lieblein and Zelen (1956) ball-bearing data is utilized for complete samples. It consists of 23 failure times of ball bearings measured in millions of revolutions. It has been exploited by [Meeker and Escobar, p. 4] to illustrate the model-fitting problem. A similar use is presented in Section 3. The data are

BallB: 17.88 28.92 33.00 41.52 42.12 45.60 48.40 51.84 51.96 54.12 67.80
68.64 68.64 68.88 84.12 93.12 98.64 105.12 105.84 127.92 128.04 173.40

A second data set was chosen to test the methods that involve censoring. It is the set of survival times for multiple myeloma patients from [Lawless, p. 337] consisting of 48 complete lifetimes, x and 17 right-censored values, t. Both are measured in months. The data are

myeloma:
x: 1 1 2 2 2 3 5 5 6 6 6 6 7 7 7 9 11 11 11 11 11 13 14
15 16 16 17 17 18 19 19 24 25 26 32 35 37 41 42 51 52 54 58 66 67 88
89 92

t: 4 4 7 7 8 12 11 12 13 16 19 19 28 41 53 57 77

Fuller details of implementation of the methods are contained in five appendices.

They treat specialized subjects and each contains supporting S-Plus listings. Their contents are:

Appendix A. Asymptotic Expansions for the Polygamma Functions.

Appendix B. Analysis for Computational Support of the Weibull Distribution.

Appendix C. Implementation of the General Censored Data Estimation Scheme.

Appendix D. Derivatives of the Incomplete Gamma Function.

Appendix E. S-Plus Listings of Miscellaneous Code.

The programs can be obtained in electronic form by contacting the author.

2. General Formula

The specialized programs follow a general structure, which is presented here. A number of computational situations are envisioned. Only the continuous case is treated. First, we deal with important support functions used in reliability and lifetime analysis. This includes a visit to the optimal planned replacement policy formulae. Second, the basic maximum likelihood estimation procedures for complete samples is presented. This includes iteration schemes for finding the solution, the information matrix, and the construction of asymptotic confidence regions. This also includes the method for reducing the sensitivity of the initialization values. Third, the development of a general technique for bias reduction is considered. Fourth, the general structure for dealing with maximum likelihood estimation for right-censored samples is presented.

Code is needed for the exploration of the properties of various models. Relationships among the various functions are summarized [Leemis, p. 55]. Those that are most useful in the present work are listed. A continuous life length random variable X has a probability density function, f(t) and a cumulative distribution function F(t). From these one can characterize others.

2a. Model Characterization Functions

The survivor function:

$$S(t) = \int_t^\infty f(u)du = 1 - F(t) = \Pr\{X \geq t\}. \tag{2.1}$$

The hazard function, age specific failure rate:

$$h(t) = f(t)/S(t) = H'(t). \tag{2.2}$$

The cumulative hazard function:

$$H(t) = \int_0^t h(u)du = -\ln[S(t)]. \tag{2.3}$$

The integrated survivor function:

$$SS(t) = \int_t^\infty S(u)du. \tag{2.4}$$

The mean residual life:

$$LL(t) = \int_t^\infty u\,f(u)du\,/\,S(t) = SS(t)/S(t) = E\{X - t \mid X > t\}. \tag{2.5}$$

2b. Cost Calculation of Planned Replacement Policies

Let us examine the effect of model choice when a planned replacement policy is considered. Such policies can be advantageous when dealing with an IFR system and the cost of a planned replacement is lower than the cost of replacing at failure. The structure of the calculation is developed in [Barlow and Proschan].

Let $k > 0$ be the cost of a planned replacement, but if the replacement must happen because of a device failure then an additional cost c is incurred to be added to k. Let R_n be the cost of the nth replacement, and X_n is the lifelength of the nth unit placed in service. The system $\{Xn\}$ is assumed to be i.i.d. The cost is random because it depends on the type of replacement.

$$R_n = k \text{ if } X_n > t \text{ and } R_n = k + c \text{ if } X_n \leq t. \tag{2.6}$$

Now we can describe the cost for the interval (0, s].

$$Z(s) = \sum_1^{N(s)} R_j$$

This cost is a random variable, but the long-term average cost stabilizes. That is

$$\frac{Z(s)}{s} \longrightarrow \frac{E\{R_1\}}{E\{X_1\}} \text{ as } s \to \infty.$$

The main formula for long-term average cost using replacement policy t, i.e., replace at age t, is

$$C(t) = \frac{k + cF(t)}{\int_0^t S(w)dw}, \tag{2.7}$$

where S(w) is the survivor function of the distribution F. The denominator of this expression may also be computed using $\mu - SS(t)$ and $\mu = E\{X\} = SS(0)$. See [Barlow and Proschan, 1981] or [Prentice and Kalbfleisch].

2c. The Likelihood Equations and Solution Technique; Structural Overview

The structure of the likelihood function has different appearances depending upon whether the data are complete or censored. There are a few general points that apply to either case. These are presented first, followed by representations of how the details can change depending upon the two cases. The parameter θ may be a vector of several components. Let $\ell(\theta)$ be the likelihood function, and

$$L(\theta) = \log[\ell(\theta)] \tag{2.8}$$

be its logarithm. We are concerned with the smooth settings in which the maximum likelihood estimates are found using gradient methods. The partial derivative with respect to the individual members of θ, when the particular subscript(s) plays no immediate role will be marked as L_θ; $L_{\theta\theta}$; and L_{θ_1,θ_2}; the latter case representing any pair of the mixed partial derivatives.

The partial derivative of L with respect to θ has expected value equal to zero and the expected square of this quantity is the negative of the expected value of the second partial of the log likelihood. The requisite smoothness conditions for interchanging the expectation operation with the appropriate partial derivatives are presumed.

The matrix $\{L_{\theta_i\theta_j}\}$ is known as the Hessian and the negative of its expected value is the (Fisher) information matrix. Call the former H, the latter nI_0 (use I_0 for the information in a single observation), and $L_{\boldsymbol{\theta}}$ the vector of partials of the log likelihood. Often the system of equations $L_{\boldsymbol{\theta}} = \mathbf{0}$ must be solved by iterative methods. This is done using a Newton-Raphson type technique. Two options are presented:

$$\theta^{(k+1)} = \theta^{(k)} - H^{-1} L_\theta \quad \text{and} \quad \theta^{(k+1)} = \theta^{(k)} + (nI_0)^{-1} L_\theta. \tag{2.9}$$

Termination of the iteration occurs when there is no change in the maximum value. Both methods require a good initialization, $\theta^{(0)}$. To some, there appears to be empirical evidence to use the second choice, when feasible, but the calculation of I_0 is often difficult.

Since convergence of the iteration function is sensitive to the initialization, $\boldsymbol{\theta}^{(0)}$, the author found relief from this problem by utilizing a golden section search along the segment $(\theta^{(k)}, \theta^{(k+1)})$. Typically, the iteration steps over-swing the maximum in this direction, often by a factor of two and sometimes by a factor of 10. It can pay to seek a local maximum in this direction. The method is implemented as follows. Let

$$\begin{aligned} \boldsymbol{\theta}_1 &= \boldsymbol{\theta}^{(k)} \;;\; \boldsymbol{\theta}_4 = \boldsymbol{\theta}^{(k+1)} \\ \boldsymbol{\theta}_2 &= 0.618\,\boldsymbol{\theta}_1 + 0.382\,\boldsymbol{\theta}_4 \\ \boldsymbol{\theta}_3 &= 0.382\,\boldsymbol{\theta}_1 + 0.618\,\boldsymbol{\theta}_4 \end{aligned} \tag{2.10}$$

and compute L at these four points. If there is a single local maximum over the segment, then it can be found by an iterative scheme:

If $L(\boldsymbol{\theta}_1)$ is the largest of the four, replace $\boldsymbol{\theta}_4 \leftarrow \boldsymbol{\theta}_3$ and return to (2.10)
If $L(\boldsymbol{\theta}_4)$ is the largest of the four, replace $\boldsymbol{\theta}_1 \leftarrow \boldsymbol{\theta}_2$ and return to (2.10).

Failing these,

If $L(\boldsymbol{\theta}_2)$ is smaller than $L(\boldsymbol{\theta}_3)$, then make the replacement $\boldsymbol{\theta}_1 \leftarrow \boldsymbol{\theta}_2$
If $L(\boldsymbol{\theta}_2)$ is larger than $L(\boldsymbol{\theta}_3)$, then make the replacement $\boldsymbol{\theta}_4 \leftarrow \boldsymbol{\theta}_3$
and return to (2.10).

Repeat the process until there is no change. Then go back to the Newton-Raphson type scheme and continue in a new direction. This golden section augmentation will slow when overly stringent convergence criteria are used. But is often preferable to the seeking of a better initialization.

2d. Likelihood System; Complete Samples

Let $X_1, X_2, \cdots, X_n$ be a random sample of life length random variables each having pdf $f(x;\theta)$ and survivor function $S(t;\theta)$. Note how the parameter θ is now included in the notation; it may be multidimensional. The likelihood function is expressed

$$\mathrm{lik}(\theta) = \prod_{i=1}^{n}[f(x_i;\theta)] \quad \text{and} \quad L = \sum_{i=1}^{n}\log[f(x_i;\theta)]. \tag{2.11}$$

The partial derivative of L, the log likelihood, with respect to θ has expected value equal to zero and that the expected square of this quantity is the negative of the expected value of the second partial of the log likelihood. The requisite smoothness conditions for interchanging the expectation operation with the requisite partial derivatives are presumed. The subscript θ is used to denote partial derivative, and the format aspects of the partial of the log likelihood take the appearance

$$L_\theta = \sum_{i=1}^{n}\frac{f_\theta(x_i;\theta)}{f(x_i;\theta)} \tag{2.12}$$

and the second order partial derivatives

$$L_{\theta\theta} = \sum_{i=1}^{n}\left\{\frac{f_{\theta\theta}(x_i;\theta)}{f(x_i;\theta)} - \left[\frac{f_\theta(x_i;\theta)}{f(x_i;\theta)}\right]^2\right\}, \tag{2.13}$$

where the double subscript θθ refers to a common component of the vector **θ**. Any mixed partial derivative of second order has the form

$$L_{\theta_1\theta_2} = \sum_{j=1}^{n}\left\{\frac{f_{\theta_1\theta_2}(x_i;\theta)}{f(x_i;\theta)} - \frac{f_{\theta_1}(x_i;\theta)f_{\theta_2}(x_i;\theta)}{f^2(x_i;\theta)}\right\}. \tag{2.14}$$

One must develop these quantities for each specific model. Should the system of equations for the mle's be nonlinear, one may use the iteration schemes described above.

The negative expectation of $\{L_{\theta_1\theta_2}\}$ is nI_0, where I_0 is the single observation information matrix. A version of the multivariate central limit theorem says that

$$\sqrt{n}(\hat{\theta}-\theta) \approx MVN(0, I_0^{-1}). \tag{2.15}$$

It follows that

$$(\hat{\theta}-\theta)' nI_0 (\hat{\theta}-\theta) \approx \chi^2_{(k)}, \text{ where k is the dimension of } \boldsymbol{\theta}. \tag{2.16}$$

This distributional point will be exploited in order to find joint confidence ellipses for the parameters.

Approximate joint confidence regions for the parameters can also be obtained using

$$G^2 = -2[L(\theta) - L(\hat{\theta})] \approx \chi^2_{(k)} \tag{2.17}$$

This too will be used for some comparisons.

2e. Bias Reduction

Maximum likelihood estimates are known to be biased generally. It is appropriate to reduce this bias. The technique will be presented using the notation for complete samples. Explicit relevant formulae will be generated where the explicit models are discussed. The use of the technique under censored sampling follow these same ultimate formulae, but the censored case has a much more difficult likelihood function and the development of the requisite computational formulae must await another time. It does present a ripe area for application and study.

This section expands upon the bias reduction analysis of mle estimates that appears in [Cox and Hinkley, p. 309]. The basic idea is to use a third order expansion of the log likelihood about the mle. The technique deals with a single component of the parameter vector and is expected to work best when the estimators of those components are not strongly correlated.

The log likelihood function L is a sum of n terms, where n is the sample size. Consider a single parameter θ. The score of an observation is the partial derivative (of one term) of the log likelihood with respect to θ. The score of the ith term (without subscript as it plays no role) to that sum will be called U; the 1st and 2nd partial derivatives will be denoted U' and U", respectively. Use the dot subscript notation to denote summation over the n terms. Thus,

$$U = \frac{\partial \ln(f)}{\partial\theta} = \frac{f_\theta}{f}; U' = \frac{f_{\theta\theta}}{f} - [\frac{f_\theta}{f}]^2; U'' = \frac{f_{\theta\theta\theta}}{f} - 3\frac{f_{\theta\theta}f_\theta}{f^2} + 2[\frac{f_\theta^2}{f^3}] \tag{2.18}$$

and the expansion may be expressed

$$0 = U.(\hat{\theta}) = U.(\theta) + (\hat{\theta} - \theta)U.'(\theta) + \frac{1}{2}(\hat{\theta} - \theta)^2 U.''(\theta) + O_p(n^{-1/2}) \tag{2.19}$$

Taking expectations through this expression, we obtain

$$E(\hat{\theta}-\theta)E\{U.'(\theta)\} \ +\mathrm{cov}\{\hat{\theta}, U.'(\theta)\} + \frac{1}{2}E(\hat{\theta}-\theta)^2 E\{U.''(\theta)\}$$

$$+\frac{1}{2}\mathrm{cov}\{(\hat{\theta} - \theta)^2,\ U.''(\theta)\} = O(n^{-1/2}). \qquad (2.20)$$

At this point introduce the notation, for a single observation,

$$\kappa_{r,\,s}(\theta) = E\{[U(\theta)]^r[U'(\theta)]^s\}, \qquad (2.21)$$

and to note that

$$E\{U''(\theta)\} = -3\kappa_{1,\,1}(\theta) - \kappa_{3,\,0}(\theta) \qquad (2.22)$$

and $E\{U.''(\theta)$ is n times the above amount.

In order to justify the above expression and what follows one should keep in mind that

$$0 = E(\frac{f_\theta}{f}) = E(\frac{f_{\theta\theta}}{f}) = E(\frac{f_{\theta\theta\theta}}{f})$$

and $\mathrm{cov}(\hat{\theta}, U'(\theta)) = O(1/n)$ and $\mathrm{cov}[(\hat{\theta} - \theta)^2 U''(\theta)] = o(1/n)$.

Further, $E[U.'(\theta) = n\,i(\theta)$, where $i(\theta)$ is the information scalar (one by one matrix) and $E(\hat{\theta} - \theta)^2 \approx 1/n\,i(\theta)$.

The bias function is $b(\theta) = E[\hat{\theta} - \theta]$. Now let's take the expectation of (2.17).

$$b(\theta)\ ni(\theta) + 1/(2\ i(\theta))\ [-3\kappa_{1\,1} - \kappa_{3\,0}] = o(1/n)$$

and this yields the approximation

$$b(\theta) \cong \frac{3\kappa_{1\,1} + \kappa_{3\,0}}{2n\,i^2(\theta)} \qquad (2.23)$$

(This is at variance with [Cox and Hinkley, p. 310, Equation (35)]).

The bias reduction is executed by using $b(\hat{\theta})$. I.e., the reduced biased estimator is

$$\theta^* = \hat{\theta} - b(\hat{\theta}). \tag{2.24}$$

2f. Likelihood System; Censored Samples

Let $X_1, X_2, \cdots, X_n$ be a random sample of life length random variables having pdf $f(x;\theta)$ and survivor function, $S(t;\theta)$.

Type I censoring. Testing of item i will stop at t_i if the item has not failed by that time.

Type II censoring. All testing stops when the rth item has failed, i.e., at $X_{(r)}$.

$$\text{Let } \delta_i = 1 \text{ if } X_i \text{ is} \le \text{(it's censoring time, Type I) } t_i \text{ or } X_{(r)} \text{ (for Type II)} \\ \qquad = 0 \text{ o.w.} \tag{2.25}$$

The likelihood function can be expressed in a common format for the two kinds of censoring,

$$\text{lik}(\theta) = \prod_{i=1}^{n} [f(x_i;\theta)]^{\delta_i} S(t_i;\theta)^{1-\delta_i} \tag{2.26}$$

Our first goal is to go through the formalities of showing that the partial derivative of the log likelihood has expected value equal to zero and that the expected square of this quantity is the negative of the expected value of the second partial of the log likelihood. The requisite smoothness conditions for interchanging the expectation operation with the requisite partial derivatives are presumed. Let the logarithm of the likelihood function be

$$L(\theta) = \sum_{i=1}^{n} [\delta_i \ln(f(x_i;\theta) + (1-\delta_i)\ln(S(t_i;\theta)], \tag{2.27}$$

and using the subscript θ to denote partial derivative, the format aspects of the partial of the log likelihood take the appearance

$$L_\theta = \sum_{i=1}^{n} [\delta_i \frac{f_\theta(x_i;\theta)}{f(x_i;\theta)} + (1-\delta_i)\frac{S_\theta(t_i;\theta)}{S(t_i;\theta)}], \tag{2.28}$$

and the second partial derivatives

$$L_{\theta\theta} = \sum_{i=1}^{n} \delta_i \{\frac{f_{\theta\theta}(x_i;\theta)}{f(x_i;\theta)} - [\frac{f_\theta(x_i;\theta)}{f(x_i;\theta)}]^2\}$$

$$+ \sum_{i=1}^{n} (1-\delta_i)\{\frac{S_{\theta\theta}(t_i;\theta)}{S(t_i;\theta)} - [\frac{S_\theta(t_i;\theta)}{S(t_i;\theta)}]^2\} \tag{2.29}$$

$$L_{\theta_1\theta_2} = \sum_{j=1}^{n} \delta_i [\frac{f_{\theta_1\theta_2}(x_i;\theta)}{f(x_i;\theta)} - \frac{f_{\theta_1}(x_i;\theta) f_{\theta_2}(x_i;\theta)}{f^2(x_i;\theta)}]$$

$$+\sum_{j=1}^{n}(1-\delta_i)[\frac{S_{\theta_1\theta_2}(t_i;\theta)}{S(t_i;\theta)} - \frac{S_{\theta_1}(t_i;\theta)S_{\theta_2}(t_i;\theta)}{S^2(t_i;\theta)}]. \qquad (2.30)$$

The mle's can be found by setting (2.28) equal to zero, and the Hessian can be computed from (2.29) and (2.30).

The expected values of these quantities involve terms related to the form

$$\int_0^t f(x;\theta)\, dx + S(t;\theta) = 1, \qquad (2.31)$$

which when differentiated respect to θ, produce the useful structures

$$\int_0^t f_\theta(x;\theta)dx + S_\theta(t;\theta) = 0; \quad \int_0^t f_{\theta\theta}(x;\theta) = 0. \qquad (2.32)$$

From this we formally show that the expected value of (2.28) is zero, and for Equation (2.29) we see that

$$-E\{L_{\theta\theta}\} = E\{\sum_{i=1}^{n} \delta_i [\frac{f_\theta(x_i;\theta)}{f(x_i;\theta)}]^2\} + \sum_{i=1}^{n} E(1-\delta_i)[\frac{S_\theta(t_i;\theta)}{S(t_i;\theta)}]^2$$

and

$$E\{L_{\theta_1}L_{\theta_2}\} = \sum_{i=1}^{n} E\{\delta_i \frac{f_{\theta_1}(x_i;\theta)f_{\theta_2}(x_i;\theta)}{[f(x_i;\theta)]^2}\} + \sum_{i=1}^{n} E(1-\delta_i)\frac{S_{\theta_1}(t_i;\theta)S_{\theta_2}(t_i;\theta)}{[S(t_i;\theta)]^2}$$

and $\qquad E\{L_{\theta_1}L_{\theta_2}\} = -E\{L_{\theta_1\theta_2}\}.$

The maximum likelihood estimates will be the same regardless of the type of censoring. However, the effect of censoring upon the information matrix and bias reduction methodology does change with the type of censoring.

3. Computational Studies

This section is likely the most useful to the practitioner. It shows how to use the programs and suggests many kinds of exploratory computations and graphs. The contents are partitioned as follows.

3a. Comparison between the Exponential and Lognormal distributions.
3b. Comparison between the Weibull and Gamma distributions.
3c. Comparison of planned replacement rules for several models.
3d. Methods for dealing with the Weibull distribution; complete samples; bias reduction; censored samples.
3e. Methods for dealing with the Gamma distribution; complete samples; bias reduction; censored samples.
3f. Methods for dealing with the Lognormal distribution; complete samples; bias reduction; censored samples.

3a. Comparison of the Exponential and Lognormal Models

The exponential distribution is a favorite because of its simplicity. The lognormal distribution also appears, being derivable from fairly plausible assumptions about certain types of failure processes. [Breiman; *Statistics*, 1973, Houghton-Mifflin, p. 197] has drawn attention to the idea that these two distributions can be adequately fitted to some failure time data, and for small samples it is difficult to discriminate between the two. In this example, the chi square goodness of fit statistic (D) is 6.2 with 4 degrees of freedom for the exponential fit, estimated mean $= 41.1$. On the other hand, the lognormal distribution produces a D value of 7.5, for $\mu = 3.3$ and $\sigma = 1.1$. His point is that the chi squared procedure has little discrimination power, even for a sample of size 50. In most cases, the choice between the two can be resolved by comparing the two q-q plots.

Figure 3.1 makes another comparison of the two distributions, this time using Exp(1) and Lognormal(0, 1). The density functions appear similar, but the other support function allows one to be more discriminating as to their properties. Generally, the lognormal hazard function has the shape of an inverted "U." This is implausible for most situations. In spite of this unattractive feature, it has been used in a number of diverse situations. See [Lawless, p. 24].

Since the hazard function for the lognormal is below that of the exponential the former has a greater survivability and a longer residual life.

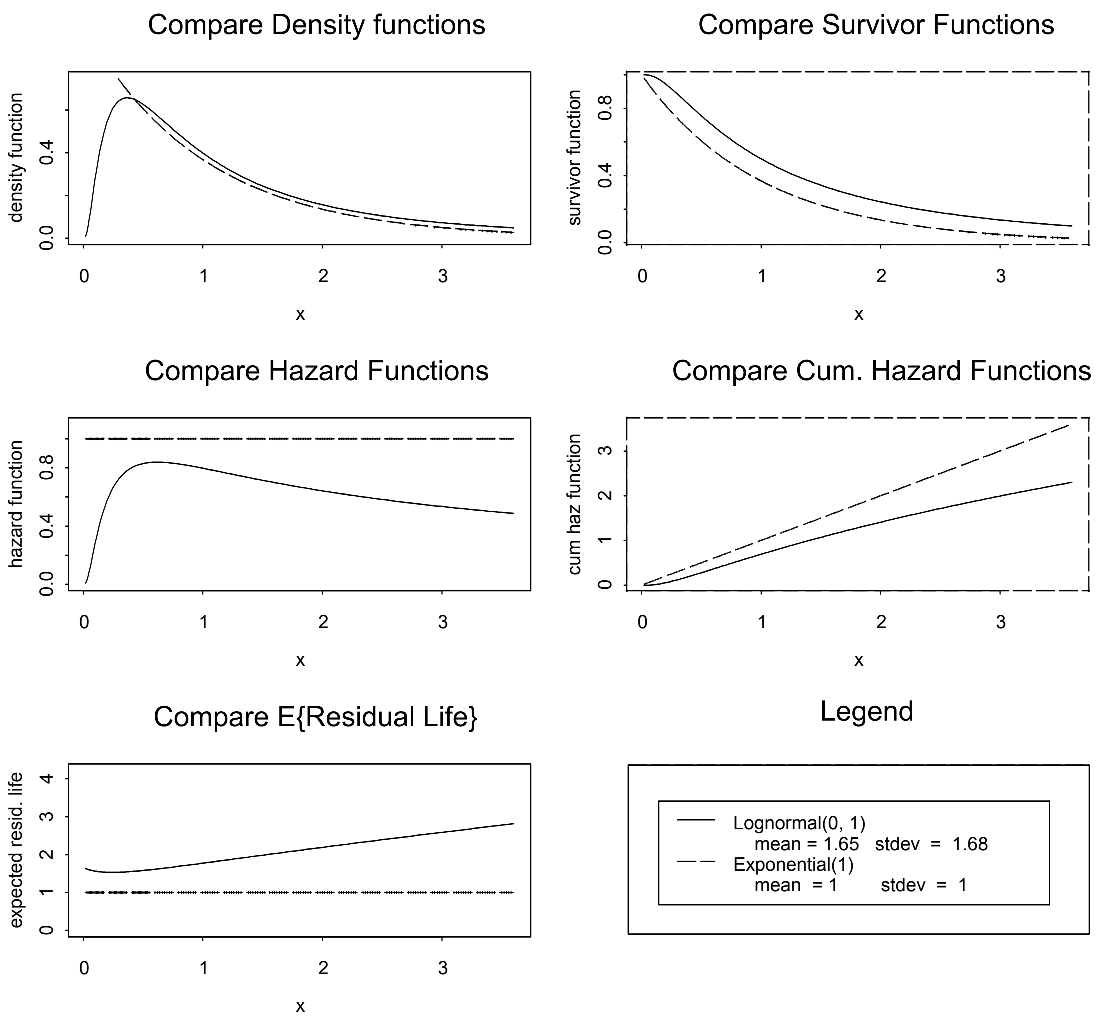

Figure 3.1
Behavior of the Lognormal Model compared with the Exponential Model.

3b. Comparison of the Weibull and Gamma Distribution Models

The Weibull distribution is a common choice for a survival distribution, as is the gamma distribution. The former is the more popular, largely because the hazard function follows a power law; decreasing for the shape parameter α being less than one, and increasing for the parameter being more than one. The hazard function for the gamma law is also DFR (decreasing failure rate) for the shape parameter smaller than one and increasing (IFR) when it is larger than one; but for large values of the variate it approaches an asymptote. Thus, there is a serious choice to be made between these two laws even though their density functions are quite similar. They are both unimodal and skewed positively. It is difficult to discriminate between the two based on complete samples. The distinctions are in the tails. Some comparisons between the two follow.

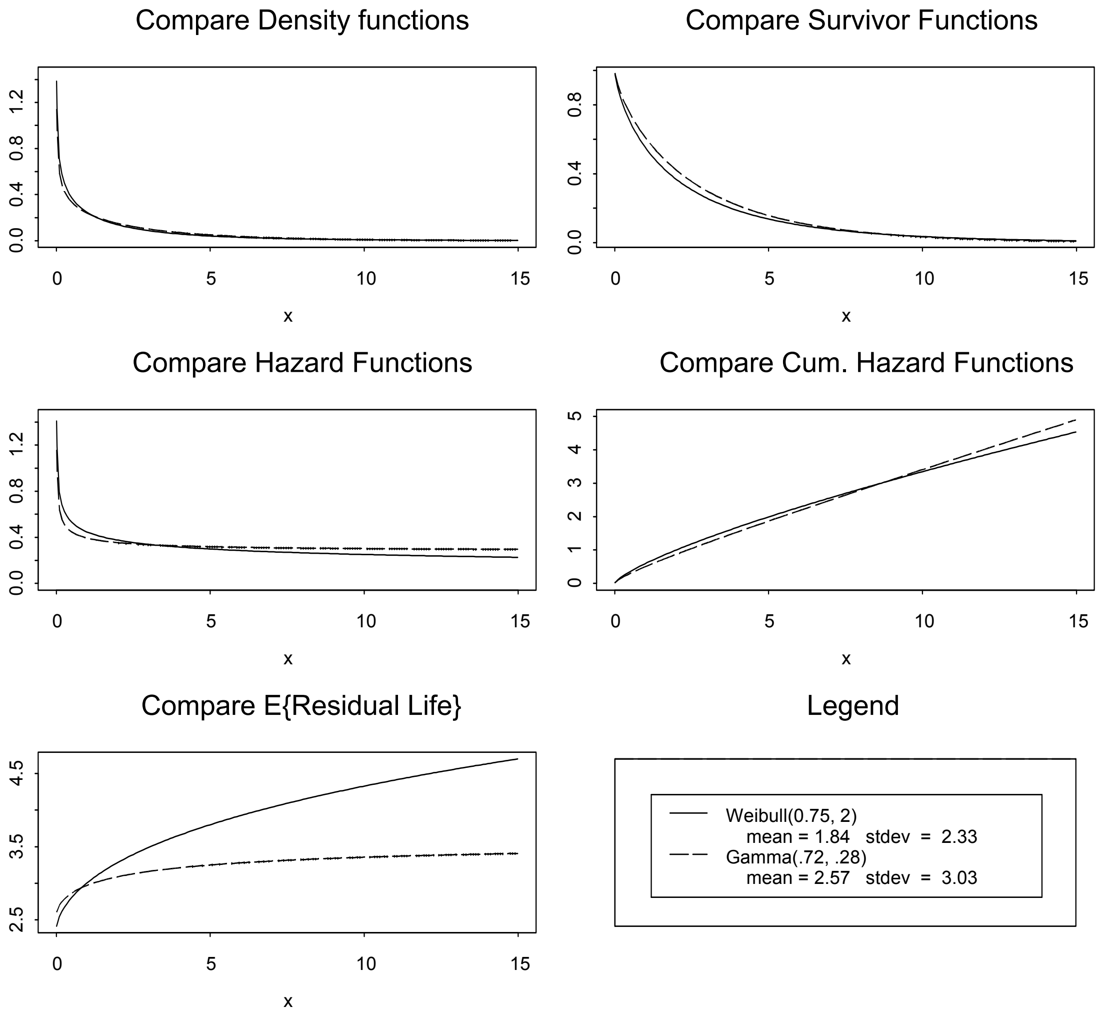

Figure 3.2
Comparison of the Weibull and Gamma Distribution Models; DFR Case.

Figure 3.2 makes the comparison between the Weibull and gamma distributions in a decreasing failure rate case. The Weibull (α = 0.75, β = 2) case was selected. A random sample of size 200 was simulated and the parameters for fitting a gamma (α, λ) distribution were estimated using maximum likelihood. The result is in the legend of Figure 3.2. The shape parameter α is unrelated to its counterpart for the Weibull distribution, but they share the same rule for discriminating between IFR and DFR. The parameter λ is called the rate parameter.

Inspection of Figure 3.2 shows that it would be difficult to make the choice between these two families without looking closely at the tails. The effect of comparing the expected residual life functions shows the effect of a finite asymptote for the gamma distribution.

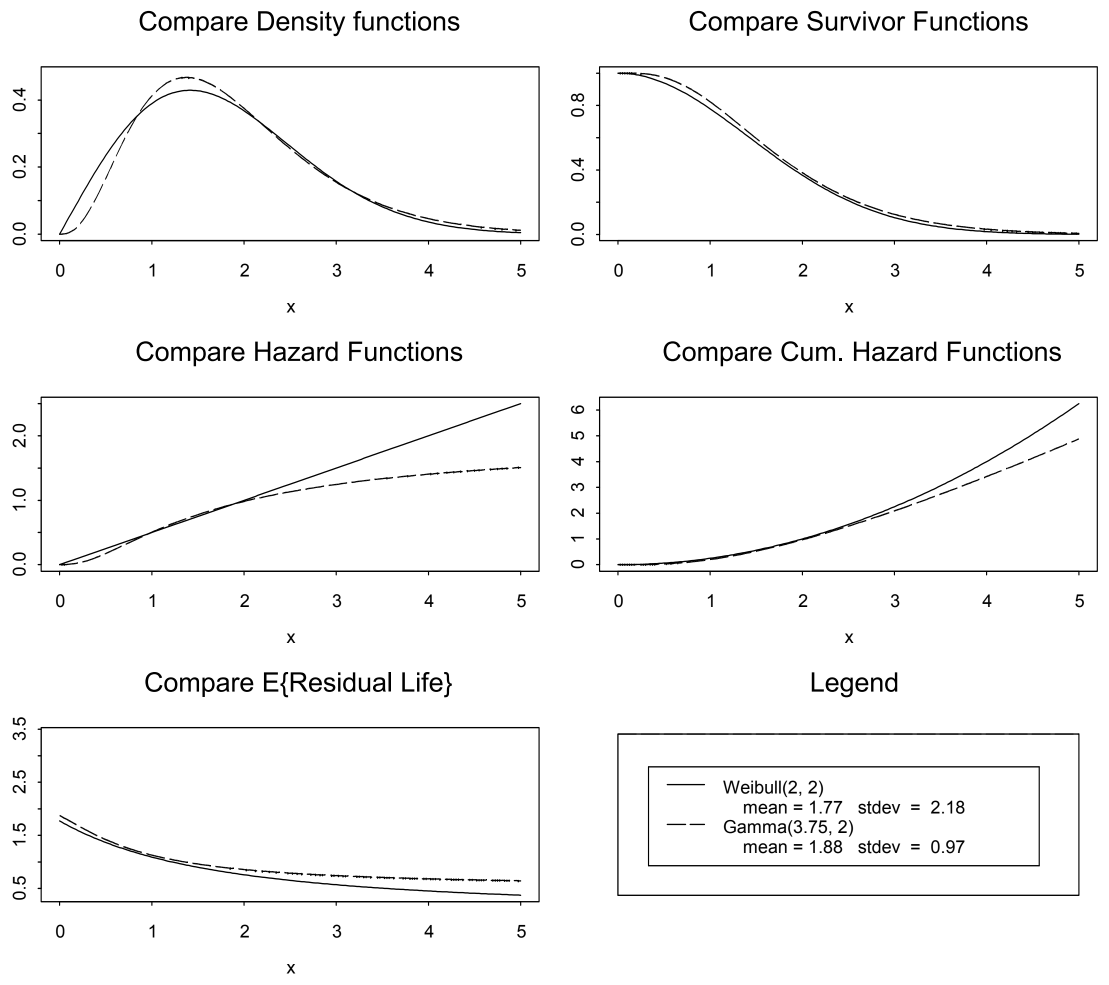

Figure 3.3
Comparison of the Weibull and Gamma Distribution Models; IFR Case.

An IFR comparison of these two distributions is made in Figure 3.3. The Weibull (2, 2) distribution was chosen (linear hazard function) and a reasonable convenient matching gamma (3.75, 2) distribution was selected after some experimentation. The story is much the same as in the previous case. The gamma hazard function has an ogive shape and exhibits a sharper separation from the Weibull hazard function. The lower hazard values translate in higher values for the expected residual life. Again, the effect of the asymptote is apparent.

S-Plus codes for the creation of these graphs are in Appendix E. See the functions `exp1.lnorm(), wei1.gam(), wei2.gam().`

3c. Cost of Planned Replacements

The material in Section 2b is used to continue our comparison of the Weibull and gamma distributions.

The relative cost curves for our two cases, Weibull (2, 2) and Gamma (3.75, 2), are superimposed for three different ratios of c/k, the additional cost c of the unplanned replacement to the basic cost of a planned replacement. The minimum cost policies are tabled:

Ratio, c/k	0.5	1.0	2.0
Min cost Weibull(2,2)	0.844	1.091	1.476
Min cost Gamma(3.75, 2)	0.800	1.043	1.379

The members of the paired cost curves track each other quite well. The costs are very flat after the initial drop off. The optimum costs are less than one standard deviation beyond the mean. The S-Plus code is cost.comp().

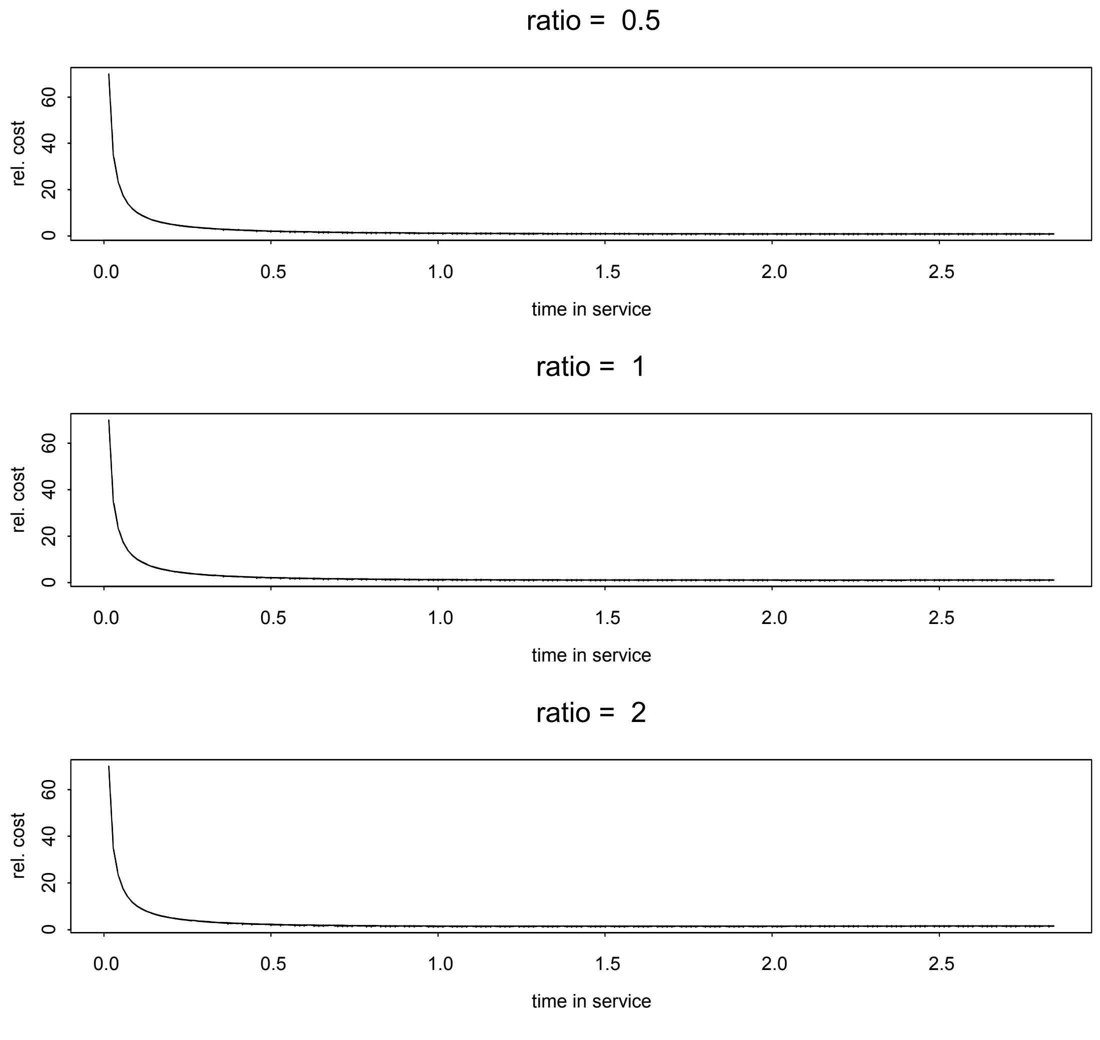

Figure 3.4
Planned replacement cost curves.

3d. Weibull Distribution

Three types of computations are illustrated: estimation for complete samples; bias reduction technique for complete samples; and estimation computations for censored samples. The model support functions are straightforward and need not be illustrated.

Complete Samples

The ball-bearing data is employed, see Section 1. It is a complete sample of 23 failure times measured in millions of revolutions. The call

```
weibull.est(BallB)                                        (3.1)
```

returns

```
shape:mle   scale:mle   samp size   shape:mm    scale:mm
2.101808    81.87422       23       2.01545     81.50336
```

and the interpretation is α = shape; β = scale; mle = maximum likelihood; and mm = method of moments. Using the mle values, the estimated information matrix is

$$I_0(\hat{\alpha},\hat{\beta}) = \begin{Bmatrix} 0.8676723 & 0.005163827 \\ 0.005163827 & 0.0006590095 \end{Bmatrix}. \tag{3.2}$$

One may graph an asymptotic confidence ellipse for (α, β), using the relationship

$$(\alpha-\hat{\alpha},\beta-\hat{\beta})n\hat{I}(\alpha-\hat{\alpha},\beta-\hat{\beta})' \approx \chi^2_{(2)} \tag{3.3}$$

for n sufficiently large. The function `ellipse (q,m,d,n0 = 100)` computes the upper and lower portions of the confidence ellipse. The inputs are q = $n\hat{I}_0$; m is the centering vector $(\hat{\alpha},\hat{\beta})$; d is the 100(1-α)th quantile of the chi square(2) distribution; and n_0 is the number of points to use in each quarter of the ellipse. The output is a $2n_0$ by 3 matrix. When one plots the superimposed graphs of each of columns 2 and 3 against column 1, the result is the ellipse. We an also plot an approximate confidence region based on the G^2 distribution, Equation (2.17). The calling sequence is

```
x <- seq(1.4, 2.9, .1); y <- seq(60, 105, 1)
zz <- Gsq.wei(BallB, x, y, m[1], m[2], n)                          (3.4)

plot(plot.wei[,1], plot.wei[,2], type="l", xlab="shape", ylab="scale",
        xlim = c(1.4, 2.9), ylim = c(60, 105))
lines(plot.wei[,1], plot.wei[,3])
contour(x, y, zz, nlevels = 1 ,v = d, add = T, lty = 3 ,labex = 0)
title(main = "90% Confidence Regions for Weibull Parameters")      (3.5)
```

The results are in Figure 3.5. The dashed curve is computed from the G^2 distribution.

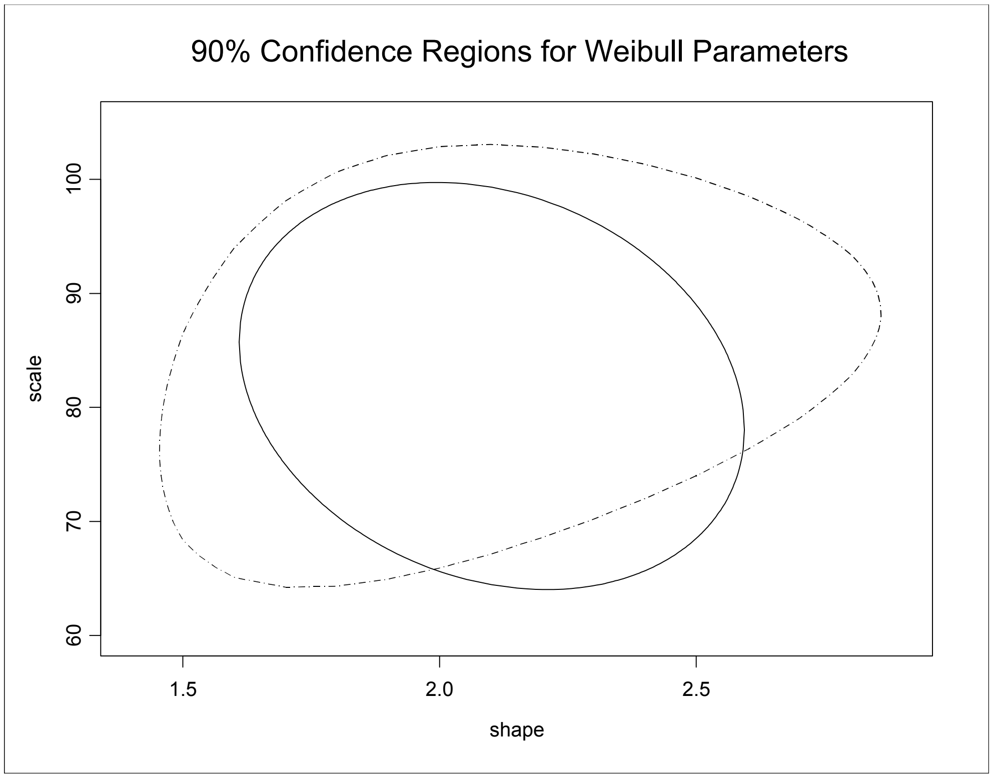

Figure 3.5

Bias Reduction

Using (2.24), (4.16) the formulae for bias reduction becomes

$$\alpha^* = \hat{\alpha}(1+\frac{4.309165}{2n}); \; \beta^* = \hat{\beta}(1+\frac{1+3/\hat{\alpha}}{2n\hat{\alpha}}). \qquad (3.6)$$

And when applied to the mle's found for the ball-bearing data, the reduced bias estimates become

$\alpha^* = 2.2987$, (mle $= 2.101808$)
$\beta^* = 83.92977$ (mle $= 81.87422$).

For a sharper comparison, let us use a Monte Carlo random sample from a specified Weibull distribution, say Weibull(1.5, 2). Let x <- rweibull(15, 1.5, 2). The estimation function returns

weibull.est(x) (3.7)

```
shape:mle   scale:mle   samp size   shape:mm    scale:mm
1.807413    1.720534       15       1.707669    1.707839
```

Use of (3.7) allows construction of the table

Parameter	**Actual**	**Max. Lik.**	**Bias Reduced**
α	1.5	1.807413	2.067028
β	2.0	1.720534	1.804933

Censored Samples

Let us treat the myeloma data set, per the Introduction, 65 observations, 17 of which are right censored; x <- myeloma[myeloma$del= =1,1] and t <- myeloma[myeloma$del= =0,1]. Begin with the finding of initial estimating values. Use the call[1]

```
th <- init.wei(x, t),                                        (3.8)
```

which returns: th

```
    u          b         alph        beta
3.949889   1.45214   0.6886389   51.92961
```

then the estimation function; use the call

```
Newt.wei(x, t, th[1:2]),                                     (3.9)
```

which returns

```
    u           b           alph         beta       flag
3.492412   0.9253385   1.080686   32.86512     0
```

(Note: The flag starts at one and is changed to zero once the bisection search option is invoked.) See Section 4.

3e. Gamma Distribution

Complete Samples

We use the parameterization (α, λ) where λ is the data parameter. Many prefer to use the scale parameter, $\beta = 1/\lambda$. Because of this, we carry the option in the example. Let us again use the ball-bearing data and illustrate the use of the programs. The immediate goal is to fit the gamma distribution and estimate the Information matrix. Model fit comparisons are made at the close of this section. Equation (5.18) is implemented in a function `gam.est()`. The application upon executing

```
th <-gam.est(BallB)                                          (3.10)
```

[1] The parameters (u, b) relate to the Extreme Value distribution; the bisection search method plays a role as well. These things are explained in Section 4.

produces the return

```
alpha-mle    lambda-mle    n    alpha-mm     lambda-mm
4.024706     0.05572775    23   3.710832     0.05138171
```

Note the output contains both the maximum likelihood estimator and the method of moments estimator. It follows that beta-mle = 17.94438 and beta-mm = 19.46218. The information matrix (see (2.16)), is estimated to be

$$I(\hat{\alpha},\hat{\lambda}) = \begin{Bmatrix} 0.2819 & -17.94 \\ -17.94 & 1295.959 \end{Bmatrix} \qquad I(\hat{\alpha},\hat{\beta}) = \begin{Bmatrix} 0.2819 & 0.0557 \\ 0.0557 & 0.0125 \end{Bmatrix}. \tag{3.11}$$

One may graph an asymptotic confidence ellipse for, say (α, β), using the relationship

$$(\alpha-\hat{\alpha},\beta-\hat{\beta})n\hat{I}(\alpha-\hat{\alpha},\beta-\hat{\beta})' \approx \chi^2_{(2)} \tag{3.12}$$

for n sufficiently large. The function `ellipse(q,m,d,n0 = 100)` computes the upper and lower portions of the confidence ellipse. The inputs are q = $n\hat{I}$; m is the centering vector $(\hat{\alpha},\hat{\beta})$; d is the 100(1-α)th quantile of the chi square(2) distribution; and n_0 is the number of points to use in each quarter of the ellipse. The output is a $2n_0$ by 3 matrix. When one plots the superimposed graphs of each of columns 2 and 3 against column 1, the result is the ellipse. For an example using the ball-bearing data: n = 23 and d = qchisq(0.9, 2); n_0 is 100. Setting plot.dat <- ellipse(q, m, d) and then

```
plot(plot.dat[,1], plot.dat[,2], type = "l")                (3.13)
lines(plot.dat[,1], plot.dat[,3])
```

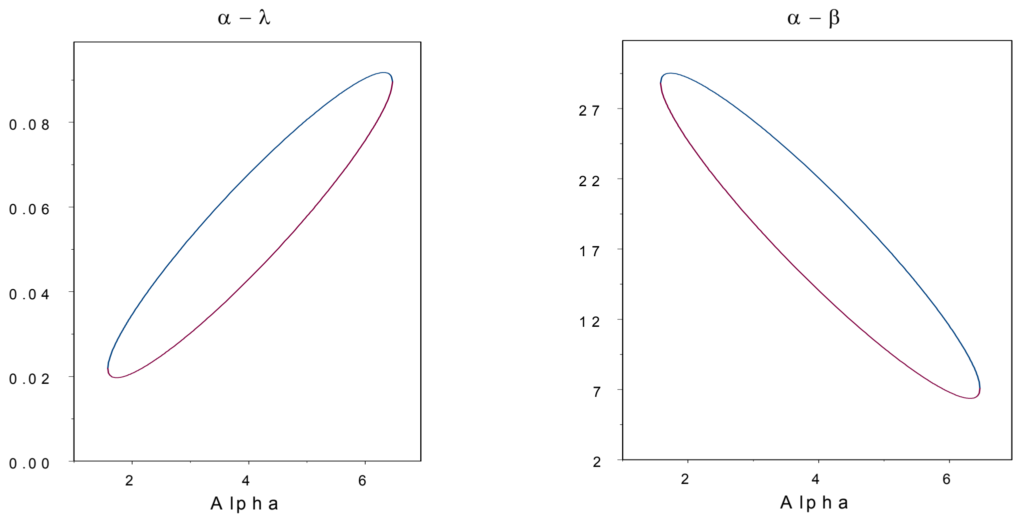

Figure 3.6
Gamma Parameters: 90% Confidence Ellipses.

The $\alpha-\lambda$ ellipse was constructed from the above code. The $\alpha-\beta$ ellipse was constructed from the former using $\beta = 1/\lambda$. It is remarkably close to the ellipse that would be based on the right-hand side of (3.12).

Bias Reduction

First, let us observe the effect of applying Equations (5.24), (5.29), and (5.34) to the ball-bearing data. The only formal assistance is offered in terms of the S-Plus function bias.gam(), which produces the constant C.

C <- bias.gam(th[1]) = -0.9936822 (3.14)

for use in (5.24). The results of all three bias reductions are

Ball-Bearing Data

Parameter	**Max. Lik**	**Bias Reduced**
α	4.02471	4.04631
λ	0.05573	0.05633
β	17.9444	17.9683

It may be more useful to test the method using a smaller sample drawn from a Gamma population with specified parameters. Accordingly, let us use

X <- rgamma(15, shape = 1.5, rate = 0.5). (3.15)

The results are in the table:

Parameter	**Actual**	**Max. Lik**	**Bias Reduced**
α	1.5	1.2668	1.2978
λ	0.5	0.40196	0.4231
β	2.0	2.4878	2.65097

Censored Samples

Let us treat the myeloma data set (App E) 65, observations 17 of which are right censored; x <- myeloma[myeloma$del= =1,1] and t <- myeloma[myeloma$del= =0,1].

First, we need initial estimates for input into the iteration method. The function init.gam(x) takes the uncensored portion of the data and computes the method of moments estimates. E.g.,

th0 <- init.gam(x) and the return is 0.98106187 0.04014575. (3.16)

Next, we use the general function Itest(x, t, th0, Newt, Llik), which will execute the iteration method described in Section 2c. The inputs are

x	the uncensored data
t	the censored data
th0	the initial estimates

Newt <- Newt.gam
Llik <- Llik.gam

The entry Itest(x, t, th0, Newt.gam, Llik.gam) (3.17)

returned

$\hat{\alpha}$ = 1.06707553; $\hat{\beta}$ = 0.03315037; number of cycles to convergence = 12.

The codes for this and support functions can be found in Appendices C and D.

3f. Lognormal Distribution

Complete Samples

Let us fit the lognormal distribution to the ball-bearing data. For complete samples this is a very simple task. Set x <- BallB and then

$\hat{\mu}$ <- mean(log(x)) = 4.150383; $\hat{\sigma}$ <- sqrt((n-1)/n)*stdev(log(x)) = 0.5216865. (3.18)

Let us also generate the 90% approximate confidence ellipse for (μ, σ), see Equation (2.16) and compare it with other confidence regions for normal data, specifically a region based on G^2 of Equation (2.17), and an exact trapezoidal-shaped region. These are explained in Section 6b. See Equations (6.16) and (6.17).

For the former we require the estimated information matrix

$$I(\hat{\mu},\hat{\sigma}) = \begin{Bmatrix} 3.674352 & 0 \\ 0 & 7.348704 \end{Bmatrix}, \tag{3.19}$$

and then m <- c(4.150383, 0.5216865); d <- qchisq(0.9, 2); plot.dat <- ellipse(n*I, m, d). The plotting sequence for Figure 3.7 is

```
plot(plot.dat[,1], plot.dat[,2], type="l", xlab="mu", ylab="sigma", ylim=c(.3,.8),
      xlim=c(3.8, 4.5))
lines(plot.dat[,1], plot.dat[,3])
contour(x, y, G2, nlevels = 1,v = d, labex = 0, lty=3, add=T)
                  (see Section 6.b for the computation of G2)
trap <- NormCT(log(BallB), 0.1, graph=F)
lines(trap[1,], trap[2,])
points(xb, s)
title(main = "90% Confidence Regions for Mu and Sigma")
```

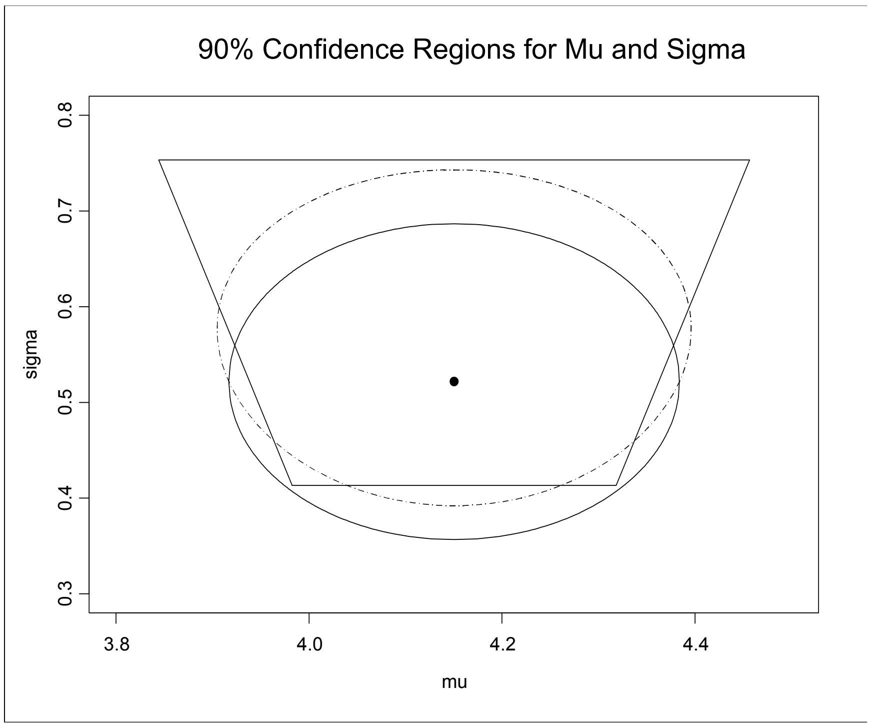

Figure 3.7

The solid ellipse is based on (2.16). The dashed curve is based on the G^2 statistic of (2.17). The trapezoid region is an exact region based on the independence of the sample mean and sample variance.

Bias Correction

The estimator for μ is unbiased and the estimator for σ need only be divided by the correction factor in (6.18). For the ball-bearing data this results in

$$\mu^* = 4.150383 \quad \text{and} \quad \sigma^* = 1.034156 \times 0.5216865 = 0.5395052 \qquad (3.20)$$

Censored Samples

Use the myeloma data from Appendix E. The initializing values are taken from the mean and standard deviation of the log of the uncensored data. Call it th0. I.e.,

```
th0 <-_c(mean(log(x)),sqrt((22/23)*var(log(x)))),                    (3.21)
```

which returns 4.1503827   0.5216865.

Then execute

```
Paramest <- Itest(x, t, th0, Newt.logn, Llik.logn)                  (3.22)
```

and the return is

$$\hat{\mu} = 4.2533626;\ \hat{\sigma} = 0.5216074;\ \text{number of cycles} = 18.$$

Model Selection

Having maximum likelihood estimates for the parameters of our competitive models for describing the ball-bearing data, it is interesting to consider how one might choose. In [Meeker and Escobar] the choice is an extended gamma distribution, a distribution that is not directly treated here. It is easy to compute the Kolmogorov-Smirnov test statistics and use them as distance functions. Meaningful p-values cannot be computed because the distributions are fitted from data. The distances are

ks.distance(BallB, Weibull) = 0.151
ks.distance(BallB, Gamma) = 0.123
ks.distance(BallB, Lognorm) = 0.090

The ball bearing set is a complete sample and we can make a graph that contains the empirical and the fitted model distributions. For an empirical distribution we use

$$\hat{F}(x_{(j)}) = j/(n+1) \text{ for j } = 1, \cdots, \text{n} \tag{3.23}$$

and plot these discrete values against the order statistics $x_{(j)}$.

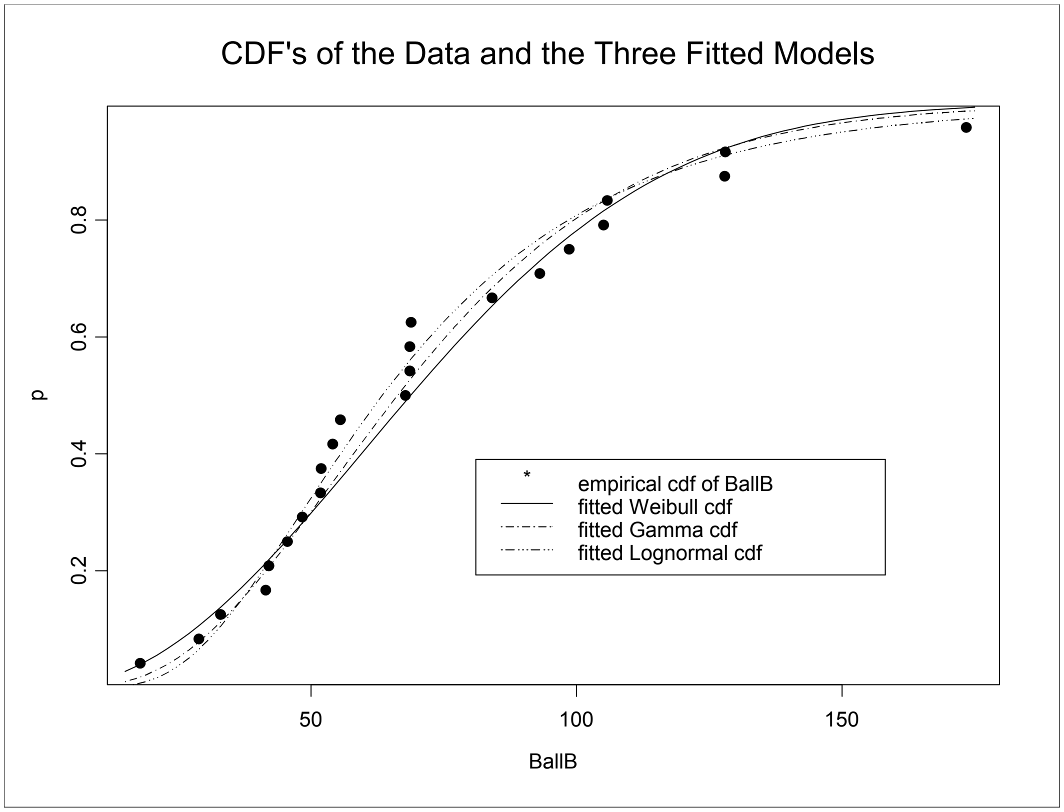

Figure 3.8

4. The Weibull Distribution

The Weibull distribution is a very popular model for reliability work, largely because of its ease of use and of its monotone hazard function, which is increasing if the shape parameter α is larger than one and decreasing when that parameter is smaller than one. The parameter β is the scale parameter. The life length random variable X has a Weibull distribution if $Y = (X/\beta)^{\alpha}$ has an Exp(1) distribution. This relationship is exploited broadly in what follows.

The five sections in this chapter describe formulae for: a) the model characterization functions; b) likelihood analysis for complete samples; c) bias reduction of maximum likelihood estimates; and d) likelihood analysis for censored samples. This last section includes details of the use of the extreme value distribution and its role in the estimation problems.

4a. Model Characterization Functions

The density and survivor functions are

$$f(x) = \frac{\alpha}{\beta}(\frac{x}{\beta})^{\alpha-1}\exp[-(\frac{x}{\beta})^{\alpha}] \quad \text{and} \quad S(t) = \exp[-(\frac{t}{\beta})^{\alpha}] \text{ for x,>0, t>0,} \tag{4.1}$$

where $\alpha > 0$ is a shape parameter and $\beta > 0$ is a scale parameter. The mean and variance are

$$\mu = \beta\,\Gamma(1 + 1/\alpha) \qquad \sigma^2 = \beta^2\,[\Gamma(1 + 2/\alpha) - \Gamma^2(1 + 1/\alpha)]. \tag{4.2}$$

The hazard function

$$h(t) = \frac{\alpha}{\beta}(\frac{t}{\beta})^{\alpha-1} . \tag{4.3}$$

The cumulative hazard function

$$H(t) = (\frac{t}{\beta})^{\alpha} . \tag{4.4}$$

The integrated survivor function

$$SS(t) = \frac{\beta}{\alpha}\int_{(\frac{t}{\beta})^{\alpha}}^{\infty} y^{1/\alpha-1}e^{-y}dy = \frac{\beta}{\alpha}\{\Gamma(1/\alpha) - \text{IncGam}[(\frac{t}{\beta})^{\alpha}]. \tag{4.5}$$

The mean residual life

$$LL(t) = SS(t)/S(t). \tag{4.6}$$

4b. Likelihood Analysis for Complete Samples

Let X_1, X_2, ⋯ , X_n be a complete random sample from a parent population, X ~ Weibull(α, β). Let ℓ be the log likelihood of a single observation, x. The direct analysis of this likelihood system follows. Liberal use is made of the fact that $(\frac{X}{\beta})^{\alpha} = Y \sim$ Exp(1).

$$\ell = \ln(\alpha) - \alpha\ln(\beta) + (\alpha-1)\ln(x) - (\frac{x}{\beta})^{\alpha} \tag{4.7}$$

$$\ell_{\alpha} = \frac{1}{\alpha} - \ln(\beta) + \ln(x) - (\frac{x}{\beta})^{\alpha}\ln(\frac{x}{\beta}) = \frac{1}{\alpha}[1+\ln(Y) - Y\ln(Y)]$$

$$\tag{4.8}$$

$$\ell_{\beta} = \frac{\alpha}{\beta}[-1 + (\frac{x}{\beta})^{\alpha}] = \frac{\alpha}{\beta}[-1+Y]$$

$$\ell_{\alpha\alpha} = -\frac{1}{\alpha^2}[1 + (\frac{x}{\beta})^{\alpha}\ln^2(\frac{x}{\beta})^{\alpha}] = -\frac{1}{\alpha^2}[1+Y\ln^2(Y)]$$

$$\ell_{\beta\beta} = \frac{\alpha}{\beta^2}[1-(\alpha+1)(\frac{x}{\beta})^{\alpha}] = \frac{\alpha}{\beta^2}[1-(\alpha+1)Y] \tag{4.9}$$

$$\ell_{\alpha\beta} = \frac{1}{\beta}[-1+(\frac{x}{\beta})^{\alpha} + (\frac{x}{\beta})^{\alpha}\ln(\frac{x}{\beta})^{\alpha}] = \frac{1}{\beta}[-1+Y+Y\ln(Y)]$$

Since $0 = E\{\ell_{\alpha}\} = E\{\ell_{\beta}\}$, we may deduce some interesting relationships.

$$E\{(\frac{X}{\beta})^{\alpha}\} = E\{Y\} = 1 \text{ and } E\{(\frac{X}{\beta})^{\alpha}\ln(\frac{X}{\beta})\} - E\{\ln(X)\} = \frac{1}{\alpha} - \ln(\beta) \tag{4.10}$$

Next, when we replace x with x_i and sum over the data, we may write the members of the Hessian as

$$L_{\alpha\alpha} = -\frac{1}{\alpha^2}\{n + \sum_{i=1}^{n}(\frac{x_i}{\beta})^{\alpha}\ln^2[(\frac{x_i}{\beta})^{\alpha}]\}$$

$$L_{\beta\beta} = \frac{\alpha}{\beta^2}\{n - (\alpha+1)\sum_{i=1}^{n}(\frac{x_i}{\beta})^{\alpha}\}$$

$$L_{\alpha\beta} = \frac{1}{\beta}\{-n + \sum_{i=1}^{n}(\frac{x_i}{\beta})^{\alpha} + \sum_{i=1}^{n}(\frac{x_i}{\beta})^{\alpha}\ln[(\frac{x_i}{\beta})^{\alpha}]\}$$

The maximum likelihood estimates are computed using an iteration function that follows readily from the pair of equations $L_\alpha = 0$ and $L_\beta = 0$. These two equations are easily extracted from (4.9). The latter yields the equation $\frac{1}{n}\sum [(\frac{x}{b})^\alpha] = 1$, which, when substituted into the former, produces

$$\frac{1}{\alpha} = \frac{1}{n}\sum \ln(x_i)(\frac{x_i}{\beta})^\alpha - \frac{1}{n}\sum \ln(x_i). \tag{4.11}$$

This in turn allows a determination of α from the left-hand side after an initial value of α is placed into the right-hand side. The iteration proceeds when the new value so determined is inserted into the right-hand side and the process repeated. This is often called the natural iteration function; it need not converge in general, but in this case it does. Its use is illustrated in Chapter 3.

The information matrix is

$$I_0 = \begin{Bmatrix} [1+\Psi'(2)+\Psi^2(2)]/\alpha^2 & \Psi(2)/\beta \\ \Psi(2)/\beta & [\alpha/\beta]^2 \end{Bmatrix}. \tag{4.12}$$

Proof. It follows from, see Appendix A,

$$E(\frac{X}{\beta})^\alpha \ln^2[(\frac{X}{\beta})^\alpha] = E\{Y \ln^2(Y)\} = \Gamma''(2)$$

$$E(\frac{X}{\beta})^\alpha \ln[(\frac{X}{\beta})^\alpha] = E\{Y \ln(Y)\} = \Gamma'(2)$$

$$\Gamma'(2) = \Psi(2) \text{ and } \Gamma''(2) = \Psi'(2) + [\Psi(2)]^2. \quad \square$$

Use of this is made in (3.3).

4c. Bias Reduction

The technique being used treats the two parameters separately.

(i) shape parameter, $\theta = \alpha$, so $U = \ell_\alpha$ when referring to (2.18).

$$U = \frac{1}{\alpha}\{1+\ln(Y) - Y\ln(Y)\}$$

$$U' = \frac{1}{\alpha^2}[1+Y\ln^2(Y)]$$

First,

$$E\{U^3\} = \frac{1}{\alpha^3}\ \{[1 + 3\Gamma'(1) - 3\ \Gamma'(2)]\ + 3[\Gamma''(1) - 2\ \Gamma''(2) + \Gamma''(3)]$$

$$+ [\ \Gamma'''(1) - 3\Gamma'''(2) + 3\Gamma'''(3) - \Gamma'''(4)]\}$$ and the expectation is

$$K_{3\,0} = \frac{1}{\alpha^3}\{-7.921007\} \text{ (see Appendix A).} \tag{4.13}$$

Second,

$$E\{U\,U'\} = \frac{1}{\alpha^3}\{1+\Gamma'(1)-\Gamma'(2)+\Gamma''(2)+\Gamma'''(2)-\Gamma'''(3)\}$$

and the expectation is

$$K_{1\,1} = \frac{1}{\alpha^3}\{-2.136823\} \text{ (see Appendix A).} \tag{4.14}$$

Third, from (4.11) the information in an observation is

$$i(\alpha) = I_0[1, 1] = [1+\Psi'(2)+\Psi^2(2)]/\alpha^2 = \frac{1}{\alpha^2}\{1.823681\}. \tag{4.15}$$

It follows from (2.20) that the bias function is

$$b(\alpha) = \frac{\alpha}{2n}(-4.309165). \tag{4.16}$$

(ii) scale parameter, $\theta = \beta$ and $U = \ell_\beta$ when referring to (2.15).

$$U = \frac{\alpha}{\beta}[Y-1\}$$

$$U' = \frac{\alpha}{\beta^2}[1-(\alpha+1)Y]$$

$$K_{30} = E\{U^3\} = (\frac{\alpha}{\beta})^3 E\{Y^3-3Y^2+3Y-1\} = 2(\frac{\alpha}{\beta})^3 \tag{4.17}$$

$$K_{11} = E\{U\,U'\} = \frac{\alpha^2}{\beta^3}E\{[Y-1][1-(\alpha+1)Y]\} = \frac{-\alpha^2(\alpha+1)}{\beta^3} \tag{4.18}$$

$$i(\beta) = I_0[,2,2] = (\alpha/\beta)^2 \tag{4.19}$$

It follows from (2.20) that the bias function is

$$b(\beta) = \frac{-\beta}{2n\alpha}(1+3/\alpha). \tag{4.20}$$

4d. Treatment of Censored Data; Use of the Extreme Value Distribution

The programs offered for complete samples use the Newton-Raphson iteration scheme in two dimensions. However, if one transforms the data by the logarithm the resulting density function is that of the extreme value distribution. The advantage of so doing allows the elimination of one of the two parameters in the system of likelihood equations. This technique is employed in treating the censored case.

The development of the relationship between the Weibull and Extreme Value distributions is essentially that appearing in [Lawless]. Begin with the survivor function

$$S(t; \alpha, \beta) = \Pr\{X > t\} = \exp\{-(t/\beta)^\alpha\}. \tag{4.21}$$

It follows that the pdf is

$$f_X(x) = \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1}\exp\{-(\frac{x}{\beta})^\alpha\}. \tag{4.22}$$

Both parameters are positive; α is the shape parameter and β is the scale parameter.

The pdf of the extreme value distribution has pdf

$$f_V(v) = \frac{1}{b}e^{(\frac{v-u}{b})}\exp\{-e^{(\frac{v-u}{b})}\} \quad for \ -\infty < v < \infty. \tag{4.23}$$

This distribution is related to the Weibull by the transformation

$$V = \log(X) \qquad X = \exp(V) \tag{4.24}$$

and the parametric identification

$$\begin{aligned} u &= \log(\beta) & \beta &= \exp(u) \\ b &= 1/\alpha & \alpha &= 1/b, \end{aligned} \tag{4.25}$$

from which it follows that $\frac{v-u}{b} = \alpha \log(\frac{x}{\beta}) = \log[(x/\beta)^\alpha]$.

Upon finding the mle's for the extreme value distribution and using the invariance property of maximum likelihood estimators one can obtain maximum likelihood estimators for the Weibull distribution. The development will not depend on whether the censoring mechanism is of Type 1, Type 2, or any other right censoring plan.

Let $X_1, X_2, \cdots, X_n$ be the life lengths of n items placed on test. Under Type 2 censoring, testing stops when r items have failed. Of course, if $r = n$, then there is no censoring. Under Type 1 censoring each expiring has a test termination time t. The structure of the likelihood system allows both cases to be treated with a single set of equations. The set D contains those life lengths that were completed prior to the termination time. The set C contains those life lengths that exceeded the allotted time The structure of the likelihood equations is given in Equations (2.27) and (2.28). The indicator variables δ_i tells us that $v_i = \ln(x_i)$ when equal to one, and $v_i = \ln(t_i)$ when equal to zero. The development should be compared with [Lawless, eq. (4.1.1)]. The log likelihood function has the structure

$$L(u, b) = \sum_D \ln(f(v_i)) + \sum_C \ln(S_V(v_i)), \tag{4.26}$$

where f has the form (4.22) and

$$S_V(v_i) = \Pr\{V_i > v_i\} = \Pr\{X > t_i\} = \exp[-\exp(\frac{v_i - u}{b})] \tag{4.27}$$

$$L = \ln lik(u,b) = -r\ln(b) + \sum_D \frac{v_i - u}{b} - \sum{}^* \exp[\frac{v_i - u}{b}],$$

$$\text{where } \sum{}^* \exp[\frac{v_i - u}{b}] = \sum_D \exp[\frac{v_i - u}{b}] + \sum_C \exp[\frac{v_i - u}{b}]$$

$$L_u = -\frac{r}{b} + \frac{1}{b}\sum_1^{r}{}^* \exp\{\frac{v_i - u}{b}\}$$

$$L_b = -\frac{r}{b} - \frac{1}{b}\sum_D \frac{v_i - u}{b} + \frac{1}{b}\sum_1^{r}{}^* \frac{v_i - u}{b}\exp(\frac{v_i - u}{b}).$$

Setting the first of these two equal to zero and solving leads to the separation

$$e^u = [\frac{1}{r}\sum{}^* \exp(\frac{v_i}{b})]^b \tag{4.28}$$

and substituting into the second equation leads to the nonlinear equation in b,

$$h(b) \;=\; \sum{}^{*} v_i \exp(\frac{v_i}{b}) / \sum{}^{*} \exp(\frac{v_i}{b}) - b - \frac{1}{r}\sum\nolimits_D v_i = 0\,. \tag{4.29}$$

This can be solved using Newton-Raphson.[2] To this end, we record the derivative

$$h'(b) \;=\; \frac{-\sum{}^{*} v_i^2 e^{v_i/b}}{b^2 \sum{}^{*} e^{v_i/b}} + \frac{[\sum{}^{*} v_i\, e^{v_i/b}]^2}{b^2[\sum{}^{*} e^{v_i/b}]^2} - 1\,. \tag{4.30}$$

Having b one obtains u from (4.28); one converts to (α, β) using (4.26).

Equation (4.27) can provide a way to get initial estimates for the parameters; it works for the complete case as well. Specifically, use the Kaplan-Meier estimate for the survivor function of V, the extreme value variate. Note that

$$y \;=\; \ln(-\ln(\hat{S}(v)) \;=\; (v - u)/b. \tag{4.31}$$

The least square estimates of u and b can serve for initialization.

[2] We have experienced lengthy oscillation using this method. Success has been achieved by using Newton-Raphson until two consecutive values of h are of opposite signs. At this point, we switch to a bisection search. The output contains a flag, which, if equal to zero, tells us that the bisection search was invoked. See the function Newt.wei().

5. Gamma Distribution

The gamma distribution mimics the Weibull distribution in the central portion, but there are major differences in the tails. The gamma distribution, for shape parameter α more than one, has an S-shaped hazard function which, although monotone increasing, approaches a finite asymptote. In this development, the rate parameter is λ.

5a. Model Characterization Functions

The density and survivor functions are

$$f(x) = \frac{\lambda^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} \exp[-\lambda x] \text{ and } S(t) = 1-\text{IncGam}(\lambda\ t) \text{ for } x,>0,\ t>0, \tag{5.1}$$

where $\alpha > 0$ is a shape parameter and $\lambda > 0$ is a rate parameter. The mean and variance are

$$\mu = \alpha/\lambda \qquad \sigma^2 = \alpha/\lambda^2. \tag{5.2}$$

The hazard function is

$$h(t) = f(t) / S(t). \tag{5.3}$$

The cumulative hazard function is

$$H(t) = -\log[S(t)]. \tag{5.4}$$

The integrated survivor function is

$$SS(t) = \int_t^{\infty} S(u)\,du = u\,S(u)\big|_t^{\infty} - \int_t^{\infty} u\,dS(u)$$

$$= -tS(t) + \int_t^{\infty} uf(u)\quad du = -t\,S(t) + \alpha \int_t^{\infty} \frac{u^{\alpha}e^{-u}}{\Gamma(\alpha+1)}\ du$$

$$= -t\,S(t,\alpha) + \alpha\,S(t,\alpha+1). \tag{5.5}$$

The mean residual life is

$$LL(t) = SS(t)/S(t) \tag{5.6}$$

5b. Likelihood Analysis for Complete Samples

Let $X_1, X_2, \cdots, X_n$ be a complete random sample from a parent population, $X \sim \text{Gamma}(\alpha, \lambda)$. The log likelihood, scores and their partial derivatives have the forms

$$\ell = \alpha \ln(\lambda) - \ln(\Gamma(\alpha) + (\alpha-1)\ln(x) - \lambda x \tag{5.7}$$

$$\ell_\alpha = \ln(\lambda) - \Psi(\alpha) + \ln(x) \tag{5.8}$$

$$\ell_\lambda = \alpha/\lambda - x \tag{5.9}$$

$$\ell_{\alpha\alpha} = -\Psi'(\alpha); \quad \ell_{\lambda\lambda} = -\alpha/\lambda^2; \quad \ell_{\alpha\lambda} = 1/\lambda \tag{5.10}$$

But if the alternative parameterization is used, $\beta = 1/\lambda$, one must adjust (5.9) and (5.10) and use

$$\ell_\beta = -\alpha / \beta + x / \beta^2; \ \ell_{\alpha\beta} = -1 / \beta^2; \ \ell_{\beta\beta} = \alpha / \beta^2 - 2\,x / \beta^3. \tag{5.11}$$

Since $0 = E\{ \ell_\alpha \}$ we may deduce an interesting relationship.

$$E\{X - \ln(X)\} = \ln(\lambda) - \Psi(\alpha) \tag{5.12}$$

It is useful to record the statistics

$$\overline{x} = \Sigma x_i/n; \quad \overline{\ln(x)} = \Sigma \ln(x_i)/n; \quad s^2 = \Sigma(x_1 - \overline{x})^2/(n-1), \tag{5.13}$$

and then the log likelihood function is easily expressed as

$$L = n \cdot \alpha \cdot \ln(\lambda) - n \cdot \ln[\Gamma(\alpha)] + (\alpha - 1)\, \Sigma \ln(x_i) - \lambda\, \Sigma x_{i,} \tag{5.14}$$

and the two components of the gradient vector are

$$L_\alpha = n\,[\, \ln(\lambda) - \Gamma'(\alpha) / \Gamma(\alpha)] + \Sigma \ln(x_i) \tag{5.15}$$

$$L_\lambda = n\,[\, \alpha / \lambda \,] - \Sigma x_i \tag{5.16}$$

Setting $L_\lambda = 0$ allows the substitution of $\alpha = \lambda \overline{x}$ into $L_\alpha = 0$. The resulting equation can be solved by Newton-Raphson iteration.

The Newton-Raphson algorithm requires initialization estimates. It is convenient to use the method of moments estimators for this purpose. Thus, we solve the equations

$$\overline{x} = \alpha / \lambda \quad \text{and} \quad s^2 = \alpha / \lambda^2, \tag{5.17}$$

and obtain $\tilde{\alpha} = \overline{x}^2/s^2$ and $\tilde{\lambda} = \overline{x} / s^2$.

Because of the elimination of λ, one requires only $\tilde{\alpha}$ for initialization into

$$g(\alpha) = \psi(\alpha) - \ln(\alpha) - \overline{\ln(x)} + \ln(\overline{x}) = 0 \tag{5.18}$$
$$g'(\alpha) = \psi'(\alpha) - 1 / \alpha.$$

This is managed by the gamma estimation function suite (see `gam.mle()`. Its use is illustrated in 3e). The S-Plus code is in Appendix E. Asymptotic expansions for the psi function and its derivatives are recorded in Appendix A.

The information matrix, developed in Appendix C, one for each parameterization

$$I(\alpha,\lambda)=\begin{Bmatrix}\psi'(\alpha) & -1/\lambda \\ -1/\lambda & \alpha/\lambda^2\end{Bmatrix} \qquad I(\alpha,\beta)=\begin{Bmatrix}\Psi'(\alpha) & 1/\beta \\ 1/\beta & \alpha/\beta^2\end{Bmatrix} \tag{5.19}$$

These developments are applied in 3e to real data. The computations include confidence regions for the two parameterizations.

5c. Quantities Needed to Execute the Bias Reduction Method

The general formulae (2.17), (2.20), and (2.23) are developed for when dealing with complete samples from the gamma population. The method adopted treats the parameters individually. Accordingly, the shape parameter α is treated first, and then the rate parameter, λ.

Case i. Shape parameter; $\theta = \alpha$ and hence $U = \ell_\alpha$

$$U = \ln(\lambda) - \Psi(\alpha) + \ln(X) \tag{5.20}$$
$$U' = -\Psi'(\alpha) \tag{5.21}$$
$$i(\alpha) = \Psi'(\alpha) \tag{5.22}$$
$$\begin{aligned} E\{U^3\} &= E\{\ln(\lambda X) - \Psi(\alpha)\}^3 = E\{\ln(Y) - \Psi(\alpha)\}^3, \text{ where } Y \sim \text{gamma}(\alpha, 1) \\ &= \psi''(\alpha) \end{aligned} \tag{5.23}$$

$E\{U'\,U\} = 0$ since U' is constant.

It follows that

$$b(\hat{\alpha}) = \frac{\Psi''(\alpha)}{2n[\Psi'(\alpha)]^2} \tag{5.24}$$

and the bias reduced estimate is

$$\alpha^* = \hat{\alpha} - C/2n \quad \text{with} \quad C = \frac{\Psi''(\hat{\alpha})}{[\Psi'(\hat{\alpha})]^2} \tag{5.25}$$

Accordingly, the quantity C must be computed case by case. The S-Plus function bias.gam() in Appendix C can be used to compute it.

Case ii. Rate parameter; $\theta = \lambda$ and hence $U = \ell_\lambda$

$$U = \frac{\alpha}{\lambda} - X, \; U' = -\frac{\alpha}{\lambda^2}, \; i(\lambda) = \alpha/\lambda^2. \tag{5.26}$$
$$E\{U^3\} = -2\alpha/\lambda^3 \tag{5.27}$$
$$E\{UU'\} = 0. \tag{5.28}$$

It follows that

$$b(\lambda) = -\frac{\lambda}{n\alpha} \tag{5.29}$$

and the bias reduced estimate is

$$\lambda^* = \hat{\lambda}(1 + \frac{1}{n\hat{\alpha}}) \tag{5.30}$$

Case iii. (Alternate parameterization), $\theta = \lambda$ and hence $U = \ell_\beta$

$$U = X/\beta^2 - \alpha/\beta \tag{5.31}$$

$$U' = -(2X - \alpha\beta)/\beta^3 \tag{5.32}$$

$$i(\beta) = (\alpha/\beta)^2. \tag{5.33}$$

Using $Y = X/\beta$ leads to $E\{Y-\alpha\}^3 = 2\alpha$ and $E\{(Y-\alpha)(2Y-\alpha)\} = 2\alpha$
$K_{30} = 2\alpha/\beta^3$; $K_{1,1} = -2\alpha/\beta^3$ and

$$b(\beta) = -2\beta / n\alpha^3 \tag{5.34}$$

and the bias reduced estimate is

$$\beta^* = \hat{\beta}(1 + 2/n\hat{\alpha}^3). \tag{5.35}$$

5d. Treatment of Censored Samples

Equations (2.25) thru (2.27) take the following forms when sampling from a censored gamma population.

$$L = r[\alpha \log(\lambda) - \psi(\alpha)] + (\alpha - 1) \sum_D \log(x_i) - \lambda \sum_D x_i + \sum_C \log[S(t_i)] \tag{5.36}$$

$$\begin{aligned} L_\alpha &= r[\log(\lambda) - \psi(\alpha)] + \sum_D \log(x_i) + \sum_C S_\alpha(t_i)/S(t_i) \\ L_\lambda &= r\,\alpha/\lambda - \sum_D x_i + \sum_C S_\lambda(t_i)/S(t_i) \end{aligned} \tag{5.37}$$

$$L_{\alpha\alpha} = -r\,\psi'(\alpha) + \sum\nolimits_C \{S_{\alpha\alpha}(t_i)/S(t_i) - [S_\alpha(t_i)/S(t_i)]^2\}$$

$$L_{\lambda\lambda} = -r\,\alpha/\lambda^2 + \sum\nolimits_C \{S_{\lambda\lambda}(t_i)/S(t_i) - [S_\lambda(t_i)/S(t_i)]^2\} \tag{5.38}$$

$$L_{\alpha\lambda} = r/\lambda + \sum\nolimits_C \{S_{\alpha\lambda}(t_i)/S(t_i) - S_\alpha(t_i)S_\lambda(t_i)/[S(t_i)]^2\}$$

The unresolved computational problems faced when dealing with Equations (5.37) and (5.38) appear in those summations over the set C. They contain derivatives of the Incomplete Gamma function and pose a major development. Our first step is to re-express these quantities in terms of he survivor function of the standard gamma random variable ($\lambda = 1$). Let us use the notation

$$S^*(t) = \frac{1}{\Gamma(\alpha)}\int_t^\infty x^{\alpha-1} e^{-x}\, dx. \tag{5.39}$$

It is easily seen that the above partial derivatives are needed only for this standard form. I.e.,

$$S(t) = S^*(\lambda t)\ ; \quad S_\alpha(t) = S_\alpha^*(\lambda t)\ ; \quad S_\alpha(t) = S_\alpha^*(\lambda t) \tag{5.40}$$

and the algorithms used are described in Appendix C. But, for the mixed partial derivative and the ones with respect to λ, we develop the formulae contained in the summary below.

Density (5.41)

$f^*(x) = x^{a-1} e^{-x} / \Gamma(\alpha)$ is used for the standard form, i.e., $\lambda = 1$.

$f(x; \alpha, \lambda) = \dfrac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\lambda x} = \lambda f^*(\lambda x)$ (or f(x) for short)

$f_x^*(x) = f^*(x)\,[\dfrac{\alpha-1}{x} - 1]$

$f_\alpha(x) = f(x)\,[\ln(\lambda x) - \Psi(\alpha)]$

$f_\lambda(x) = (\alpha/\lambda - x)\, f(x)$

$f_{\alpha\lambda}(x) = [(\alpha/\lambda - x)\,(\ln(\lambda x) - \Psi(\alpha)) + 1/\lambda]\, f(x).$

Survivor (5.42)

$$S(t;\alpha,\lambda) = \int_t^\infty f(x)dx = S^*(\lambda t), \text{ where } S^*(t) = \int_t^\infty f^*(x)dx$$

$$S_\alpha^*(t) = \int_t^\infty f^*(x)[\ln(x) - \Psi(\alpha)]dx$$

$$S_\alpha(t) = S_\alpha^*(\lambda t)$$

$$S_\lambda(t) = -t\, f^*(\lambda t) = -\frac{t}{\lambda} f(t)$$

$$S_{\alpha\lambda}(t) = -t\, f^*(\lambda t)[\ln(\lambda t) - \Psi(\alpha)] = -\frac{t}{\lambda} f(t)[\ln(\lambda t) - \Psi(\alpha)].$$

Systems of second partial derivatives (5.43)

$$f_{\alpha\alpha}(x) = f(x)\{[\log(\lambda x) - \Psi(\alpha)]^2 - \Psi'(\alpha)\}$$

$$f_{\alpha\lambda}(x) = f(x)\{(\alpha/\lambda - x)[\log(\lambda x) - \Psi(\alpha)] + 1/\lambda\}$$

$$f_{\lambda\lambda}(x) = f(x)\{[\alpha/\lambda - x]^2 - \alpha/\lambda^2\}$$

$$S_{\alpha\alpha}(t) = S_{\alpha\alpha}^*(\lambda t)$$

$$S_{\alpha\alpha}^*(t) = \int_t^\infty f^*(x)[\ln(x) - \Psi(\alpha)]^2 dx - \Psi'(\alpha)S(t)$$

$$S_{\alpha\lambda}(t) = -t\, f^*(\lambda t)[\ln(\lambda t) - \Psi(\alpha)]$$

$$S_{\lambda\lambda}(t) = -t^2 f^*(\lambda t)[\frac{\alpha - 1}{\lambda t} - 1]$$

6. The Lognormal Distribution

The lifetime X is lognormal, LN(μ, σ^2) if Y = log(X) is N((μ, σ^2). The model can be derived from fairly plausible assumptions and often found suitable for representing lifetimes, especially when large values are not of interest. Some applications are cited in {Lawless, p. 24]. This distribution has some strange properties; e.g., its hazard function is an inverted bathtub.

There is great support for the normal distribution, and this support translates to the lognormal distribution. For purposes of leveraging this support it is convenient to draw attention to some properties.

Properties of the Normal Distribution

Use φ (x) for the N(0, 1) density and $\Phi(x)$ for its cumulative distribution function. That is

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}) \qquad (6.1)$$

for the standard normal pdf and the properties:

$$\varphi'(x) = -x\,\varphi(x); \quad \Phi(x) = \int_{-\infty}^{x} \varphi(v)dv. \qquad (6.2)$$

Further, the survivor function for the N(0, 1) case

$$S(t) = 1 - \Phi(t); \qquad (6.3)$$

and the integrated survivor function can be developed

$$SS(t; 0, 1) = \int_t^\infty [1 - \Phi(x)]dx = \int_t^\infty \int_x^\infty \varphi(v)dv\,dx = \int_x^\infty \varphi(v)[\int_t^v dx]dv$$
$$= \int_x^\infty (v-t)\,\varphi(v)dv = -\int_t^\infty \varphi'(v)dv - t\int_t^\infty \varphi(v)\,dv$$
$$= \varphi(t) - t\,[1 - \Phi(t)]. \qquad (6.4)$$

The general N(μ, σ^2) case can be expressed using

$$f(x) = \frac{1}{\sigma}\varphi(\frac{x-\mu}{\sigma}) \qquad \text{and} \qquad S(t) = \int_{\frac{t-\mu}{\sigma}}^{\infty} \varphi(v)\,dv \qquad (6.5)$$

and the integrated survivor function is

$$SS(t) = \sigma \{\varphi (\frac{t-\mu}{\sigma}) - (\frac{t-\mu}{\sigma})[1 - \Phi (\frac{t-\mu}{\sigma})]\}. \qquad (6.6)$$

So much for the normal distribution. Let us turn to the lognormal distribution.

6a. Model Characterization Functions

The density and survivor functions for LN(μ, σ^2) are

$$f(x) = \frac{1}{\sigma x}\varphi(\frac{\log(x)-\mu}{\sigma}) \text{ and } S(t) = 1 - \Phi(\frac{\log(t)-\mu}{\sigma}) \text{ for x, >0, t>0,} \qquad (6.7)$$

where μ and $\sigma > 0$ take their usual interpretation. The mean and variance are

$$\text{mean}(X) = \exp[\mu + \sigma^2/2] \qquad \text{var}(X) = \exp[2\mu+\sigma^2]\,[\exp(\sigma^2) - 1]. \qquad (6.8)$$

The hazard function, or age specific failure rate is

$$h(t) = f(t)/S(t). \qquad (6.9)$$

The cumulative hazard function is

$$H(t) = - \log[S(t)]. \qquad (6.10)$$

The integrated survivor function is

$$SS(t) = [1 - \Phi(\frac{\log(t)-\mu}{\sigma})]\,\{ e^{\mu+\sigma^2/2} - t\}. \qquad (6.11)$$

The mean residual life is

$$LL(t) = SS(t)/S(t). \qquad (6.12)$$

Only the integrated survivor function is a bit intricate to understand. Let's first treat a related integral.

$$\int_a^\infty [1 - \Phi(w)]\, e^{\sigma w}\, dw = \frac{1}{\sigma}\int_a^\infty [1 - \Phi(w)]\, d(e^{\sigma w})$$

$$= \frac{1}{\sigma}\{ - e^{\sigma a}\,[1 - \Phi(a)] + \int_a^\infty e^{\sigma w}\varphi(w)dw\}$$

and use the complete the square technique in the integral part

$$\frac{1}{\sigma}\{-e^{\sigma a}[1-\Phi(a)] + e^{\sigma^2/2}\int_a^\infty \varphi(w-\sigma)dw\}.$$

With this in mind, let us turn to

$$SS(t) = \int_t^\infty [1-\Phi(\frac{\log(v)-\mu}{\sigma})]dv = \sigma e^{\mu}\int_a^\infty [1-\Phi(w)]\, e^{\sigma w}\, dw,$$

where a = (log(t) − μ)/σ. Now we put in the related part and complete the function in a computable form.

$$SS(t) = e^{\mu}\{-e^{\sigma a}[1-\Phi(a)] + e^{\sigma^2/2}\int_{a-\sigma}^{\infty} \varphi(w)dw\}$$
$$= [1-\Phi(a-\sigma)]\, e^{\mu+\sigma^2/2} - t\,[1-\Phi(a)]. \qquad \square \qquad (6.13)$$

6b. Complete Samples

Let $X_1, X_2, \cdots, X_n$ be a complete random sample from a parent population, $X \sim LN(\mu, \sigma^2)$. The relationship connecting Y = log(X) does not involve the parameters. So, the well-known formulae for the maximum likelihood estimates of μ and σ using $y_1, y_2, \cdots, y_n$, the logarithms of the data, provide the maximum likelihood estimates. The invariance principle is invoked. Further, the information matrix will be the same.

$$I_0(\mu, \sigma) = \begin{Bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{2}{\sigma^2} \end{Bmatrix} \qquad (6.14)$$

Note: Had we been estimating σ^2 instead of σ, the lower right entry would have been $1/(2\sigma^2)$.

Since the normal and log normal distributions have the same set of parameters, then the joint confidence regions are the same. The approximate ellipses can be generated in the same way as before; but an exact trapezoidal-shaped confidence region is also available, and it is interesting to compare the two. The former can be found by using (6.14) in Equation (2.16). The latter is based on the independence of the sample mean and variance. Let

$$\bar{Y} = \frac{1}{n}\sum Y_i \quad and \quad SS = \sum (Y_i - \bar{Y})^2 \qquad (6.15)$$

when $Y_1, Y_2, \cdots, Y_n$ is a random sample from $N(\mu, \sigma^2)$.

The former is distributed according to N(μ, σ^2/n) and the latter by $\sigma^2\chi^2_{(n-1)}$. It follows that a joint confidence region for (μ, σ) can be obtained from setting the product

$$\Pr\{-z_0 \le \sqrt{n}\,(\frac{\bar{X}-\mu}{\sigma}) \le z_0\}\Pr\{k_1 \le SS/\sigma^2 \le k_2\} \;=\; 1-\alpha. \tag{6.16}$$

Let the first factor have probability $1-\alpha_1$ and the second $1-\alpha_2$. A default position might be to use $(1-\alpha)^{½}$ for each, but it is useful to be more flexible. We choose to use a parameter p, 0 < p < 1, such that

$$1-\alpha_1 \;=\; (1-\alpha)^{p} \quad\text{and}\quad 1-\alpha_2 \;=\; (1-\alpha)^{1-p}.$$

Also, for greater flexibility, let us use a parameter q, 0 < q < 1, in the second factor so that

$$\Pr\{\chi^2_{(n-1)} < k_1\} \;=\; q\alpha_1 \text{ and } \Pr\{\chi^2_{(n-1)} > k_2\} \;=\; (1-q)\alpha_1.$$

This provides relief from the practice of splitting the tails evenly in the asymmetric chi square distribution.

Each factor can be pivoted separately, i.e.,

$$\Pr\{\bar{X}-z_0\sigma/\sqrt{n} \le \mu \le \bar{X}+\sigma/\sqrt{n}\}\Pr\{\sqrt{SS/k_2} \le \sigma \le \sqrt{SS/k_1}\ \}. \tag{6.17}$$

The confidence region is the intersection of the two events. The range of values for σ in the second factor is used in the interval limits of the first factor. This capability is programmed into the function `NormCI(x, alph, p, q, graph = T)`, a plotting function that accepts the data x and α, where 1 − α is the confidence level. If graph = F, then the graph is not drawn and the return is the set of vertices of the trapezoid. The function contains defaults p = ½, q = ½.

6c. Bias Reduction

The estimators for μ is unbiased, but not so for the estimator for σ. In this latter case, we can use an exact correction based on the expectation the maximum likelihood estimator for σ, i.e.,

$$E\{s\} \;=\; \sigma\sqrt{\frac{2}{n}}\frac{\Gamma(n/2)}{\Gamma((n-1)/2)} \tag{6.18}$$

and the exact bias adjustment is obtained from (6.18).

Note 1: [Barndorff-Nielsen and Cox, 1994, p. 187] post the reciprocal of the third factor in (6.18).

Note 2: If one is using the unbiased estimator for σ^2, then the factor $\sqrt{2/n}$ should be replaced with $\sqrt{2/(n-1)}$.

6d. Censored Samples

We require partial derivatives of f and S, (6.7) for use in (2.28), (2.29), and (2.30). The are recorded here. Begin with those of the first order, using $z = \frac{\ln(x) - \mu}{\sigma}$ and $w = \frac{\ln(t) - \mu}{\sigma}$

$$f_\mu(x) = \frac{-1}{x\sigma^2}\,\varphi'\left(\frac{\ln(x)-\mu}{\sigma}\right) = \frac{1}{x\sigma^2}\,z\,\varphi(z) \;=\; z\,f(x)/\sigma$$

$$f_\sigma(x) = \frac{1}{x\sigma^2}\,\varphi(z)\{z^2-1\} \;=\; (z^2-1)\,f(x)/\sigma$$

$$S_\mu(t) \;=\; \frac{1}{\sigma}\varphi(w)$$

$$S_\sigma(t) \;=\; \frac{w}{\sigma}\varphi(w)$$

and those of the second order,

$$f_{\mu\mu}(x) = \frac{1}{x\sigma^3}\,\varphi(z)(z^2-1) \;=\; (z^2-1)\,f(x)/\sigma^2$$

$$f_{\sigma\sigma}(x) = \frac{2}{x\sigma^3}\,\varphi(z)\{1 - z - 3z^2 + z^3\} \;=\; 2\,f(x)[1 - z - 3z^2 + z^3]/\sigma^2$$

$$f_{\mu\sigma}(x) = \frac{1}{x\sigma^3}\,\varphi(z)\{z(z^2-3)\} \;=\; z(z^2-3)\,f(x)/\sigma^2$$

$$S_{\mu\mu} \;=\; \frac{w}{\sigma^2}\varphi(w)$$

$$S_{\sigma\sigma} \;=\; \frac{1}{\sigma^2}w(w^2-2)\,\varphi(w)$$

$$S_{\mu\sigma} \;=\; \frac{1}{\sigma^2}(w^2-1)\,\varphi(w)\,.$$

The program for estimating μ and σ is `ltest()`. Its use is illustrated in Section 3.

7. Summary and Conclusions

The paper offers great flexibility to those practitioners and students who wish to make exploratory computations and learn the behavior of models. Those possessing a modicum of programming capability can expand the choices given here to other models and situations as the structure offers a template that facilitates additional expansion. Much remains to be learned. What follows is a listing of what has been learned thus far. We comment on the items as presented in Section 3.

Survival distribution selection must be based on careful modeling and good judgment. This area of statistical work has the disadvantage that the collection of large quantities of data requires time, often unacceptable amounts of time. Yet the comparisons of models using somewhat small data sets does not allow one to examine the tail behavior. The sole use of histograms and q-q plots does not suffice. The graphing of the other support functions, especially the hazard function and mean residual life, can aid the practitioner in modeling and the making of choices. Such is illustrated in Sections 3a and 3b.

The graphing of the behavior of planned replacement policies can be quite useful to those involved in the generation of maintenance policies. The problem treated in 3c deals solely with the relative costs of planned versus random replacements. The most interesting point is the flat behavior of the cost curves once the original high costs fall away. This is useful to the maintenance planner because the replacement model is generically a very simple one, one that does not consider the costs of arranging to make the replacements at the optimal point in time. Such costs are likely to be high and it is comforting to know that the optimal point is contained in a very broad window. One need not give ground to the problems of performing maintenance at inconvenient times.

Sections 3d, 3c, and 3f illustrate the implementation of the general materials presented in Section 2 for the Weibull, Gamma, and Lognormal distributions, respectively.

There are four areas of specialization for each:

Computation of the support functions.
Maximum likelihood parameter estimation for complete samples.
Implementation of bias reduction techniques.
Maximum likelihood estimation for right censored samples.

Summary of each is made in turn.

Most software systems provide the capability to compute density, cumulative distribution, and quantile functions. The extension of such capabilities to the set of survival distribution support functions has yet to occur. The three cases presented here provide illustration as to what must be done.

Maximum likelihood estimation for complete samples is popular and well understood, yet they are hard to compute. The algorithms presented here have high precision. The statistics texts do not provide much in the way of support for joint confidence regions for multi-parameter models. The author believes them to be useful. They bear witness to Bellman's "curse of dimensionality." It is quite remarkable how large these regions are for samples of size 50 and 100.

Maximum likelihood estimators have long been known to be biased, but there appears little available commentary on the importance of this bias. The issue has not received conspicuous attention. The method implemented here was taken from [Cox and Hinkley]; it is a first order approximation based on the marginal distribution of the individual parameters. The limited computational results presented are not very interesting; but the stage is set for a broadly based Monte Carlo simulation study geared to learn the behavior of the method. At present, it appears that it may be useful only for rather small sample sizes.

Methods for managing censored data may be available in the Splida and Reliasoft systems. But these are opaque. The open methods that appear here are limited, but offer a beginning. The functions `Itest()` and `Golden()` are generic. In concept, they may be applied with any right-censored data set modeled with any distribution that is smooth in its parameters, and any finite dimension of the parameter space. However, they have not been broadly tested. In fact, their use was not competitive with the bisection search method devised for the Weibull distribution. See Section 4.

The idea of alternating the application of Newton-Raphson iteration with a golden section search may be new. It should be useful whenever the log likelihood is a concave function of the parameters. It appears to converge in a reasonably short amount of time, in spite of its low dependence upon the quality of the initializing point.

The computation of derivatives of the incomplete gamma function came up in the treatment of the gamma distribution for censored samples. A method of high precision was developed and appears in Appendix D. It is suitable for parallel processing. Mostly the values are precise to at least nine decimal places.

The computation of the information matrix and the bias reduction term is quite difficult for censored samples. Indeed the calculation depends upon the type of censoring. Much depends on the information matrix. Also, there is dependence on the ratio of the number of uncensored sample to the total sample size. The quality of the estimation results will deteriorate as the number of uncensored observations becomes small.

Locations of S-Plus Listings in the Appendices

Appendix A. psi(), psi.bas(), g1(), g2(), g3(), g4()

Appendix B. weibull.mle(), weibull.mm(), weibull.mle(), weibull.est(), Newt.wei(),

Init.wei(), KapM(), convtowei()

Appendix C. Itest(), Golden(), init.gam(), Llik.gam(), Newt.gam(), Lp.gam(), Llik.logn(), Newt.logn(), Lp.logn()

Appendix D. Surv.gam(), surv.gam(), surva.gam(), survaa.gam(), trans1(), trans2(), trans3()

Appendix E. gam.est(), ellipse(), exp1.lnorm(), wei1.gam(), wei2.gam(), wei.cost(), NormCT(), area.comp(), seg.comp(), sol.pt(), Newt.wei(), exp.lnorm(), Ext.newt(), cost.comp(), Gsq.norm(), Gsq.gam(), Gsq.wei()

Appendix A

Asymptotic Expansions for the Polygamma Functions

The Psi function is defined as the derivative of the log gamma function. The recursion formula for the Gamma function translates into a like formula for the Psi function. Thus,

$$\psi(z) = d(\ln(\Gamma(z))/dz \quad \text{and} \quad \psi(z+1) = \psi(z) + 1/z. \tag{A.1}$$

Asymptotic expansions are especially advantageous because high precision is available using few terms, provided z is large. This can be easily exploited in computations involving the Polygamma functions. Thus, if z + r is sufficiently large to obtain appropriate accuracy, then use

$$\psi(z) = \psi(z+r) - 1/z - 1/(z+1) - \ldots - 1/(z+r-1) \tag{A.2}$$

$$\psi'(z) = \psi'(z+r) + 1/z^2 + 1/(z+1)^2 + \ldots + 1/(z+r-1)^2 \tag{A.3}$$

and so on. It remains to record the expansions for large z, [Abramowitz and Stegun]

$$\psi(z) = \ln(z) - 1/2z - 1/12z^2 + 1/120z^4 - 1/252z^6 + \ldots \tag{A.4}$$

$$\psi'(z) = 1/z + 1/2z^2 + 1/6z^3 - 1/30z^5 + 1/42z^7 - 1/30z^9 + \ldots \tag{A.5}$$

$$\psi''(z) = -1[1/z^2 + 1/z^3 + 1/2z^4 - 1/6z^6 + 1/6z^8 - 3/10z^{10} +]$$

$$\psi'''(z) = 2/z^3 + 3/z^4 + 2/z^5 - 1/z^7 + 4/3z^9 - 3/z^{11} +$$

$$\psi^{(n)}(z) = (-1)^{n+1} [(n-1)!/z^n + n!/2z^{n+1} + \Sigma B_{2k} (2k+n-1)!/(2k)!z^{2k+n}] \tag{A.6}$$

and $\{B_{2k}\}$ are the Bernoulli numbers. See [Abramowicz and Stegun].

The error in these expansions is sized by the first term ignored. They work best for z large, say z > 10. For smaller values of z one should make adjustments based on the recursive formula for the gamma function.

$$\begin{aligned}
&\Gamma(z+a) = (z+a-1)\cdots(z+1)(z)\Gamma(z) \\
&\Psi(z+a) = \sum_{0}^{a-1} \frac{1}{z+j} + \Psi(z) \\
&\Psi'(z+a) = -\sum_{0}^{a-1} \left(\frac{1}{z+j}\right)^2 + \Psi'(z)
\end{aligned} \tag{A.7}$$

$$\psi''(z+a) = 2\sum_{0}^{a-1}(\frac{1}{z+j})^3$$

$$\psi'''(z+a) = -6\sum_{0}^{a-1}(\frac{1}{z+j})^4 .$$

For these reasons, two programs have been written: psi.bas (z) takes a vector argument z and returns three rows of output, the function and its first two derivatives. For small values of z, one can choose an integer a and call psi (z,a) to achieve precision based on (A.7). The default value of a is 10. Adjustments are made for all components of z less than a.

On occasion we require the derivatives of the gamma function. These are obtainable from the psi function and its derivatives. The first four are as follows.

$$\begin{aligned}
\Gamma'(z) &= \Gamma(z)\,\psi(z) \\
\Gamma''(z) &= \Gamma(z)\{\,\psi'(z) + \psi^2(z)\,\} \\
\Gamma'''(z) &= \Gamma(z)\{\psi''(z) + 3\psi'(z)\psi(z) + \psi^3(z)\} \qquad\qquad (A.8) \\
\Gamma^{iv}(z) &= \Gamma(z)\{\psi'''(z) + 4\psi''(z)\psi(z) + 6\psi'(z)\psi^2(z) + 3[\psi'(z)]^2 + \psi^4(z)\}.
\end{aligned}$$

Useful relationships

$$\int_0^\infty x^{r-1}\ln^s(x)e^{-x}dx = \Gamma^{(s)}(r) \text{ and } \Gamma^{(r+1)}(v) = \sum_{j=1}^{r}\binom{r}{j}\Gamma^{(j)}(v)\,\Psi^{(r-j)}(v)$$

$$\Psi(n)=\Psi(1)+\sum_{k=1}^{n-1}\frac{1}{k}; \quad \Psi'(n)=\Psi'(1)-\sum_{k=1}^{n-1}\frac{1}{k^2}; \quad \Psi''(n)=\Psi''(1)+2\sum_{k=1}^{n-1}\frac{1}{k^3} .$$

S-Plus Codes

```
psi
function(z, a = 10){
# fname is psi; programmer; R. Read
# derivative log gamma and two subsequent derivatives
# Uses recursive formula for min(z) < a and the
# psi.bas function for large z (asymptotic expansion)
        a <- floor(a)
        s <- a - floor(min(z))
        j <- 1:s
        if(s > 0) {
                temp0 <- apply(1/outer(z - 1, j, "+"), 1, sum)
                temp1 <- apply(1/outer(z - 1, j, "+")^2, 1, sum)
                temp2 <- apply(1/outer(z - 1, j, "+")^3, 1, sum)        }
        else {
                temp0 <- temp1 <- temp2 <- 0}
        if(s <= 0)
                s <- 0
        out <- psi.bas(z + s)
        out[1,  ] <- out[1,  ] - temp0
        out[2,  ] <- out[2,  ] + temp1
        out[3,  ] <- out[3,  ] - 2 * temp2
        out}

psi.bas
function(z){
# fname is psi.bas; programmer:  R. Read
# asymptotic expansion for first three derivatives
# of the log gamma function; z may be a vector.
        K <- length(z)
        coef0 <- c(-2, -12, 120, -252)
        m0 <- matrix(rep(coef0, rep(K, 4)), ncol = 4)
        z0 <- c(z, z^2, z^4, z^6)
        coef1 <- c(1, 2, 6, -30, 42, -30)
        m1 <- matrix(rep(coef1, rep(K, 6)), ncol = 6)
        z1 <- c(z, z^2, z^3, z^5, z^7, z^9)
        coef2 <- c(-1, -1, -2, 6, -6)
        m2 <- matrix(rep(coef2, rep(K, 5)), ncol = 5)
        z2 <- c(z^2, z^3, z^4, z^6, z^8)
        out <- log(z) + apply(1/(m0 * z0), 1, sum)
        out <- rbind(out, apply(1/(m1 * z1), 1, sum))
        out <- rbind(out, apply(1/(m2 * z2), 1, sum))
        out}

g1
function(z){
# fname is g1
# first derivative of the gamma function
        out <- gamma(z) * psi(z)[1,  ]
        return(out)}

g2
function(z){
# fname is g2
# second derivative of the gamma function
        tmp <- psi(z)
        out <- gamma(z) * (tmp[2,  ] + tmp[1,  ]^2)
        return(out)}

g3
function(z){
# fname is g3
# third derivative of the gamma function
        tmp <- psi(z)
        out <- gamma(z) * (tmp[3,  ] + 3 * tmp[2,  ] * tmp[1,  ] + tmp[1,  ]^3)
        return(out)}

g4
function(z){
# fname is g4
# fourth derivative of the gamma function
        tmp <- psi(z)
        out <- gamma(z) * (tmp[4,  ] + 4 * tmp[3,  ] * tmp[1,  ]
                + 6 * tmp[2,  ] * tmp[1,  ]^2 + 4 * tmp[2,  ]^2 + tmp[1,  ]^4)
        return(out)}
```

Appendix B

Analysis for Computational Support of the Weibull Distribution

This appendix contains the numerical analysis and S-Plus code for executing the support functions developed in Chapter 4. Fundamental support functions for the Weibull distribution are primitives in the S-Plus system. Capabilities for computing the psi and gamma functions and their derivatives are contained in Appendix A. The fact that $Y = (X/\beta)^{\alpha}$ has an Exp(1) distribution when $X \sim \text{Weibull}(\alpha, \beta)$ has been exploited and the properties of the Exponential distributions are familiar.

It remains to describe the computational techniques used to compute maximum likelihood estimates for the shape and scale parameters (α, β) under complete and censored sampling.

Complete Samples

The maximum likelihood estimates are computed using the natural iteration function described in Chapter 4. The method of moments estimates are used for initialization. The Newton-Raphson Iteration (B.1) is used to find these latter estimates. The functions utilized are

weibull.mle(); weibull.mm(); weibull.est();

the last of these functions calls the previous two and displays both kinds of estimates.

Their use is illustrated in Section 3 and the S-plus listings appear in the S-Plus section of this appendix.

Right-Censored Samples

The formulae in Section 4d require computational development. The transform the extreme value distribution enables one to execute a one-dimensional search for the parameter b, see (4.24) and (4.28). However, it appears that the Newton-Raphson scheme undergoes considerable oscillation when applied in this setting and some modifications are in order. The basic idea is to use Newton-Raphson until the function h of (4.28) oscillates and then switch to a bisection search.

A little more detail can be useful. The basic iteration is to use

$$b_{n+1} = b_n - h(b_n) / h'(b_n) \qquad \text{(B.1)}$$

and record the two most recent pairs, call them (b_1, h_1) and (b_2, h_2). At each step check the sign of the product $h_1 \times h_2$; when it turns negative, change to the bisection search. I.e.,

set b = $(b_1 + b_2)/2$; compute h(b);
if $h \times h_1 < 0$ set h_2 = h and b_2 = b; otherwise set h_1 = h and b_1 = b; (B.2)
repeat until no change in b, or h ≈ 0, or both.

The S-Plus function that does this is called Newt.wei().

The initialization exploits Equation (4.30). There we have a straight-line relationship between v and y, which can be expressed using a least squares fit. Then set b = 1/slope and u = x-intercept. The conversion of $\{v_i\}$ to $\{y_i\}$ is accomplished using the Kaplan-Meier estimator for the survivor function. It was decided to write a simple code to accomplish this.

The basic Kaplan-Meier estimation has the following rules. Pool and order the uncensored and censored times, call them $\{x_i^*\}$. Let d_i be the number that died at x_i^*, i.e., exclusive of those that were censored at that value. Set n_0 = n and $n_i = n_{i-1} - d_i$ for all of the unique values of $\{x_i^*\}$. These are the number at risk values. Then the estimated survivor function is, at the data points,

$$S(x_i^*) = \prod_{j<i} \frac{n_j - d_j}{n_j}. \qquad \text{(B.3)}$$

These details are executed by the functions init.wei() and KapM(); listings below.

S-Plus Codes

```
weibull.mle
function(x, a0 = 1, ep = 0.0001){
# fname is weibull.mle
# natural iteration function; a0 is initial shape parameter
# output is the pair (shape, scale)
        lxb <- mean(log(x))
        a <- a0
        repeat {
                ainv <- mean(log(x) * x^a)/mean(x^a) - lxb
                a <- 1/ainv
                if(abs(a - a0) < ep)
                        break
                a0 <- a
        }
        b <- (mean(x^a))^(1/a)
        out <- c(a, b)
        out}

weibull.mm
function(x, a0 = 1, ep = 0.0001){
```

```
# fname is weibull.mm
# returns the method of moments estimator for the shape (a) and scale (b)
# parameters of the weibull distribution. The initialization is a0. The data are x.
        xb <- mean(x)
        s2 <- var(x)
        a <- a0
        repeat {
                g <- lgamma(1 + 2/a) - 2 * lgamma(1 + 1/a) - log(1 + s2/xb^2)
                gp <- (psi(1 + 2/a)[1] - psi(1 + 1/a)[1]) * (-2/a^2)
                a <- a0 - g/gp
                if(a < 0)
                        a <- 0.1
                if(abs(a - a0) < ep)
                        break
                a0 <- a }
        b <- xb/gamma(1 + 1/a)
        out <- c(a, b)
        out}

weibull.mle
function(x, a0 = 1, ep = 0.0001)
{# fname is weibull.mle
# natural iteration function; a0 is initial shape parameter
# output is the pair (shape, scale)
        lxb <- mean(log(x))
        a <- a0
        repeat {
                ainv <- mean(log(x) * x^a)/mean(x^a) - lxb
                a <- 1/ainv
                if(abs(a - a0) < ep)
                        break
                a0 <- a }
        b <- (mean(x^a))^(1/a)
        out <- c(a, b)
        out}

weibull.est
function(x){
# fname is weibull.est
# Input is a random sample from the Weibull distribution.
# Output has five components: shape and scale parameter estimates
# using max likelihood, sample size, shape and scale parameter
# estimates using method of moments
        n <- length(x)
        mm <- weibull.mm(x)
        ml <- weibull.mle(x)
        out <- c(ml, n, mm)
        names(out) <- c("shape:mle", "scale:mle", "samp size", "shape:mm", "scale:mm")
        out}

Newt.wei
function(x, t, param, ep = 0.0001){
# fname is Newt.wei
# Newton-Raphson method, then bisection search used to find mle
# for the Weibull distribution using the extreme value distribution technique.
# Data input is (x, t); param initialization is (u, b), b>0
# output is (u, b, alph, beta). r is cardinality of uncensored set
        r <- length(x)
        v <- log(x)
        vb <- mean(v)
        lt <- log(t)
        b <- param[2]
        u <- param[1]
        vv <- c(v, lt)
        b2 <- b1 <- b
        h1 <- h2 <- 0
        j <- 1
        flag <- 1
        repeat {
                if(flag == 1) {
                        D <- sum(exp(vv/b))
                        N <- sum(vv^2 * exp(vv/b))
                        A <- sum(vv * exp(vv/b))
                        h <- A/D - b - vb
                        hp <- - N/(D * b^2) + A^2/(D * b)^2 - 1
                        b <- b1 - h/hp
                        if(b <= 0)
                                b <- 0.01
                        if(max(abs(h), abs(b - b1)) < ep)
                                break
                        if(j/2 != round(j/2)) {
                                b1 <- b
                                h1 <- h}
                        if(j/2 == round(j/2)) {
                                b2 <- b
                                h2 <- h}
                        j <- j + 1
                        if(sign(h2 * h1) < 0)
                                flag <- 0}
                if(j == 25)
                        break
                if(flag == 0) {
                        b <- (b1 + b2)/2
                        D <- sum(exp(vv/b))
                        A <- sum(vv * exp(vv/b))
                        h <- A/D - b - vb
                        if(sign(h * h1 < 0)) {
                                h2 <- h
                                b2 <- b}
                        if(sign(h * h2 < 0)) {
                                h1 <- h
                                b1 <- b}
                        if(max(abs(h), abs(b - b1)) < ep)
                                break
                        j <- j + 1
                        if(j == 50)
                                break} }
        u <- b * log(D/r)
        alph <- 1/b
        beta <- exp(u)
        out <- c(u, b, alph, beta, flag)
        names(out) <- c("u","b",”alph","beta","flag")
        return(out)}

init.wei
function(x, t){
# fname is init.wei
# provides initial estimates for the extreme value distribution
# i.e., y = log(weibull), parameters (u, b). The pair (x, t)
# represents n observations (duplicate values required) where
# the 't' values are the censoring values. Output is a four vector;
# first two are (u, b) and the last two are (alpha, beta).
# lsfit to the log survival fnc technique is utilized.
        S <- KapM(x, t)
        w <- log( - log(S))
        y <- sort(log(c(x, t)))
        yy <- unique(y)
        yb <- mean(yy)          # w <- lsfit(yy, S)$coef
        slope <- sum(w * (yy - yb))/sum((yy - yb)^2)
        interc <- mean(w) - slope * yb
        b <- 1/slope
        u <- - b * interc
        alph <- slope
        beta <- exp(u)
        names(out) <- c("u", "b", "alph", "beta")
        out <- c(u, b, alph, beta)
        return(out)}
KapM
```

```
function(x, t){
# fname is KapM
# produces the Kaplan-Meier estimate of the survivor function for
# data c(x, t) where x are the actual death times and t are the
# censored values.
        y <- sort(c(x, t))
        yy <- unique(y)
        dd <- d <- tab <- table(y)      # number of deaths at y
        n <- length(y)
        k <- length(yy)
        nr <- n - tab    # initial number at risk
        tt <- unique(t)
        ind <- (1:k)[tt == yy]
        tabt <- table(tt)
        kk <- length(ind)
        if(kk > 0) {
                for(j in 1:kk)
                        dd[ind[j]] <- d[ind[j]] - tabt[j]}
        SS <- (nr - dd)/nr
        S <- cumprod(SS)
        return(S)}
```

Appendix C

Implementation of the General Censored Data Estimation Scheme

The iterative estimation scheme described in Section 2 has been implemented for two out of three distributions, namely the gamma and lognormal distributions. The case of the Weibull distribution can be managed more directly as described in Section 4 and Appendix B.

Much of the structure can be treated generally. Because S-Plus functions have the capability to accept other functions as input, it is possible to write generic code for the process. Such is exploited for the values of the log likelihood, the golden section search, and the change of gradient direction effected by a single iteration of the Newton-Raphson scheme. The name Itest is short for iterative estimation.

The basic call is

```
Paramest <- Itest( x, t, th0, Newt, Llik, ep = .0001),
```

where x is the set of uncensored survival times; t is the set of right-censored times realized; th0 is the initialization point in the parameter space; Newt is a generic function that produces an updated value for the parameter; and Llik is a generic function that computes the log likelihood for the targeted distribution family. The value of ep is used to control the precision of the estimate. Of course, the inputs to Newt and Llik must be generated by the parent function.

The function also calls a golden section search function. Its role is to seek the best value for th along the line segment connecting th previous and the output of Newt. Having made that selection, the parent program calls Newt to find a new direction for search. The output has the structure

```
Paramest = th[1], th[2], N.iter,
```

where N.iter is the number of cycles through Newt.

S-Plus Codes

```
Itest
function(x, t, th0, Newt, Llik, ep = 0.0001, N.iter = 50)
{# fname is Itest
# alternates the use of golden section and Newton-Raphson
# to find 2-D mle's. The input N.iter puts a cap on the
# number of iterations of Newton-Raphson.
        recth <- NULL
        k <- 1
        repeat {if(k == 50) break
```

```
				th1 <- Newt(x, t, th0)
				if(max(abs(th0 - th1)) < ep)
					break
				recth <- rbind(recth, c(th0, th1))
				k <- k + 1
				th0 <- Golden(x, t, th0, th1, Llik)[[1]]}
			out <- c(recth[k - 1, 3:4], round(k))
			if(k == N.iter)
				out <- c("No convergence")
			return(out)}

Golden
function(x, t, th0, th1, Llik, ep1 = 0.5)
{# fname is Golden
# performs the golden section search along the direction
# th0 to th1. The function returns the value of th that
# maximizes Llik in that direction. The return is the
# new th and its (proportional) distance from th0.
# This function is called by Itest, from which it gets ep
	p1 <- c(0.618, 0.382)
	p2 <- rev(p1)   #thinit <- cbind(th0, th1)
	t0 <- th0
	len0 <- sqrt(sum((th1 - th0)^2))
	j <- 1
	repeat {j <- j + 1
		th4 <- p1 * th0 + p2 * th1
		th6 <- p2 * th0 + p1 * th1
		th <- cbind(th0, th4, th6, th1)
		L <- Llik(x, t, th)
		if(j > 5) {
			ep2 <- (max(L) - min(L))/20
			ep2 <- max(ep2, ep)
			ep1 <- min(ep1, ep2)}
		if(j == 100)
			break
		rnk <- rank(L) # print(L)
		if(rnk[3] == 4)
			th0 <- th4
		if(rnk[2] == 4)
			th1 <- th6
		if(rnk[4] == 4)
			th0 <- th4
		if(rnk[1] == 4)
			th1 <- th6
		if(max(abs(th0 - th1)) < ep1) break    # print(c(th0, th1))}
	len1 <- sqrt(sum((th1 - t0)^2))
```

```
        out <- list(th = th1, prop = len1/len0)
        if(j == 100) {
                out <- c("Too many Golden iterations")}
        return(out)}
```

Input functions when using the Gamma distribution.

Begin with the computation of the initializing point.

```
init.gam
function(x)
{# fname is init.gam
# initializes the estimates using the method of moments applied to
# the uncensored part of the data
# x <- dat[dat[, 2] == 1, 1]
        xb <- mean(x)
        s2 <- var(x)
        lam <- xb/s2
        alph <- lam * xb
        out <- c(alph, lam)
        out}

Llik.gam
function(x, t, th)
{# fname is Llik.gam
# computes the log likelihood for censored gamma data
# x in the uncensored life lengths, t is the censored set
# th is a matrix of parameter values;
# first row is alph, second is lam
        if(length(th) == 2) th <- matrix(th, 2, 1)
        k <- ncol(th)
        r <- length(x)
        lg <- lgamma(th[1,  ])
        lx <- sum(log(x))
        sx <- sum(x)
        L <- rep(0, k)
        for(i in 1:k) {
                S <- 1 - pgamma(th[2, i] * t, th[1, i], 1)
                L[i] <- r * (th[1, i] * log(th[2, i]) - r * lg[i]) +
                        (th[1, i] * -1) * lx - th[2, i] * sx + sum(log(S))}
        return(L)}

Newt.gam
function(x, t, th)
{# fname is Newt.gam
# program executes a single iterative update of th (theta)
# for the mle estimation of th (alph, lam) for censored gamma
# distribution data; x is the set of observed life times and t
# the set of censored (right) times.
        alph <- th[1]
        lam <- th[2]
        ps <- psi(alph)
        r <- length(x)
        S <- Surv.gam(alph, lam * t, 20, ep = 1e-005)
        La <- r * (log(lam) - ps[1]) + sum(log(x)) + sum(S[2,  ]/S[1,  ])
        f <- dgamma(lam * t, alph, 1)
        Slam <-  - t * f
        Ll <- (r * alph)/lam - sum(x) + sum(Slam/S[1,  ])
        Laa <-  - r * ps[2] + sum(S[3,  ]/S[1,  ] - (S[2,  ]/S[1,  ])^2)
        Salphlam <-  - t * f * (log(lam * t) - ps[1])
        Slamlam <-  - t^2 * f * ((alph - 1)/(lam * t) - 1)
        Lal <- r/lam + sum(Salphlam/S[1,  ] - (S[2,  ] * Slam)/S[1]^2)
        Lll <- ( - r * alph)/lam^2 + sum(Slamlam/S[1,  ] - (Slam/S[1,  ])^2)
        Lt <- c(La, Ll)
        H <- matrix(c(Laa, Lal, Lal, Lll), 2, 2)
        th0 <- th
        th <- th0 - solve(H, Lt)          # print(th)
        if(th[1] < 0)
                th[1] <- 0.01
        if(th[2] < 0)
                th[2] <- 0.01
        dist <- max(abs(th - th0))        # print(dist)
        return(th)}
```

The function `Surv.gam()` is described in Appendix D.

```
Lp.gam
function(x, t, th0)
{# fname is Lp.gam
# creates the gradient vector Lp and the Hessian matrix H
# for the censored gamma family sampling
        alph <- th[1]
        lam <- th[2]
        S <- Surv.gam(alph, lam * t, a0 = 20)
        xm1 <- (log(lam * x) - psi(alph)[1,  ])/lam
        xm2 <- alph/lam - x
        tm1 <- (S[2,  ] - psi(alph)[1,  ])/S[1,  ]
        tm2 <- ( - t * dgamma(lam * t, alph))/S[1,  ]
        Lp1 <- sum(xm1) + sum(tm1)
        Lp2 <- sum(xm2) + sum(tm2)
        H11 <- sum(xm1^2) + sum(tm1^2)
        H22 <- sum(xm2^2) + sum(tm2^2)
        H12 <- sum(xm1 * xm2) + sum(tm1 * tm2)
        out <- list(Lp = c(Lp1, Lp2), H = matrix(c(H11, H12, H12, H22), 2, 2))
        out}
```

Input functions when using the Lognormal distribution.

Initialize with mean(log(x)) and stdev(log(x))

```
Llik.logn
function(x, t, th)
{# fname is Llik.logn
# Computes the log likelihood for censored log normal data
# x is the set of uncensored life lengths, t is the censored set.
# th is the matrix of parameter values; first row is mu,
# second is sigma. The output is a set of log likelihood values.
        if(length(th) == 2) th <- matrix(th, 2, 1)
        k <- ncol(th)
        r <- length(x)
        d <- length(t)
        z1 <- outer(log(x), th[1,  ], "-")/th[2,  ]
        z2 <- outer(x, th[2,  ], "*")
        cc <- log(2 * pi)
        L1 <- apply( - log(z2) - 0.5 * cc - 0.5 * z1, 2, sum)
        w <- outer(log(t), th[1,  ], "-")/th[2,  ]
        Sn <- matrix(1 - pnorm(w), d, k)
        L2 <- apply(log(Sn), 2, sum)
        L <- L1 + L2
        return(L)}
```

```
Newt.logn
function(x, t, th)
{# fname is Newt.logn
# program executes a single iterative update of th (theta)
# for the mle estimation of th (mu, sig) for censored lognormal
# distribution data; x is the set of observed life times and t
# the set of (right) censored times.
        mu <- th[1]
        sig <- th[2]
        out <- Lp.logn(x, t, th)
        Lt <- out[[1]]
        H <- out[[2]]
        th0 <- th
        th <- th0 - solve(H, Lt)        # print(th)
        if(th[2] < 0)
                th[2] <- 0.01
        dist <- max(abs(th - th0))      # print(dist)
        return(th)}
```

```
Lp.logn
function(x, t, th0)
{# fname is Lp.logn
# Creates the gradient vector Lp and the Hessian matrix H
# for the censored lognormal family sampling.
# The output is a list; Lp and H
        mu <- th0[1]
        sig <- th0[2]
        z <- (log(x) - mu)/sig
        w <- (log(t) - mu)/sig
        fw <- dnorm(w)
        Sw <- 1 - pnorm(w)
        Lp1 <- sum(z) + sum(fw/(sig * Sw))
        Lp2 <- sum((z^2 - 1)/sig) + sum((w * fw)/(sig * Sw))
        H11 <- sum((z^2 - 1)/sig^2 - z^2) + sum((w * fw)/(sig^2 * Sw))
        H11 <- H11 - sum(fw/(sig * Sw)^2)
        H12 <- sum((z * (z^2 - 3))/sig^2 - (z * (z^2 - 1))/sig)
        H12 <- H12 - sum(((w^2 - 1) * fw)/(sig^2 * Sw))
        H22 <- sum(2 * (1 - z - 3 * z^2 + z^3) - ((z^2 - 1)/sig)^2)
        H22 <- H22 + sum((w * (w^2 - 2) * fw)/(sig^2 * Sw) - ((w * fw)/(sig * Sw))^2)
        out <- list(Lp = c(Lp1, Lp2), H = matrix(c(H11, H12, H12, H22), 2, 2))
        return(out)}
```

Appendix D

Derivatives of the Incomplete Gamma Function

The function `Surv.gam()` is needed in Section 5 for use in the parameter estimation programs for censored gamma data. The analysis is too lengthy to be included there and the development may have general interest. Basically we need to compute the first and second derivatives of the Incomplete Gamma function with respect to the shape parameter.

The formulas used are taken from [Abramowicz and Stegun] and their notation is adopted. We focus on [Abramowicz and Stegun, (6.5.1), 6.5.4), and (6.5.29)]

$$P(a, x) = \frac{1}{\Gamma(a)}\int_0^x y^{a-1}e^{-y}dy \tag{D.1}$$

$$\gamma^*(a,x) = x^{-a}\,P(a,x) = e^{-x}\sum_{j=0}^{\infty}\frac{x^j}{\Gamma(a+j+1)}. \tag{D.2}$$

The power series in (D.2) plays a key role. It will be used for values of x smaller than a. With this understanding, let us develop a crude, but useful, bound for the remainder after n terms.

$$\sum_{j=n}^{\infty}\frac{x^j}{\Gamma(a+j+1)} = \frac{1}{\Gamma(a+n)}\sum_{j=n}^{\infty}\frac{x^j}{(a+n)\cdots(a+j)} < \frac{(a+n)^n}{\Gamma(a+n)}\sum_{n}^{\infty}\frac{x^j}{(a+n)^j}$$

$$= \frac{x^n}{\Gamma(a+n)}\sum_{r=0}^{\infty}\left(\frac{x}{a+n}\right)^r = \frac{x^n}{\Gamma(a+n)}\frac{a+n}{a+n-x} < \frac{x^n}{\Gamma(a+n)}. \tag{D.3}$$

This inequality allows one to choose n given a, x with x < a so that the power series can be made as precise as desired.

The issue of x larger than a is managed by the recursive formula [Abramowicz and Stegun, (6.5.21)]

$$P(a+1, x) = P(a, x) - \frac{x^a e^{-x}}{\Gamma(a+1)}, \tag{D.4}$$

which, upon repeated use, yields

$$\text{Inc(r)} = P(a-r,x) - P(a,x) = e^{-x}\sum_{j=1}^{r}\frac{x^{a-j}}{\Gamma(a+1-j)}. \tag{D.5}$$

Viewing the goal as the computation of P(a-r, x), one may always choose r so that x < a and then the power series portion can finish the job. Such is the tactical plan. Both (D.2) and (D.5) may be differentiated termwise.

$$\gamma_a^*(a,x) = -e^{-x}\sum_{j=0}^{\infty}\frac{x^j}{\Gamma(a+j+1)}\Psi(a+j+1) \tag{D.6}$$

$$\gamma_{aa}^*(a,x) = e^{-x}\sum_{j=0}^{\infty}\frac{x^j}{\Gamma(a+j+1)}[\Psi^2(a+j+1)-\Psi'(a+j+1)] \tag{D.7}$$

$$Inc_a(r) = e^{-x}\sum_{j=1}^{r}\frac{x^{a-j}}{\Gamma(a+1-j)}[\ln(x)-\Psi(a+1-j)] \tag{D.8}$$

$$Inc_{aa}(r)=e^{-x}\sum_{j=1}^{r}\frac{x^{a-j}}{\Gamma(a+1-j)}\{[\ln(x)-\Psi(a+1-j)]^2-\Psi'(a+1-j)\}\,. \tag{D.9}$$

Using (D.1) and (D.2) we can express the survival function and derivatives of the standard gamma random variable.

$$\mathrm{S(a, x)} = 1 - \mathrm{x}^{\mathrm{a}}\gamma^*(\mathrm{a, x}) = 1 - \mathrm{e}^{-\mathrm{x}}\sum_{j=0}^{\infty}\frac{x^{a+j}}{\Gamma(a+j+1)} \tag{D.10}$$

$$\mathrm{S_a(a, x)} = -e^{-x}\sum_{j=0}^{\infty}\frac{x^{a+j}}{\Gamma(a+1+j)}[\ln(x)-\Psi(a+1+j)] \tag{D.11}$$

$$\mathrm{S_{aa}(a, x)} = -e^{-x}\sum_{j=0}^{\infty}\frac{x^{a+j}}{\Gamma(a+1+j)}\{[\ln(x)-\Psi(a+1+j)]^2-\Psi'(a+1+j)\} \tag{D.12}$$

Our goal is to orchestrate these formulas into a computational package usable for the censored sampling likelihood equations. Moreover, the package should accept vector input for x. The smaller values of x are easily managed using the power series. For other values it is wise to use the tactic of shifting to larger values of the parameter a using (D.5), (D.8), and (D.9). Accordingly, we partition the (a, x) plane into three horizontal strips

v1: $0 < x < 8$; v2: $8 \le x < a_0$; v3: $a_0 \le x$,

and a0 can be selected for power series truncation error control. Our programs use a0 = 20. Further, each strip is partitioned selectively for purposes of invoking the tactic of shifting to larger values of the parameter a. The rule in place is

v1: $a < a_0$; v2: $a < 2a_0$; v3: $a < 3a_0$.

The main program that executes these rules is Surv.gam(). Sixteen terms are used in the truncate power series. For these, Equations (D.10), (D.11), and (D.12) are computed using the functions

surv.gam(): surva.gam(); survaa.gam(), respectively.

When it is appropriate to shift to larger values of a, we invoke (D.5), (D.8), and (D.9). These functions are called

trans1(); trans2(); trans3(), respectively.

The output of Surv.gam is a three-row matrix and number of columns equal to the cardinality of the data x. The three rows contain the zeroth, first and second derivatives.

S-Plus Codes

```
Surv.gam
function(a, x, a0 = 20, ep = 1e-009)
{# fname is Surv.gam
# Integrates the programs surva.gam, survaa.gam
# and trans2, trans3 to comput the survival function
# and its derivatives wrt a and aa for the standard
# gamma distribution with shape parameter a
# and vector x. The input ep is for precision control.
        len <- length(x)
        S <- 1 - pgamma(x, a)
        v1 <- x < 8
        v2 <- 8 <= x & x < a0
        v3 <- a0 <= x
        Sa <- Saa <- rep(0, len)
        if(sum(v1 == T) > 0) {
                if(a >= a0) {
                        Sa[v1] <- surva.gam(a, x[v1], a0)
                        Saa[v1] <- survaa.gam(a, x[v1], a0)}
                else {
                        r1 <- ceiling(a0 - a)
                        Sa[v1] <- surva.gam(a + r1, x[v1], a0) - trans2(a, x[v1], r1)
                        Saa[v1] <- survaa.gam(a + r1, x[v1], a0) - trans3(a, x[v1], r1)}}
        if(sum(v2 == T) > 0) {
                if(a >= 2 * a0) {
                        Sa[v2] <- surva.gam(a, x[v2], a0)
                        Saa[v2] <- survaa.gam(a, x[v2], a0)}
                else {
                        r2 <- ceiling(2 * a0 - a)
                        Sa[v2] <- surva.gam(a + r2, x[v2], a0) - trans2(a, x[v2], r2)
                        Saa[v2] <- survaa.gam(a + r2, x[v2], a0) - trans3(a, x[v2], r2)}}
        if(sum(v3 == T) > 0) {
                if(a >= 3 * a0) {
                        Sa[v3] <- surva.gam(a, x[v3], a0)
                        Saa[v3] <- survaa.gam(a, x[v3], a0)}
                else {
                        r3 <- ceiling(3 * a0 - a)
                        Sa[v3] <- surva.gam(a + r3, x[v3], a0) - trans2(a, x[v3], r3)
                        Saa[v3] <- survaa.gam(a + r3, x[v3], a0) - trans3(a, x[v3], r3)}}
        out <- rbind(S, Sa, Saa)
        out}

surv.gam
function(a, x, a0)
{# fname is surv.gam
# creates first 16 terms of the power series expansion of
# the tail of the Incomplete gamma function. Useful
# when a is large; tmp is neg of gammastar
        ind <- 0:15
        k <- length(x)
        X <- outer(x, ind, "^")
        fac <- matrix(gamma(a + 1 + ind), k, 16, byrow = T)
        tmp <- exp( - x) * x^a * apply(X/fac, 1, sum)
        S <- 1 - tmp
        S}

surva.gam
function(a, x, a0)
{# fname is surva.gam
# creates first 16 terms of the power series expansion of
# the derivative of the Incomplete gamma function. Useful
# when a is large; tmp is neg of gammastar sub a
        ind <- 0:15
        k <- length(x)
        X <- outer(x, ind, "^")
        fac <- matrix(psi(a + 1 + ind)[1,  ]/gamma(a + 1 + ind), k, 16, byrow = T)
        tmp <- exp( - x) * x^a * apply(X * fac, 1, sum)
        Sa <- log(x) * ( - pgamma(x, a)) + tmp
        Sa}

survaa.gam
function(a, x, a0)
{# fname is survaa.gam
# deals with Saa for a large. Includes the sum of
# the first 16 terms in the power series expansion
# of x to the a times gammastar sub aa
        ind <- 0:15
        k <- length(x)
        X <- outer(x, a + ind, "^")     # Pa <- exp( - x) * apply(X/matrix(gamma(a + 1 +
ind), k, 16, byrow = T), 1, sum)
        brac <- outer(log(x), psi(a + 1 + ind)[1,  ], "-")
        den <- matrix(gamma(a + 1 + ind), k, 16, byrow = T)
        tmp <- matrix(psi(a + 1 + ind)[2,  ], k, 16, byrow = T)
        Saa <- exp( - x) * apply((X * (brac^2 - tmp))/den, 1, sum)
        Saa}

trans1
function(a, x, r)
{# fname is trans1
# finite series transition of standard gamma cdf
# from small shape parameter to large shape parameter.
# The output is P(a-r,x)-P(a,x)= Inc
        k <- length(x)
        ind <- 1:r
        N <- length(ind)
        X <- outer(x, a - ind, "^")
        den <- matrix(gamma(a + 1 - ind), k, N, byrow = T)
        S <- Inc <- exp( - x) * apply(X/den, 1, sum)
        S}

trans2
function(a, x, r)
{# fname is trans2
# finite series transition of the derivative of the standard
# gamma survivor function from small shape parameter to large
# shape parameter. The output is Sa(a+r,x)-Sa(a,x)=Inca
        k <- length(x)
        ind <- 1:r
        first <- matrix(0, r, k)
        for(j in 1:k)
                first[, j] <- dgamma(x[j], a + ind)
        sec <- matrix(0, r, k)
        for(j in 1:k)
                sec[, j] <- dgamma(x[j], a + ind) * psi(a + ind)[1,  ]
        Sa <- Inca <- log(x) * apply(first, 2, sum) - apply(sec, 2, sum)
        Sa}
trans3
function(a, x, r)
{# fname is trans3
# finite series transition of the 2nd derivative of the standard
# gamma survivor function from small shape parameter to large
# shape parameter. The output is Saa(a+r,x)-Saa(a,x)=Inca
        k <- length(x)
```

```
ind <- 1:r
dens <- matrix(0, k, r)
brac <- outer(log(x), psi(a + ind)[1,  ], "-")
for(i in 1:k)
        dens[i,  ] <- dgamma(x[i], a + ind)
sec <- matrix(psi(a + ind)[2,  ], k, r, byrow = T)
Saa <- Incaa <- apply(dens * (brac^2 - sec), 1, sum)
Saa}
```

Appendix E

Listings of Miscellaneous Code

```
gam.est
function(x, ep = 1e-006)
{# fname is gam.est; programmer: R. Read
# returns the mle estimates for the gamma distribution
# base upon the data x. Newton-Raphson iteration.
# output is maximum likelihood for shape, rate, n, method
# of moments for shape, rate
        n <- length(x)
        xb <- mean(x)
        lnbx <- mean(log(x))
        ss <- var(x)
        rate <- xb/ss
        lam0 <- rate    # initialize with meth of moments ests.
        shape <- xb^2/ss
        x0 <- lnbx - log(xb)
        alph <- shape
        repeat {
                r0 <- shape
                lam <- shape/xb
                g <- log(shape) + x0 - psi(shape)[1, ]
                gp <- 1/shape - psi(shape)[2, ]
                shape <- shape - g/gp
                if(abs(shape - r0) < ep)
                        break}
        rate <- shape/xb
        out <- c(shape, rate, n, alph, lam0)
        names(out) <- c("alpha-mle", "lambda-mle", "n", "alpha-mm", "lambda-mm")
        out}
```

The function ellipse returns a three-column matrix. Column 1 contains the horizontal and the other two columns provide the two lobes for the ellipse. The user must construct the plot.

```
ellipse
function(q, m, d, n0 = 100)
{# fname is ellipse. Revised December 2002
# q is the matrix of the quadratic form (2x2): (x, y)q(x, y)'
# m is the centering vector; (m[x], m[y])
# d is the (squared) distance value for the contour
# n0 is the half-size of the number of points in each lobe
# of the ellipse.
```

```
        a <- q[1, 1]
        b <- q[1, 2]
        c1 <- q[2, 2]
        det <- a * c1 - b * b
        x0 <- sqrt((c1 * d)/det) - 1e-008        # = max(x); latter term helps insure that the
fuzz won't stop the program
        x <- sqrt((seq(1, n0, 1) * x0)/n0) * sqrt(x0)   # Refines the partition for the
extreme values of x.
        x <- c( - rev(x), 0, x)
        rad <- sqrt(d * c1 - det * x * x)
        y1 <- ( - b * x)/c1 + rad/c1
        y2 <- ( - b * x)/c1 - rad/c1
        x <- x + m[1]
        y1 <- y1 + m[2]
        y2 <- y2 + m[2]
        z <- matrix(c(x, y1, y2), ncol = 3)
        z}
```

The following function produces Figure 3.1. Once executed, the user must point the cursor to a place in the lower right space of the plots and click. This will print the legend and close out the function.

```
exp1.lnorm
function(mu = 0, sig = 1)
{# fname is exp1.lnorm
# survival functions plotted to compare Exp(1) w/lognormal(mu, sig)
    x <- c(seq(0.02, 0.55, 0.01), seq(0.56, 3.6, 0.02))
    y <- dnorm((log(x) - mu)/sig)/(x * sig)
    Sl <- 1 - pnorm(log(x) - mu)/sig
    he <- rep(1, length(x))
    xx <- x[ - (1:27)]
    Se <- exp( - xx)
    yy <- exp( - xx)
    par1 <- par
    par(mfrow = c(3, 2))
    plot(x, y, ylim = c(0, 0.75), ylab = "density function", type = "l")
    lines(xx, yy, lty = 4)
    title(main = "Compare Density functions    ")
    plot(x, exp( - x), ylab = "survivor function", type = "l", lty = 4)
    title(main = "Compare Survivor Functions    ")
    lines(x, Sl)
    plot(x, y/Sl, ylim = c(0, 1.1), ylab = "hazard function", type = "l")
    lines(x, he, lty = 4)
    title(main = "Compare Hazard Functions    ")
    plot(x, x, ylab = "cum haz function", type = "l", lty = 4)
```

```
     lines(x, - log(Sl))
     title(main = "Compare Cum. Hazard Functions    ")
     SSl <- exp(mu + 0.5 * sig) * (1 - pnorm(x - sig)) - exp(mu + sig * x) *
(1 - pnorm(x))
     Ll <- SSl/Sl
     a <- max(Ll)
     b1 <- min(x)
     b2 <- max(x)
     plot(x, Ll, ylim = c(0, 1.5 * a), ylab = "expected resid. life", type = "l")
     lines(x, he, lty = 4)
     title(main = "Compare E{Residual Life}    ")
     x <- c(0, 1, 1, 0, 0)
     y <- rev(x)      # plot(box(), xlab = "", ylab = "", axes = F)
     par(mar = c(4.9, 3, 3.8, 0.9))
     plot(x, y, type = "l", xlab = "", ylab = "", axes = F)
     lines(c(0, 0), c(1, 0), lty = 1)
     lines(c(0, 1), c(1, 1), lty = 4)  #text(0.4, 0.4, Ltext)
     title(main = "Legend")
     par <- par1
     leg.names <- c("Lognormal(0, 1) ", "    mean = 1.65   stdev  =  1.68",
"Exponential(1) ",
               "    mean  = 1        stdev  =  1")
     legend(locator(1), leg.names, lty = c(1, 0, 4, 0))}
```

The following code produces Figure 3.2. Once plotted the user must click on the legend box in order to place information there and close out the function.

```
wei1.gam
function(alph = 0.75, beta = 2)
{# fname is wei1.gam
# survival functions plotted to compare Weibull(3/4, 2) w/gamma(.72, rate= .28)
# gamma parameters are mle's from simulated rweibull(200, 3/4, 2)
     x <- c(0.01, seq(0.02, 15, 0.08))
     y <- dweibull(x, 0.75, 2)
     Sw <- 1 - pweibull(x, 0.75, 2)
     Sg <- 1 - pgamma(x, 0.72, 0.28)
     yy <- dgamma(x, 0.72, 0.28)
     par1 <- par
     par(mfrow = c(3, 2))
     plot(x, y, ylim = c(0, 1.45), ylab = "density function", type = "l")
     lines(x, yy, lty = 4)
     title(main = "Compare Density functions    ")
     plot(x, Sw, ylab = "survivor function", type = "l")
     title(main = "Compare Survivor Functions    ")
     lines(x, Sg, lty = 4)
```

```
    plot(x, y/Sw, ylim = c(0, 1.45), ylab = "hazard function", type = "l")
    lines(x, yy/Sg, lty = 4)
    title(main = "Compare Hazard Functions    ")
    plot(x,  - log(Sw), ylab = "cum haz function", type = "l", ylim = c(0, 5))
    lines(x,  - log(Sg), lty = 4)
    title(main = "Compare Cum. Hazard Functions    ")
    SSw <- (8/3) * gamma(4/3) * (1 - pgamma((0.5 * x)^0.75, 4/3))
    Lw <- SSw/Sw
    plot(x, Lw, ylab = "expected resid. life", type = "l")
    SSg <-  - x * Sg + (0.72/0.28) * (1 - pgamma(x, 1.72, 0.28))
    Lg <- SSg/Sg
    lines(x, Lg, lty = 4)
    title(main = "Compare E{Residual Life}    ")
    x <- c(0, 1, 1, 0, 0)
    y <- rev(x)      # plot(box(), xlab = "", ylab = "", axes = F)
    par(mar = c(4.9, 3, 3.8, 0.9))
    plot(x, y, type = "l", xlab = "", ylab = "", axes = F)
    lines(c(0, 0), c(1, 0), lty = 1)
    lines(c(0, 1), c(1, 1), lty = 4)  #text(0.4, 0.4, Ltext)
    title(main = "Legend")
    par <- par1
    leg.names <- c("Weibull(0.75, 2) ", "    mean = 1.84   stdev  =  2.33",
"Gamma(.72, .28) ", "    mean = 2.57   stdev  =  3.03")
    legend(locator(1), leg.names, lty = c(1, 0, 4, 0))}
```

The following function produces Figure 3.3. Once plotted the user must click on the legend box in order to place information there and close out the function.

```
wei2.gam
function(alph = 2, beta = 2)
{# fname is wei2.gam
# survival functions plotted to compare Weibull(2, 2) w/gamma(3.75, rate = 2)
# gamma parameters are close to the mle's from simulated rweibull(200, 2, 2)
    x <- seq(0, 5, 0.025)
    y <- dweibull(x, 2, 2)
    Sw <- 1 - pweibull(x, 2, 2)
    Sg <- 1 - pgamma(x, 3.75, 2)
    yy <- dgamma(x, 3.75, 2)
    par1 <- par
    par(mfrow = c(3, 2))
    plot(x, y, ylim = c(0, 0.48), ylab = "density function", type = "l")
    lines(x, yy, lty = 4)
    title(main = "Compare Density functions    ")
    plot(x, Sw, ylab = "survivor function", type = "l")
    title(main = "Compare Survivor Functions    ")
    lines(x, Sg, lty = 4)
    plot(x, y/Sw, ylab = "hazard function", type = "l")
    lines(x, yy/Sg, lty = 4)
    title(main = "Compare Hazard Functions    ")
    plot(x,  - log(Sw), ylab = "cum haz function", type = "l")
    lines(x,  - log(Sg), lty = 4)
    title(main = "Compare Cum. Hazard Functions    ")
    SSw <- gamma(0.5) * (1 - pgamma((0.5 * x)^2, 0.5))
    Lw <- SSw/Sw
    yM <- max(Lg)
    SSg <-  - x * Sg + (3.75/2) * (1 - pgamma(x, 4.75, 2))
    Lg <- SSg/Sg
    ym <- min(Lw)
    plot(x, Lw, ylab = "expected resid. life", type = "l", ylim = c(ym, yM))
    lines(x, Lg, lty = 4)
    title(main = "Compare E{Residual Life}    ")
    x <- c(0, 1, 1, 0, 0)
    y <- rev(x)      # plot(box(), xlab = "", ylab = "", axes = F)
    par(mar = c(4.9, 3, 3.8, 0.9))
    plot(x, y, type = "l", xlab = "", ylab = "", axes = F)
    lines(c(0, 0), c(1, 0), lty = 1)
    lines(c(0, 1), c(1, 1), lty = 4)  #text(0.4, 0.4, Ltext)
    title(main = "Legend")
    par <- par1
    leg.names <- c("Weibull(2, 2) ", "   mean = 1.77   stdev = 2.18", "Gamma(3.75, 2)
", "   mean = 1.88   stdev = 0.97")
    legend(locator(1), leg.names, lty = c(1, 0, 4, 0))}

wei.cost
function(rat, alpha, beta)
{# fname is wei.cost
# rat is the ratio of
    xx <- ceiling(log(500))
    x2 <- beta * xx^(1/alpha)
    x <- seq(0, x2, length = 200)
    y <- (x/beta)^alpha
    cost <- ((alpha/beta) * (1 + rat * pgamma(y, 1)))/(gamma(1/alpha) * pgamma(y,
1/alpha))
    m <- min(cost)
    M <- max(x)
    ind <- match(m, cost) # print(x[ind], m)
# plot(x, cost, type = "l")
    out <- c(ind, M, m)
    out}

gam.cost
```

```
function(rat, alpha, lam)
{# fname is gam.cost
# rat is the ratio
# alpha & lam are the parameters of the gamma distribution
# output is the min cost planned replacement policy
    x2 <- alpha/lam + (3 * sqrt(alpha))/lam
    x <- seq(0.02, x2, length = 200)
    Sg <- 1 - pgamma(x, alpha, lam)
    cost <- (1 + rat * pgamma(x, alpha, lam))/(x * Sg + (alpha/lam) * pgamma(x,
1 + alpha, lam))
    m <- min(cost)
    M <- max(x)
    ind <- match(m, cost)
    x0 <- round(x[ind], 2)
    print(c(ind, x0, m))     # plot(x, cost, type = "l")
    out <- c(ind, x0, M, m)
    out}

NormCT
function(x, alph, graph = T)
{# fname is NormCT
# this is a plotting function that produces 1-alph level
# confidence trapezoids for (mu, sig) for normal data x.
# If graph = F then the output is a row matrix containing
# the vertices of the trapezoid.
    out <- NULL
    n <- length(x)
    alph1 <- 1 - sqrt(1 - alph)
    z0 <- - qnorm(alph1/2)
    k1 <- qchisq(alph1/2, n - 1)
    k2 <- qchisq(1 - alph1/2, n - 1)
    SS <- (n - 1) * var(x)
    xb <- mean(x)
    s <- stdev(x)
    sigl <- sqrt(SS/k2)
    sigu <- sqrt(SS/k1)
    xlmin <- xb - (z0 * sigl)/sqrt(n)
    xlmax <- xb + (z0 * sigl)/sqrt(n)
    xumin <- xb - (z0 * sigu)/sqrt(n)
    xumax <- xb + (z0 * sigu)/sqrt(n)
    xx <- c(xlmin, xlmax, xumax, xumin, xlmin)
    yy <- c(sigl, sigl, sigu, sigu, sigl)
    if(graph == T) {
            plot(xx, yy, type = "l", xlab = "mu", ylab = "sigma")
            title(main = "Confidence Trapezoid for Normal Distribution")
            points(xb, s)}
```

```
     else out <- rbind(xx, yy)
     return(out)}


area.comp
function(x, w, u0 = 0, y0 = 0, rnd = 4)
{# fname is area.comp
# Computes the signed net areas separating the empirical
# cdf's of the ordered sets x and w. These cdf's are polygonal
# curves which are connected with straight line segments. The
# two data sets are of the same length. It seems necessary to
# do some rounding because of "fuzz" problems; hence the input
# rounding quantity, "rnd".
     n0 <- length(x)
     x <- round(x, rnd)
     w <- round(w, rnd)
     out <- matrix(0, ceiling((1 + n0)/2), 5)
     jj <- 1
     repeat {
          out[jj, ] <- seg.comp(x, w, u0, y0, n0, jj)
          u0 <- round(out[jj, 4], rnd)
          tmp <- round(out[jj, 5], rnd)
          y0 <- tmp - floor(tmp)
          x <- x[x > u0]
          w <- w[w > u0]
          n1 <- length(x)
          n2 <- length(w)
          if(n1 == 0 | n2 == 0)
               break
          if(all(x == w))
               break
          jj <- jj + 1}
     out <- out[1:jj, ]
     out}


seg.comp
function(x, w, u0, y0, n0, jj)
{# fname is seg.comp
# Computes the areas under the polygonal curves, between
# two knots, and returns their difference (signed). A flag
# is set = 0 if the x cdf is above the w cdf, and set = 1
# otherwise. The x and w vectors are mono increasing; n is
# the number of points in the full sets. The initial points
# (u0, y0) mark the beginning of the segment; the crossover
# point (u1, y1) is the segment end and is computed internally;
# when segment is open ended, it is (x[n], n). The marker f1
# is set to one when the segment is close on the right.
# f2 = 1 means the right knot is not a data point.
    n <- f <- y1 <- length(x)        # initialization
    f1 <- flag <- rect <- f2 <- 0
    u1 <- max(x[1], w[1])
    ss <- sort(c(x, w))      # first look for superpositions and remove
    repeat {
            if(x[1] != w[1])
                    break
            u0 <- x[1]
            x <- x[-1]
            w <- w[-1]
            ss <- ss[ - (1:2)]
            n <- f <- n - 1
            if(n == 1 | n == 0) {
                    f <- 1
                    y1 <- 1
                    break}
            else {
                    if(x[1] == w[1]) {
                            ss <- ss[ - (1:2)]
                            u0 <- x[1]
                            x <- x[-1]
                            w <- w[-1]
                            n <- f <- y1 <- length(x)
                            if(n == 0)
                              break }}}
# Set the flag & initiate
    if(n > 0)
            j <- 1:n
    if(length(ss) > 0) {
            ind <- j[x[j] >= w[j]]
            if(w[1] == ss[1]) {
                    flag <- 1
                    ind <- j[w[j] >= x[j]]   }
            if(is.na(ind[1]) == F) {
                    f1 <- 1
                    f <- ind[1]}}
# initialize the end corrections and the center computations
    area1 <- area2 <- adj1 <- adj2 <- 0
    u1 <- max(x[f], w[f])
    y1 <- f
    if(f > 1 & f1 == 1) {
            P1 <- c(x[(f - 1):f])
            P2 <- c(w[(f - 1):f])
```

```
                tout <- sol.pt(P1, P2)
                u1 <- tout[1]
                y1 <- tout[2] + f
                if(x[f] != w[f]) {
                        f2 <- 1
                        f <- f - 1}}
# prepare for the open ended case
        if(n > 0) {rect <- f * abs(x[n] - w[n])
                j <- 1:n
                area1 <- ((x[1] - u0) * (1 + y0))/2
                area2 <- ((w[1] - u0) * (1 + y0))/2
                if(f > 1) {area1 <- area1 + ((f1 * f2 * (u1 - x[f - 1]) * (y1 + f - 1))/2)
                        area2 <- area2 + ((f1 * f2 * (u1 - w[f - 1]) * (y1 + f - 1))/2)
                        ff <- f  # prep adjustment for existence of interior part of segment
                        if(x[f] != w[f])
                                ff <- ff - 1
                        adj1 <- 0.5 * (x[ff] * (2 * ff + 1) - x[1]) - sum(x[1:ff])
                        adj2 <- 0.5 * (w[ff] * (2 * ff + 1) - w[1]) - sum(w[1:ff])        }
                area1 <- area1 + (1 - f1) * (1 - flag) * rect
                area2 <- area2 + (1 - f1) * flag * rect }
        net <- (area1 + adj1 - area2 - adj2)/n0
        out <- c(net, flag, f, u1, y1)
        names(out) <- c("net", "flag", "seg.length", "hend", "vend")
        out}



sol.pt
function(P1, P2)
{# fname is sol.pt
# finds the crossover solution point
# for two cdf's that have the same number
# of pts in the horiz & equi spaced in the vertical.
# If both delx and delq are zero, then y is set to
# y = -1. Otherwise 0 < = y < 1
        x1 <- P1[1]
        x2 <- P1[2]
        w1 <- P2[1]
        w2 <- P2[2]
        delx <- x2 - x1
        delw <- w2 - w1
        if(delx == 0 & delw == 0) {
                x <- x1
                y <- -1 }
        if(delx == 0 & delw > 0) {
                x <- x1
```

```
            y <- (x - w1)/delw      }
    if(delx > 0 & delw == 0) {
            x <- w1
            y <- (x - x1)/delx      }
    if(delx > 0 & delw > 0) {
            x <- (x1 * delw - w1 * delx)/(delw - delx)
            y <- (x - w1)/delw
            if(x1 == w1)
                    y <- 0  }
    out <- c(x, y)
    out}

sol.pt
function(P1, P2)
{# fname is sol.pt
# finds the crossover solution point
# for two cdf's that have the same number
# of pts in the horiz & equi spaced in the vertical.
# If both delx and delq are zero, then y is set to
# y = -1. Otherwise 0 < = y < 1
    x1 <- P1[1]
    x2 <- P1[2]
    w1 <- P2[1]
    w2 <- P2[2]
    delx <- x2 - x1
    delw <- w2 - w1
    if(delx == 0 & delw == 0) {
            x <- x1
            y <- -1 }
    if(delx == 0 & delw > 0) {
            x <- x1
            y <- (x - w1)/delw}
    if(delx > 0 & delw == 0) {
            x <- w1
            y <- (x - x1)/delx}
    if(delx > 0 & delw > 0) {
            x <- (x1 * delw - w1 * delx)/(delw - delx)
            y <- (x - w1)/delw
            if(x1 == w1)
                    y <- 0}
    out <- c(x, y)
    out}

Newt.wei
function(x, t, param, ep = 0.0001)
{# fname is Newt.wei
```

```
# Newton-Raphson method, then bisection search used to find mle
# for the Weibull distribution using the extreme value distribution technique.
# Data input is (x, t); param initialization is (u, b), b > 0
# output is (u, b, alph, beta). r is cardinality of uncensored set
    r <- length(x)
    v <- log(x)
    vb <- mean(v)
    lt <- log(t)
    b <- param[2]
    u <- param[1]
    vv <- c(v, lt)
    b2 <- b1 <- b
    h1 <- h2 <- 0
    j <- 1
    flag <- 1
    repeat {
            if(flag == 1) {
                    D <- sum(exp(vv/b))
                    N <- sum(vv^2 * exp(vv/b))
                    A <- sum(vv * exp(vv/b))
                    h <- A/D - b - vb
                    hp <-  - N/(D * b^2) + A^2/(D * b)^2 - 1
                    b <- b1 - h/hp
                    if(b <= 0)
                            b <- 0.01
                    if(max(abs(h), abs(b - b1)) < ep)
                            break
                    if(j/2 != round(j/2)) {
                            b1 <- b
                            h1 <- h}
                    if(j/2 == round(j/2)) {
                            b2 <- b
                            h2 <- h}
                    j <- j + 1
                    if(sign(h2 * h1) < 0)
                            flag <- 0}
            if(j == 25)
                    break
            if(flag == 0) {
                    b <- (b1 + b2)/2
                    D <- sum(exp(vv/b))
                    A <- sum(vv * exp(vv/b))
                    h <- A/D - b - vb
                    if(sign(h * h1 < 0)) {
                            h2 <- h
                            b2 <- b}
                    if(sign(h * h2 < 0)) {
                            h1 <- h
                            b1 <- b}
                    if(max(abs(h), abs(b - b1)) < ep)
                            break
                    j <- j + 1
                    if(j == 50)
                            break} }
    u <- b * log(D/r)
    alph <- 1/b
    beta <- exp(u)
    out <- c(u, b, alph, beta, flag)
    names(out) <- c("u", "b", "alph", "beta", "flag")
    return(out)}


exp.lnorm
function(mu = 0, sig = 1)
{# fname is exp.lnorm
# survival functions plotted to compare Exp(1) w/lognormal(mu, sig)
    x <- c(seq(0.02, 0.55, 0.01), seq(0.56, 3.6, 0.02))
    y <- dnorm((log(x) - mu)/sig)/(x * sig)
    Sl <- 1 - pnorm(log(x) - mu)/sig
    he <- rep(1, length(x))
    xx <- x[ - (1:27)]
    Se <- exp( - xx)
    yy <- exp( - xx)
    split.screen(c(3, 2))
    oldpar <- par()
    on.exit(par(oldpar))
    screen(1)
    par(cex = 0.8, mar = c(5, 6, 4, 2))
    plot(x, y, ylim = c(0, 0.75), ylab = "density function", type = "l")
    lines(xx, yy, lty = 4)
    title(main = "Compare Density functions    ")
    screen(2)
    par(cex = 0.8, mar = c(5, 6, 4, 2))
    plot(x, exp( - x), ylab = "survivor function", type = "l", lty = 4)
    title(main = "Compare Survivor Functions    ")
    lines(x, Sl)
    screen(3)
    par(cex = 0.8, mar = c(5, 6, 4, 2))
    plot(x, y/Sl, ylim = c(0, 1.1), ylab = "hazard function", type = "l")
    lines(x, he, lty = 4)
    title(main = "Compare Hazard Functions    ")
    screen(4)
    par(cex = 0.8, mar = c(5, 6, 4, 2))
    plot(x, x, ylab = "cum haz function", type = "l", lty = 4)
    lines(x,  - log(Sl))
    title(main = "Compare Cum. Hazard Functions    ")
    screen(5)
    par(cex = 0.8, mar = c(5, 6, 4, 2))
    SSl <- exp(mu + 0.5 * sig) * (1 - pnorm(x - sig)) - exp(mu + sig * x) *
(1 - pnorm(x))
    Ll <- SSl/Sl
    a <- max(Ll)
    b1 <- min(x)
    b2 <- max(x)
    plot(x, Ll, ylim = c(0, 1.5 * a), ylab = "expected resid. life", type = "l")
    lines(x, he, lty = 4)
    title(main = "Compare E{Residual Life}    ")
    screen(6)
    par(cex = 0.8, mar = c(5, 6, 4, 2))
    x <- c(0, 1, 1, 0, 0)
    y <- rev(x)      # plot(box(), xlab = "", ylab = "", axes = F)
    plot(x, y, type = "n", xlab = "", ylab = "", axes = F)  ##       lines(c(0, 0), c(1, 0),
lty = 1)
###         lines(c(0, 1), c(1, 1), lty = 4)  #text(0.4, 0.4, Ltext)
    leg.names <- c("Lognormal(0, 1)", "mean = 1.65", "sd = 1.68", "Exponential(1)",
"mean = 1", "sd = 1", "", "")
    leg.xy <- locator(1)
    legend(leg.xy$x, leg.xy$y, leg.names, lty = c(1, 0, 0, 4, 0, 0, 0, 0), cex = 1)
    close.screen(all = T)}

ext.newt
function(input, dat, ep = 10^-4)
{# fname is ext.newt
# Newton's method applied to extreme value distribution
    u <- u0 <- input[1]
    b <- b0 <- input[2]
    x <- dat[dat[, 2] == 1, 1]
    a <- dat[, 2]
    w <- a * dat[, 1] + (1 - a) * dat[, 3]
    r <- sum(a)
    g0 <- 0
    repeat {g <- sum(w * exp(w/b))/sum(exp(w/b)) - b - mean(x)
            g1 <- sum(w^2 * exp(w/b))/sum(exp(w/b))
            g2 <- sum(w * exp(w/b))/sum(exp(w/b))
            gp <-  - (g1 + g2^2 - 1)/b^2
            b <- b0 - g/gp
            if(b < 0)
                    b <- 0.2
```

```
                if(max(abs(b - b0), abs(g - g0)) < ep)
                        break
                g0 <- g
                b0 <- b}
        u <- b * log(sum(exp(w/b))/r)
        return(c(u, b))}

cost.comp
function(rat)
{# fname is cost.comp
# Compares the cost of planned replacement curves of our two competing IFR models
Weibull(2, 2) & gamma(3.75, rate = 2)
# Three sets of curves are generated, one for each of the n ratios in the input rat
        n <- length(rat)
        aw <- 2
        bw <- 2
        xx <- log(500)
        x2 <- 0.25 * bw * xx * (1/aw)
        ag <- 3.75
        lam <- 2
        x1 <- ag/lam + (1 * sqrt(ag))/lam
        x3 <- max(x1, x2)       # range of variate values
        x <- seq(0, x3, length = 200)
        y <- (x/bw)^aw          #y <- dweibull(x, 2, 2)
        Sw <- 1 - pweibull(x, 2, 2)
        Sg <- 1 - pgamma(x, 3.75, 2) #yy <- dgamma(x, 3.75, 2)
        par1 <- par
        par(mfrow = c(n, 1))
        numw <- (aw/bw) * (1 + outer(rat, pgamma(y, 1), "*"))
        denw <- matrix(gamma(1/aw) * pgamma(y, 1/aw), byrow = T, nrow = n, ncol =
200)
        costw <- numw/denw
        numg <- 1 + outer(rat, pgamma(x, ag, lam), "*")
        deng <- matrix(x * Sg + (ag/lam) * pgamma(x, 1 + ag, lam), byrow = T, nrow =
n, ncol = 200)
        costg <- numg/deng
        for(j in 1:n) {
                plot(x, costw[j,  ], ylab = c("rel. cost"), xlab = c("time in service"), type =
"l")
                lines(x, costg[j,  ], lty = 4)
                title(main = paste("ratio = ", rat[j]))   }
        mcostw <- apply(costw, 1, min)
        mcostg <- apply(costg, 1, min)
        out <- rbind(mcostw, mcostg)#x <- c(0, 1, 1, 0, 0)
# y <- rev(x)           # plot(box(), xlab = "", ylab = "", axes = F)
# par(mar = c(4.9, 3, 3.8, 0.9))
# plot(x, y, type = "l", xlab = "", ylab = "", axes = F)
# lines(c(0, 0), c(1, 0), lty = 1)
# lines(c(0, 1), c(1, 1), lty = 4)     #text(0.4, 0.4, Ltext)
# title(main = "Legend")
    par <- par1      #leg.names <- c("Weibull(2, 2) ", "   mean = 1.77   stdev = 2.18",
#    "Gamma(3.75, 2) ", "   mean = 1.88   stdev = 0.97")
# legend(locator(1), leg.names, lty = c(1, 0, 4, 0))
    out}

gam.cost
function(rat, alpha, lam)
{# fname is gam.cost
# rat is the ratio
# alpha & lam are the parameters of the gamma distribution
# output is the min cost planned replacement policy
    x2 <- alpha/lam + (3 * sqrt(alpha))/lam
    x <- seq(0.02, x2, length = 200)
    Sg <- 1 - pgamma(x, alpha, lam)
    cost <- (1 + rat * pgamma(x, alpha, lam))/(x * Sg + (alpha/lam) * pgamma(x,
1 + alpha, lam))
    m <- min(cost)
    M <- max(x)
    ind <- match(m, cost)
    x0 <- round(x[ind], 2)
    print(c(ind, x0, m))     #plot(x, cost, type = "l")
    out <- c(ind, x0, M, m)
    out}

ks.dist
function(dat, mod)
{# fname is ks.dist
# the distance between the data cdf and the model cdf
# using the Kolmorgoroc-Smirnov distance function. The
# values of mod must be the cdf of the model evaluated
# at the values of dat.
    n <- length(dat)
    dat <- sort(dat)
    mod <- sort(mod)
    p <- (1:n)/n
    pl <- c(0, p[ - n])
    ks <- max(abs(c(mod - p, pl - mod)))
    return(ks)}

Gsq.norm
function(x, y, xb, s, n)
{# fname is Gsq.norm
```

```
# This program computes the vertical component, z, of the
# G-squared statistic over the grid work that results
# from the x, y values, using the normal likelihood
# function. Thus, one can construct contour plots of
# (x, y, z). xb and s are the mle's of mu and sigma from
# a random sample of size n. x values go with mu;
# y with sigma
    SS <- n * s^2
    tmp1 <- 2 * n * log(y/s) + SS/y^2 - n
    tmp2 <- n * outer((xb - x)^2, y^2, "/")
    zz <- tmp1 + t(tmp2)
    z <- t(zz)
    return(z)}


Gsq.gam
function(dat, x, y, ah, bh, n)
{# fname is Gsq.gam
# This program computes the vertical component, z, of the
# G-squared statistic over the grid work that results
# from the x, y values, using the gamma(alpha, beta)likelihood
# function. Thus, one can construct contour plots of
# (x,y,z). ah and bh are the mle's of alpha and beta from
# a random sample, dat, of size n. x values go with alpha;
# y with beta
    xb <- mean(dat)
    blx <- mean(log(dat))
    za <-  - lgamma(ah) - ah * log(bh) + ah * blx - ah     # scalar term
    z1 <- lgamma(x) - x * blx      # vector terms
    z21 <- outer(x, log(y), "*")
    z22 <- xb * matrix(1/y, length(x), length(y), byrow = T)
    zz <- za + z1 + z21 + z22
    out <- 2 * n * zz
    return(out)}

Gsq.wei
function(dat, x, y, ah, bh, n)
{# fname is Gsq.wei
# This program computes the vertical component, z, of the
# G-squared statistic over the grid work that results
# from the x, y values, using the Weibull likelihood
# function. Thus, one can construct contour plots of
# (x, y, z). ah and bh are the mle's of shape and scale
# a random sample, dat, of size n. x values go with ah;
# y with bh
    blx <- mean(log(dat))
```

```
z0 <- log(ah) - ah * log(bh) + ah * blx - 1      # scalars
z1 <- log(x) + x * blx  # vectors
v <- outer(x, log(y), "*")
xba <- apply(outer(dat, x, "^"), 2, sum)/n
zz <- z0 - z1 + v + xba * exp( - v)
z <- 2 * n * zz
return(z)}
```

REFERENCES

Abramowitz, M. and Stegun, I.R. (1964). *Handbook of Mathematical Functions*, National Bureau of Standards, Applied Mathematics Series 55.

Ascher, H. and Feingold, H. (1984). *Repairable Systems Reliability*, Marcel Dekker.

Barlow, R.E. and Proschan, F. (1981). *Statistical Theory of Reliability and Life Testing*, Wiley.

Barndorff-Nielsen, O.E. and Cox, D.R. (1994). *Inference and Asymptotics*, Chapman and Hall.

Billmann, B., Antle, C., and Bain, L.J. (1972). "Statistical Inferences from Censored Weibull Samples," Technometrics, **14**, pp. 831-840.

Chambers, J.M. (1977). *Computational Methods for Data Analysis*, Wiley.

Cox, D.R. and Hinckley, D.V. (1974), *Theoretical Statistics*, Chapman and Hall, p. 309.

Erdelyi, A. (1955). *Asymptotic Expansions*, Dover.

Gross, A.J. and Clark, V.A. (1975). *Survival Distributions: Reliability Applications in the Biomedical Sciences*, Wiley.

Harter, H.L. and Moore, A.H. (1967). "Asymptotic Variances and Covariances of Maximum Likelihood Estimators, from Censored Samples, of the Parameters of the Gamma and Weibull Distributions," Annals of Math. Stat., **38**, pp. 557-570.

Kalbfleisch, J.D. and Prentice, R.L. (1980). *The Statistical Analysis of Failure Time Data*, Wiley.

Kumamoto, H. and Henley, E.J. (1996). "Probabilistic Risk Assessment and Management for Engineers and Scientists," *IEEE Press*.

Lawless, J.F. (1981). *Statistical Models and Methods for Lifetime Data*, Wiley.

Lawless, J.F. (1975). "Construction of Tolerance Bounds for the Extreme Value and Weibull Distributions," Technometrics, **177**, pp. 255-261.

Leemis, L.M. (1995). *Reliability*, Prentice Hall.

Leitch, R.D. (1995). *Reliability Analysis for Engineers*, Oxford.

Lewis, E.E. (1996). *Introduction to Reliability Engineering*, Wiley.

Meeker, W.A. and Escobar, L.A. (1998). *Statistical Methods for Reliability Data*, Wiley.
Modarres, M. (1993). *Reliability and Risk Analysis*, Dekker.

O'Conner, P.D.T. (1981). *Practical Reliability Engineering*, Heyden.

Proschan, F. (1963). "Theoretical Explanation of observed Decreasing Failure Rate," Technometrics, **5**, pp. 375-383.

Wilk, M.B., Gnanadesikan, R., and Huyett, M.J. (1962). "Estimation of Parameters of the Gamma Distribution Using Order Statistics," Biometrika, **49**, pp. 525-545.

INITIAL DISTRIBUTION LIST

1. Research Office (Code 09)..1
   Naval Postgraduate School
   Monterey, CA 93943-5000

2. Dudley Knox Library (Code 013)...2
   Naval Postgraduate School
   Monterey, CA 93943-5002

3. Defense Technical Information Center..2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

4. Richard Mastowski (Editorial Assistant)...2
   Department of Operations Research
   Naval Postgraduate School
   1411 Cunningham Road
   Monterey, CA 93943-5219

5. Professor Robert R. Read (Code OR/Re) 3
   Department of Operations Research
   Naval Postgraduate School
   1411 Cunningham Road
   Monterey, CA 93943-5219

6. Associate Professor Samuel E. Buttrey (Code OR/Sb) 1
   Department of Operations Research
   Naval Postgraduate School
   1411 Cunningham Road
   Monterey, CA 93943-5219

7. Assistant Professor Robert A. Koyak (Code OR/Kr) 1
   Department of Operations Research
   Naval Postgraduate School
   1411 Cunningham Road
   Monterey, CA 93943-5219

8. Associate Professor Lyn R. Whitaker (Code OR/Wh) 1
   Department of Operations Research
   Naval Postgraduate School
   1411 Cunningham Road
   Monterey, CA 93943-5219

9. Senior Lecturer David Olwell (Code OR/Ol) 1
Department of Operations Research
Naval Postgraduate School
1411 Cunningham Road
Monterey, CA 93943-5219

10. Distinguished Professor Donald P. Gaver, Jr. (Code OR/Gv) 1
Department of Operations Research
Naval Postgraduate School
1411 Cunningham Road
Monterey, CA 93943-5219

11. Professor Patricia A. Jacobs (Code OR/Jc)..1
Department of Operations Research
Naval Postgraduate School
1411 Cunningham Road
Monterey, CA 93943-5219

12. Professor Moshe Kress (Code OR)..1
Department of Operations Research
Naval Postgraduate School
1411 Cunningham Road
Monterey, CA 93943-5219

13. Professor Alan R. Washburn (Code OR/Wa) 1
Department of Operations Research
Naval Postgraduate School
1411 Cunningham Road
Monterey, CA 93943-5219

14. Major Scott G. Frickenstein, USAF..1
Chief, Commander's Action Group
Office of the Superintendent
HQ, USAFA/CCX
2304 Cadet Drive, Suite 342
USAF Academy, CO 80840-5001