**Estimating the median of the one parameter**
**Weibull distribution**

H.J. Newton
Department of Statistics, Texas A&M University

Abstract

We compare the sample median $\hat{M}$ and maximum likelihood estimator $\tilde{M}$ of the median of the one parameter Weibull distribution based on a random sample $X_1, \ldots, X_n$ of size $n$. We do this both analytically and via a simulation study. The results show that $\tilde{M}$ is far superior to $\hat{M}$. The Splus and Fortran code used in the paper are included, and we show that judicious use of Fortran can give great savings in the time needed to produce the results in the paper.

## 1. The one parameter Weibull distribution

The pdf $f(x)$, cdf $F(x)$, and quantile function $Q(u)$ of the Weibull distribution for $x > 0$, $\alpha > 0$, and $u \in (0, 1)$ are given by

$$f(x) = \alpha x^{\alpha-1} e^{-x^\alpha}, \qquad F(x) = 1 - \exp(-x^\alpha), \qquad Q(u) = -\log(1-u)^{1/\alpha}.$$

The mean $\mu$, variance $\sigma^2$, and median $M = Q(1/2)$ are given by

$$\mu = \Gamma(1 + 1/\alpha), \qquad \sigma^2 = \Gamma(1 + 2/\alpha) - (\Gamma(1 + 1/\alpha))^2, \qquad M = Q(1/2) = g(\alpha) = \log(2)^{1/\alpha}.$$

Graphs of the pdf plotted from $Q(.001)$ to $\min(Q(.999), 3)$ are given in Figure 1 for $\alpha = 1, 2, 3, 4$. These graphs show that $f$ becomes more symmetric as $\alpha$ increases, and that $M$ approaches one although never reaching one.

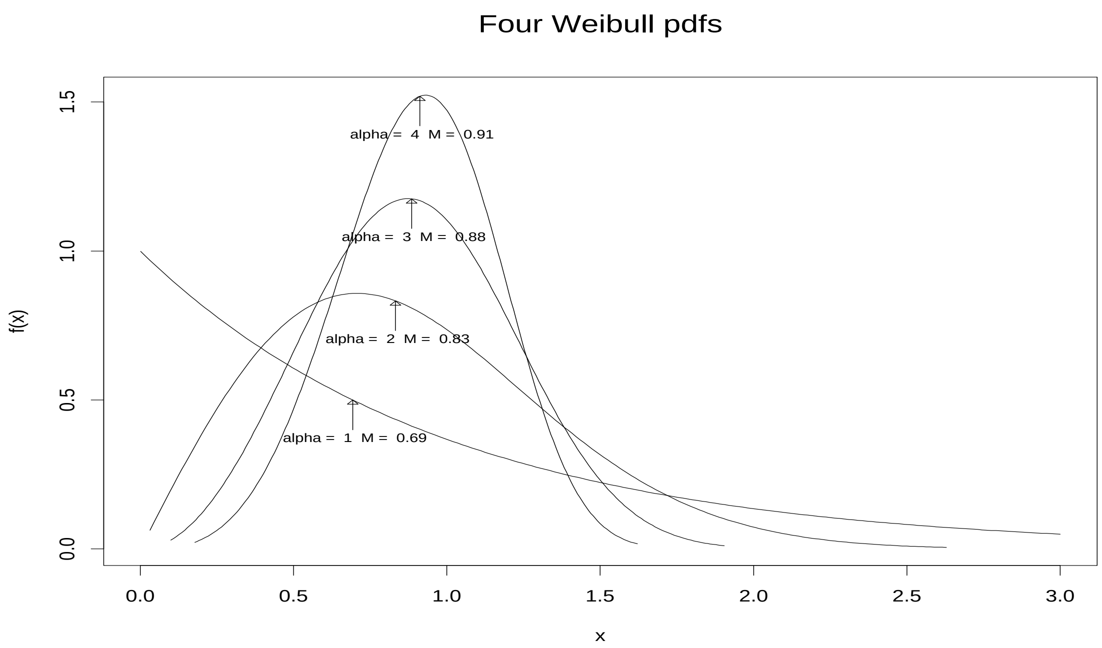


Figure 1. Plot of four Weibull density functions.

## 2. Estimating the Weibull median

In this section we define our two estimators and discuss their analytical properties.

### 2.1. The sample median and maximum likelihood estimator of the median

Given a sample $X_1, \ldots, X_n$, we consider estimating $M$ by the sample median

$$\hat{M} = \frac{X_{([(n+1)/2])} + X_{([(n+2)/2])}}{2},$$

where $X_{(k)}$ is the $k$th smallest element in the sample, and by the maximum likelihood estimator (MLE)

$$\tilde{M} = g(\hat{\alpha}) = \log(2)^{1/\hat{\alpha}},$$

where $\hat{\alpha}$ is the MLE of $\alpha$.

### 2.2. Using Newton's method to get the MLE

In general, the MLE of a parameter $\theta$ of a distribution having density $f$ given data $X_1, \ldots, X_n$ from that distribution is obtained by maximizing the likelihood function

$$L(\theta) = \prod_{i=1}^{n} f(X_i; \theta),$$

or equivalently, by maximizing the log likelihood

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log f(X_i; \theta).$$

Newton's method can be used to find $\hat{\theta}$ by finding the zero of $l'(\theta)$, that is, given a starting value $\theta_0$, we iterate

$$\theta_{i+1} = \theta_i - l'(\theta_i)/l''(\theta_i),$$

until convergence is reached.

Because $\alpha$ must be positive for the Weibull distribution, great care must be used in choosing a starting value $\alpha_0$ and in doing the iteration to ensure that all $\alpha_i$'s are positive. For example, since $M = g(\alpha) = (\log(2))^{1/\alpha}$, it would be tempting to use $\alpha_0 = g^{-1}(\hat{M}) = \log(\log(2))/\log(\hat{M})$ as the starting value. Since $\log(\log 2) = -0.3665129$, for $\alpha_0$ to be positive, we must have $\hat{M} > 1$. Unfortunately, even though $M$ itself is always less than one, it is possible for the median of a sample of size $n$ to be greater than one. For example, if $n = 2k+1$, the chance that $\hat{M} > 1$ is the probability that at least $k+1$ $X$'s in a sample of size $n$ from a Weibull distribution are greater than one. This probability is $P(B \geq k+1)$ where $B$ has the binomial distribution with parameters $n$ and

$$p = P(X > 1) = 1 - F(1) = \exp(-1) = 0.1679,$$

no matter what the value of $\alpha$ is. For $n = 11, 21, 51, 101$ the probabilities that $\hat{M}$ exceed one are 0.181, 0.106, 0.027, and 0.003, respectively.

An alternative starting value can be found as follows. We have $Q(u) = (-\log(1-u))^{1/\alpha}$ which means that $\log(-\log(1-u)) = \alpha \log(Q(u))$, which from our data we can approximate by

$$\log(-\log(1-u_k)) = \alpha X_{(k)}, \qquad k = 1, \ldots, n,$$

where $u_k = (k - .5)/n$. Thus

$$\alpha_0 = \frac{\sum_{k=1}^{n} x_k y_k}{\sum_{k=1}^{n} x_k^2}$$

would be a possible starting value, where $x_k = \log(X_{(k)})$ and $y_k = \log(-\log(1-u_k))$. Notice that it is possible, but unlikely, for this $\alpha_0$ to be negative.

Since there is no guarantee in

$$\alpha_{i+1} = \alpha_i - l'(\alpha_i)/l''(\alpha_i),$$

that the $\alpha_i$'s will remain positive, we will think of $\alpha$ as being $e^\beta$ where $\beta$ can be any real number (this is called a *reparametrization* of the problem). This forces $\alpha$ to be positive. We can get $\hat{\alpha} = e^{\hat{\beta}}$, where $\beta_0 = \log(\alpha_0)$ and

$$\beta_{i+1} = \beta_i - l'(\beta_i)/l''(\beta_i),$$

where $l(\beta)$ means $l(\alpha)$ with $\alpha$ written as $e^\beta$, while $l'(\beta)$ and $l''(\beta)$ mean the first two derivatives of $l(\beta)$ with respect to $\beta$. These three functions are given by

$$\begin{aligned}
l(\beta) &= n\beta + (e^\beta - 1)\sum_{i=1}^n \log x_i - \sum_{i=1}^n x_i^{e^\beta}, \\
l'(\beta) &= n + e^\beta \sum_{i=1}^n \log x_i - e^\beta \sum_{i=1}^n x_i^{e^\beta} \log x_i, \\
l''(\beta) &= e^\beta \sum_{i=1}^n \log x_i - e^\beta \left\{ \sum_{i=1}^n x_i^{e^\beta} \log x_i + e^\beta \sum_{i=1}^n x_i^{e^\beta} (\log x_i)^2 \right\}.
\end{aligned}$$

### 2.3. Comparing the two estimators of the median

The standard method for comparing two estimators $\hat{\theta}_1$ and $\hat{\theta}_2$ of some parameter $\theta$ is to first verify that they are unbiased (at least asymptotically as $n \to \infty$), that is, their average value over many samples is $\theta$, and then to compare $\hat{\theta}_1$ and $\hat{\theta}_2$ in terms of how far they are from $\theta$ on the average for many samples, that is, compare their standard errors

$$\mathrm{se}(\hat{\theta}_i) = \sqrt{\mathrm{Var}(\hat{\theta}_i)} = \sqrt{\mathrm{E}(\hat{\theta}_i - \theta)^2}.$$

In our present case, both $\hat{M}$ and $\tilde{M}$ are asymptotically unbiased. Unfortunately, $\mathrm{se}(\hat{M})$ and $\mathrm{se}(\tilde{M})$ are too difficult to determine analytically. The usual thing to do in a case where $\mathrm{se}(\hat{\theta})$ cannot be evaluated is to estimate it by the root mean square error

$$\mathrm{rmse}(\hat{\theta}) = \sqrt{\frac{1}{r}\sum_{i=1}^r (\hat{\theta}_i - \theta)^2},$$

where $\hat{\theta}_1, \ldots, \hat{\theta}_r$ are the values of the estimator $\hat{\theta}$ for $r$ replications.

To get an idea of $\mathrm{se}(\hat{\theta})$ at least for large sample size $n$, one can usually find analytically a quantity called the asymptotic standard error of $\hat{\theta}$, denoted by $\mathrm{ase}(\hat{\theta})$, satisfying

$$\lim_{n\to\infty} \frac{\mathrm{ase}(\hat{\theta})}{\mathrm{se}(\hat{\theta})} = 1.$$

For a sample quantile $\hat{Q}(p)$ from a sample of size $n$ from a distribution having pdf $f$,

$$\mathrm{ase}(\hat{Q}(p)) = \frac{1}{\sqrt{p(1-p)nf^2(Q(p))}},$$

which gives for the median (the case $p = 1/2$)

$$\mathrm{ase}(\hat{M}) = \frac{1}{\sqrt{4nf^2(M)}}. \tag{1}$$

For an MLE $\hat{\theta}$, $\text{ase}(\hat{\theta}) = \sqrt{\text{avar}(\hat{\theta})}$, where

$$\text{avar}(\hat{\theta}) = \frac{1}{-n\text{E}\left(\frac{\partial^2 \log f(X;\theta)}{\partial \theta^2}\right)}.$$

To get $\text{ase}(\tilde{M})$ we use the general result that

$$\text{avar}(g(\hat{\theta})) = (g'(\theta))^2 \, \text{avar}(\hat{\theta}).$$

### 2.4. Determining asymptotic standard errors of MLE's

In most cases, the most difficult part of finding asymptotic standard errors is finding the expectation in $\text{avar}(\hat{\theta})$. If we estimate this expectation by just the average value of the second derivative of $\log f$, then this is just $l''(\theta)/n$. Thus one way to approximate $\text{ase}(\theta)$ is $1/l''(\theta)$, which itself can be estimated by $1/l''(\hat{\theta})$.

Another way to approximate the expectation is to calculate the integral involved numerically. For example, we can just use the $N$ rectangular sums over $X$ from $Q(.001)$ to $Q(.999)$, that is,

$$\text{E}\left(\frac{\partial^2 \log f(X;\theta)}{\partial \theta^2}\right) = \int \frac{\partial^2 \log f(x;\theta)}{\partial \theta^2} f(x;\theta)dx \doteq h \sum_{i=1}^{N} \frac{\partial^2 \log f(x_i;\theta)}{\partial \theta^2} f(x_i;\theta),$$

where $x_i = Q(.001) + (i-1) * h$, $i = 1, \ldots, N$, with $h = (Q(.999) - Q(.001))/(N-1)$.

### 2.5. Asymptotic standard errors for Weibull median estimators

From (1), we have

$$\text{ase}(\hat{M}) = \frac{1}{\alpha(\log(2))^{(\alpha-1)/\alpha}\sqrt{n}},$$

since $M = (\log(2)^{1/\alpha}$. For $\tilde{M}$ we have

$$g'(\alpha) = (\log 2)^{1/\alpha}(\log(\log 2))(-1/\alpha^2),$$

and

$$\begin{aligned} -E\left(\frac{\partial^2 \log f(X;\alpha)}{\partial \alpha^2}\right) &= \int_0^\infty \left[\frac{1}{\alpha^2} + x^\alpha (\log x)^2\right] \alpha x^{\alpha-1} e^{-x^\alpha} dx \\ &= \frac{1}{\alpha^2}\left[1 + \int_0^\infty t(\log t)^2 e^{-t} dt\right] && (2) \\ &= \frac{1}{\alpha^2}\left[1 + \Gamma''(2)\right] && (3) \\ &= \frac{1}{\alpha^2}\left[1 + (-2\gamma + \gamma^2 + \pi^2/6)\right] && (4) \\ &= \frac{1}{\alpha^2}\left[(\gamma-1)^2 + \pi^2/6\right]. \end{aligned}$$

where to get (2) we used the substitution $t = x^\alpha$, to obtain (3) we used

$$\Gamma^{(n)}(x) = \frac{d^n}{dx^n}\Gamma(x) = \frac{d^n}{dx^n}\int_0^\infty t^{x-1}e^{-t}dt = \int_0^\infty \frac{d^n}{dx^n} t^{x-1}e^{-t}dt = \int_0^\infty t^{x-1}(\log t)^n e^{-t}dt,$$

and for (4), we will show in Section 2.6 that

$$\Gamma''(2) = -2\gamma + \gamma^2 + \pi^2/6, \qquad (5)$$

where

$$\gamma = \lim_{n\to\infty}\left[\sum_{k=1}^{n}\frac{1}{k} - \log n\right] \doteq 0.57721$$

is called Euler's constant.

Thus we have

$$\text{ase}(\tilde{M}) = -\frac{(\log 2)^{1/\alpha}\log(\log 2)}{\alpha\sqrt{n}\sqrt{(\gamma-1)^2+\pi^2/6}}.$$

Finally, in terms of asymptotic standard errors, we have

$$\frac{\text{ase}(\hat{M})}{\text{ase}(\tilde{M})} = -\frac{\sqrt{(\gamma-1)^2+\pi^2/6}}{\log 2\log(\log 2)} \doteq 5.316,$$

that is, the asymptotic statndard error of the sample median is 5.316 times larger than that of the MLE of the median, independent of the value of $\alpha$.

### 2.6. Derivation of equation (5)

Since $\Gamma(x+1) = x\Gamma(x)$, we have $\Gamma''(x+1) = x\Gamma''(x) + 2\Gamma(x)$, and so $\Gamma''(2) = \Gamma''(1) + 2\Gamma'(1)$. We will show $\Gamma'(1) = -\gamma$ and $\Gamma''(1) = \gamma^2 + \pi^2/6$, from which our desired result follows.

If we define

$$\psi(x) = \frac{d}{dx}\log\Gamma(x) = \Gamma'(x)/\Gamma(x),$$

we have $\psi(1) = \Gamma'(1)$ since $\Gamma(1) = 1$. We will show $\psi(1) = -\gamma$. Also, differentiating $\psi(x) = \Gamma'(x)/\Gamma(x)$ gives $\Gamma''(1) = \psi^2(1) + \psi'(1)$, so we must also show $\psi'(1) = \pi^2/6$. We will in fact show that

$$\psi(x) = -\gamma - \frac{1}{x} + \sum_{n=1}^{\infty}\left(\frac{1}{n} - \frac{1}{x+n}\right), \tag{6}$$

from which $\psi(1) = -\gamma$ (because terms in successive elements in the series cancel), and differentiating (6) term by term gives

$$\psi'(x) = \frac{1}{x^2} + \sum_{n=1}^{\infty}\frac{1}{(x+n)^2},$$

which gives

$$\psi'(1) = \sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{\pi^2}{6}.$$

To show (6), we have

$$\psi(x+1) = \frac{\Gamma'(x+1)}{\Gamma(x+1)} = \frac{\Gamma(x) + x\Gamma'(x)}{x\Gamma(x)} = \frac{1}{x} + \psi(x),$$

which successive substitution gives for a positive integer $n$

$$\psi(x+n) = \psi(x) + \sum_{k=0}^{n-1}\frac{1}{x+k}.$$

Thus from this, the definition of $\gamma$, and the fact that $\psi(x+n) - \log n \to 0$ (see Abramowitz and Stegun (1974), *Handbook of Mathematical Functions*, Dover, eq. 6.3.18)

$$\begin{aligned} 0 &= \lim_{n\to\infty}[\psi(x+n) - \log n] = \lim_{n\to\infty}\left[\psi(x) + \sum_{k=0}^{n-1}\frac{1}{x+k} - \left(\sum_{k=1}^{n}\frac{1}{k} - \gamma\right)\right] \\ &= \psi(x) + \gamma + \frac{1}{x} - \lim_{n\to\infty}\sum_{k=1}^{n-1}\left(\frac{1}{k} - \frac{1}{x+k}\right) \end{aligned}$$

which gives

$$\psi(x) = -\gamma - \frac{1}{x} + \sum_{k=1}^{\infty} \left( \frac{1}{k} - \frac{1}{x+k} \right),$$

as required, thus completing the derivation of (6).

## 3. A simulation study

For each $\alpha = 1, 2, 3, 4$ and each $n = 10, 20, 50, 100$ we

- Generated 1000 samples of size $n$ from a Weibull distribution having parameter $\alpha$ and calculated $\hat{M}$ and $\tilde{M}$ for each sample,
- Calculated the average and root mean squared errors of the $\hat{M}$'s and $\tilde{M}$'s.
- Found kernel density estimates from the 1000 $\hat{M}$'s and 1000 $\tilde{M}$'s. The bandwidth for each density estimate was determined as the smallest width (starting with the range of the 1000 values divided by 20) giving a smooth, unimodal estimate.

## 4. Results

In Table 1 we present the numerical results of the simulation. Notice that the pairs of entries next to mean, ase, and rmse are for $\hat{M}$ and $\tilde{M}$, respectively.

**Table 1. The numerical results of the simulation study**

| | | Values for $\alpha$ | | | |
|---|---|---|---|---|---|
| Sample Size | | 1 | 2 | 3 | 4 |
| 10 | true median | 0.6931 | 0.8326 | 0.8850 | 0.9124 |
| | mean | 0.7568—0.6993 | 0.8402—0.8364 | 0.8881—0.8881 | 0.9097—0.9145 |
| | ase | 0.3162—0.0600 | 0.1899—0.0360 | 0.1345—0.0255 | 0.1040—0.0197 |
| | rmse | 0.3326—0.0604 | 0.1808—0.0348 | 0.1278—0.0260 | 0.0958—0.0201 |
| 20 | true median | 0.6931 | 0.8326 | 0.8850 | 0.9124 |
| | mean | 0.7188—0.6964 | 0.8348—0.8351 | 0.8889—0.8863 | 0.9124—0.9133 |
| | ase | 0.2236—0.0424 | 0.1342—0.0255 | 0.0951—0.0180 | 0.0735—0.0139 |
| | rmse | 0.2133—0.0423 | 0.1263—0.0255 | 0.0928—0.0185 | 0.0743—0.0144 |
| 50 | true median | 0.6931 | 0.8326 | 0.8850 | 0.9124 |
| | mean | 0.7089—0.6966 | 0.8300—0.8334 | 0.8850—0.8863 | 0.9147—0.9127 |
| | ase | 0.1412—0.0268 | 0.0849—0.0161 | 0.0601—0.0114 | 0.0465—0.0088 |
| | rmse | 0.1381—0.0269 | 0.0832—0.0155 | 0.0590—0.0118 | 0.0452—0.0086 |
| 100 | true median | 0.6931 | 0.8326 | 0.8850 | 0.9124 |
| | mean | 0.6939—0.6941 | 0.8336—0.8325 | 0.8861—0.8856 | 0.9113—0.9130 |
| | ase | 0.1000—0.0189 | 0.0600—0.0114 | 0.0425—0.0080 | 0.0329—0.0062 |
| | rmse | 0.1694—0.1396 | 0.0789—0.0535 | 0.0495—0.0279 | 0.0334—0.0065 |

In Figure 2 we give the 16 pairs of density estimates for the different combinations of $\alpha$ and $n$.

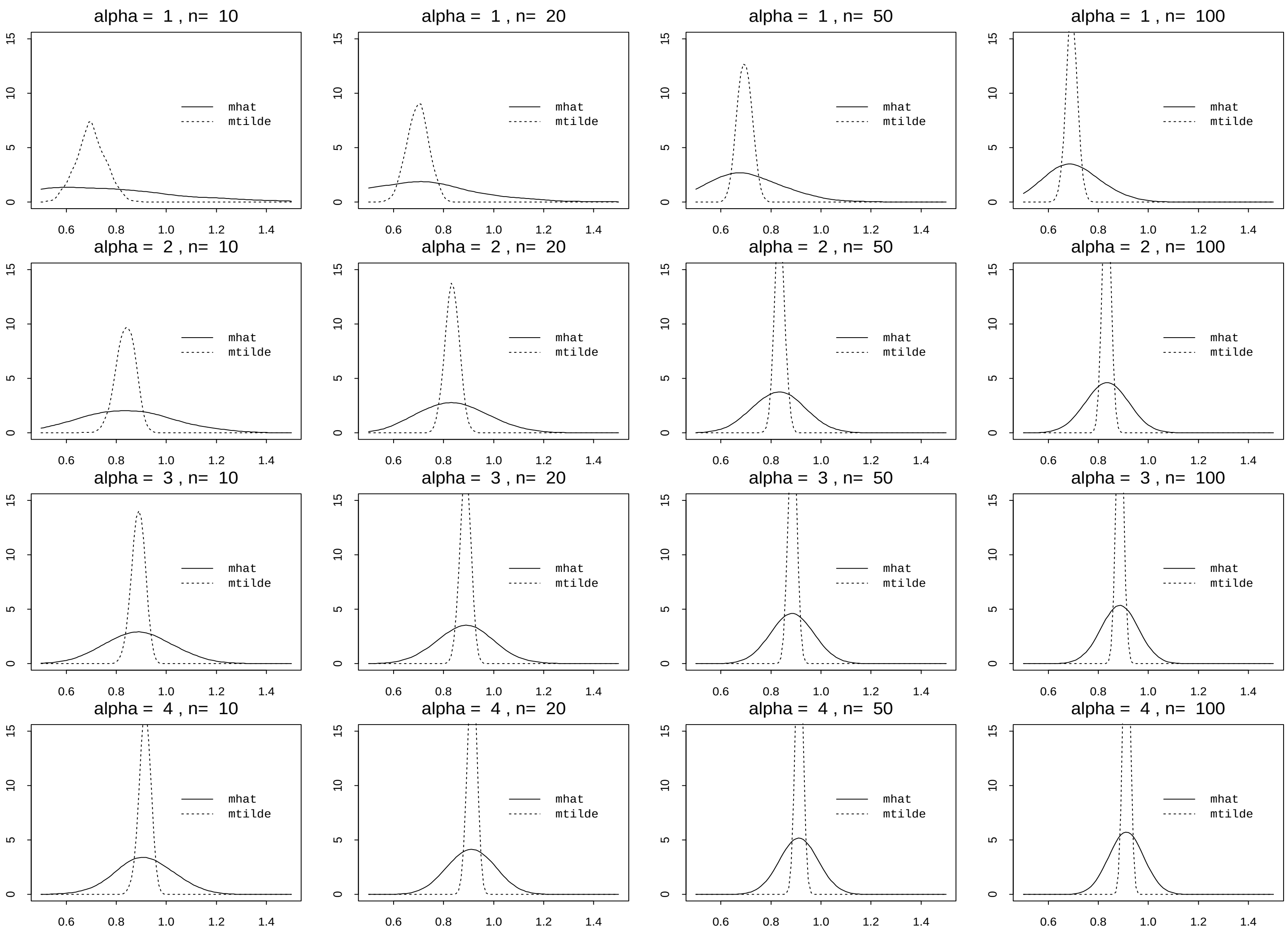


**Figure 2.** Kernel density estimates for the 1000 $\hat{M}$'s and $\tilde{M}$'s.

## 5. Summary

Both the analytical and simulation results show overwhelmingly that $\tilde{M}$ is a much better estimator of $M$ than $\hat{M}$. While both are approximately unbiased, the asymptotic standard error for $\hat{M}$ is more than five times larger than that of $\tilde{M}$. Further, the close agreement of the rmse's to the ase's, even for moderate sample sizes, show that the ase's are valid measures of variability in this example. Finally, except for small $n$, the density estimates suggest that both estimators are well approximated by a normal distribution. One further topic we might study would be to look at the bias in the two estimators for small samples.

In terms of timing, it took less than 30 seconds on one of our fastest Sun workstations to obtain the results of the simulation using the Splus function `tableweib` that calls the Fortran subroutine `m2weib` for each combination of $\alpha$ and $n$. The Splus finction `slowloops` described below can be used to find the 1000 $\hat{M}$'s and $\tilde{M}$'s for all 16 combinations of $\alpha$ and $n$. To do so took almost 30 minutes. Thus this small use of Fortran decreased the time for the simulation by a factor of approximately 60.

## 6. Computer code

The results obtained in this report were obtained using Splus with the exception of a Fortran subroutine called from Splus to obtain the 1000 $\hat{M}$'s and $\tilde{M}$'s in the simulation study in Section 3.

### 6.1. The Fortran subroutine

Here is the Fortran subroutine which receives an $n \times 1000$ matrix of Weibull samples from Splus and returns the $\hat{M}$'s and $\tilde{M}$'s. The Splus function `m2weib` is the interface routine for this subroutine.

```
      subroutine hw6(x,nreps,n,ndim,mhat,mtilde,ier)
c***********************************************************************
c
c  Subroutine take a matrix x whose nreps columns are each a random sample
c  from a one-parameter Weibull of size n and return two arrays of length reps:
c
c  mhat:   the sample medians
c  mtilde: the MLE's of the median
c
c  If the MLE fails to converge for any sample, hw6 returns ier=1
c  Otherwise, ier=0
c
c  ndim is the row dimension of x in the calling routine
c
c  subprograms called : qsort (in 6041.f)
c
c***********************************************************************

      double precision x(ndim,nreps),mhat(nreps),mtilde(nreps)
      double precision a,la,aold,s1,s2,s3,lx,xa,l1,l2,del,eps
      double precision bold,uk,xk,yk,sxx,sxy

      integer n,nreps,ndim,i,j,k,ier,maxit

      data eps,maxit/1.e-3,100/

      ier=0

      a=log(2.)
      la=log(a)

c  loop over reps:

      do 100 i=1,nreps

c  get sample median:

         call qsort(x(1,i),n)
         mhat(i)=(x((n+1)/2,i)+x((n+2)/2,i))/2.

c  starting value for Newton's method:

         sxx=0.0
         sxy=0.0

```

```
         do 101 k=1,n
            uk=(k-.5)/n
            yk=log(-log(1-uk))
            xk=log(x(k,i))
            sxx=sxx+xk*xk
            sxy=sxy+xk*yk
 101      continue

         aold=sxy/sxx
         bold=log(aold)

c  Newton's method iteration:

         do 20 j=1,maxit

            s1=0.0
            s2=0.0
            s3=0.0

            do 10 k=1,n
               lx=log(x(k,i))
               xa=exp(aold*lx)
               s1=s1+lx
               s2=s2+xa*lx
               s3=s3+xa*lx*lx
 10         continue

            l1=n+aold*s1-aold*s2
            l2=aold*s1-aold*s2-aold*aold*s3
            del=l1/l2

c  if converge, go to next reps loop, else next iteration:

            if((abs(del).lt.eps*abs(bold)).or.(abs(l1).lt.eps)) then
               mtilde(i)=a**(1/aold)
               go to 100
            else
               bold=bold-del
               aold=exp(bold)
            endif

 20       continue

c  getting to here means nonconvergence:

          mtilde(i)=del
          ier=1
          return

 100  continue

      return
      end
```

### 6.2. The Splus functions

The file listed in the next section contains all of the Splus code used in this paper:

1. The file starts with functions `dweib`, `pweib`, `qweib`, and `rweib` which find the density, cdf, and quantile functions of the Weibull and generates random numbers.

2. Next come functions `medweib` (calculate the median of a Weibull distribution), `asemhat` (get the ase of $\hat{M}$), and `asemtilde` and `asemtilde1` which calculate the ase of $\tilde{M}$ by the rectangular sum approximation and by the exact formula.

3. The function `rmse` calculates the root mean squared error of an estimator while `dload` is a simple function that does the dynamic loading of the Fortran code.

4. The function `plot4weib` was used to obtain the Weibull pdf plots in Figure 1.

5. The function `tableweib` was used to get the results in Table 1.

6. The function m2weib is the Splus interface to the Fortran subroutine used to calculate $\hat{M}$'s and $\tilde{M}$'s.

7. The function densweib was used to obtain the kernel density estimate plots in Figure 2.

8. The file ends with the functions mhatweib and slowloops, neither of which were used in the paper, but are included to illustrate some interesting ideas. The function mhatweib shows how one can obtain the sample medians of a collection of random samples without using a loop, while slowloops is included in case someone wants to see how long the simulation would take without using Fortran.

### 6.3. The Splus code

```
dweib   <- function(x,alpha) { return(alpha*x^(alpha-1)*exp(-x^alpha)) }

pweib   <- function(x,alpha) { return(1-exp(-x^alpha)) }

qweib   <- function(u,alpha) { return((-log(1-u))^(1/alpha)) }

rweib   <- function(n,alpha) { return(qweib(runif(n),alpha)) }

medweib <- function(alpha)   { return((log(2))^(1/alpha)) }

asemhat <- function(n,alpha)     # ase for sample median from Weibull(alpha)
{
  return(1/(alpha*(log(2))^((alpha-1)/alpha)*sqrt(n)))
}


asemtilde <- function(n,alpha)   # ase for median MLE by rectangular sums
{
    N <- 10000
    x <- seq(qweib(.0001,alpha),qweib(.9999,alpha),length=N)
    h <- x[2]-x[1]

    avar <- 1/((n/alpha^2) +
               n*alpha*h*sum(x^(2*alpha-1)*(log(x)^2)*exp(-x^alpha)))

    return(sqrt(avar)*abs((log(2))^(1/alpha)*log(log(2))/(alpha^2)))
  }

asemtilde1 <- function(n,alpha)  # ase for sample median by final formula
{
  return(-(log(2)^(1/alpha)*log(log(2)))/(alpha*sqrt(n)*sqrt((.577216-1)^2+
             pi^2/6)))
}

dload <- function() # dynamically load stat 604 library and hw6.o
{
  dyn.open(c('604l.o','hw6.o'))
  invisible()
}

rmse <- function(thetahat,theta) # get RMSE for a vector of estimates
{
  z <- thetahat-theta
  return(sqrt(sum(z*z)/length(z)))
}

plot4weib <- function()
#------------------------------------------------------------
#
#  Plot Weibull densities for alpha=1,2,3,4 from Q(.001) to min(Q(.999,3)
#  and put alpha and median on the plot
#
#------------------------------------------------------------
{

    nl <- 201
    x  <- fx <- matrix(0,nl,4)

    for(i in 1:4) {

      x[,i]  <- seq(qweib(.001,i),min(qweib(.999,i),3),length=nl)
      fx[,i] <- dweib(x[,i],i)

```

```
    }

    plot(c(min(x),max(x)),c(min(fx),max(fx)),type='n',xlab='x',ylab='f(x)',
         main='Four Weibull pdfs')

    for(i in 1:4) {

      lines(x[,i],fx[,i])
      med  <- medweib(i)
      fmed <- dweib(med,i)
      arrows(med,fmed-.1,med,fmed,size=.1)
      text(med,fmed-.125,paste(' alpha = ',i,' M = ',round(med,2)),cex=.75)

    }

}

tableweib <- function(seed=12345)
#-------------------------------------------------------------
#
#  For alpha=1,2,3,4 and n=10,20,50,100:
#
#    1. Find true median (via mweib) and ase's for sample median
#       and MLE of median (via asemhat and asemtilde)
#
#    2. Generate 1000 samples of size n from Weibull(alpha)
#
#    3. Call Fortran (via m2weib) to get the 1000 sample medians and MLE's
#
#    4. Find means and rmse's for the medians and MLE's
#
#  Then put tables of these quantities in the file tableweib.out and
#  return the medians and MLE's in a list having members mhat and mtilde
#
#-------------------------------------------------------------
{

    set.seed(seed)
    alpha <- 1:4
    n <- c(10,20,50,100)

    med <- rep(0,4)
    ier <- ase1 <- ase2 <- m1bar <- m2bar <- rmse1 <- rmse2 <- matrix(0,4,4)

    mhatmat <- mtildemat <- matrix(0,1000,16)

    jj <- 0

    for(i in 1:4) {

        for(j in 1:4) {

        jj <- jj+1

        zz <- m2weib(matrix(rweib(1000*n[j],alpha[i]),n[j],1000))
        mhatmat[,jj]  <- zz$mhat
        mtildemat[,jj] <- zz$mtilde

        print(zz$ier)

        med[i] <- medweib(alpha[i])

        ier[i,j] <- zz$ier
        ase1[i,j] <- asemhat(n[j],alpha[i])
        ase2[i,j] <- asemtilde(n[j],alpha[i])

        m1bar[i,j] <- mean(zz$mhat)
        m2bar[i,j] <- mean(zz$mtilde)

        print(m1bar[i,j])

        rmse1[i,j] <- rmse(zz$mhat,med[i])
        rmse2[i,j] <- rmse(zz$mtilde,med[i])

      }

    }

    sink('tableweib.out')
    for(i in 1:4) {
```

```
      print(round(med,4))
      print(round(ier[,i],4))
      print(round(ase1[,i],4))
      print(round(ase2[,i],4))
      print(round(m1bar[,i],4))
      print(round(m2bar[,i],4))
      print(round(rmse1[,i],4))
      print(round(rmse2[,i],4))

    }

    sink()

    return(list(mhat=mhatmat,mtilde=mtildemat))

}

m2weib <- function(x)
#------------------------------------------------------------
#
#  Call Fortran to get sample medians and MLE's of median from
#  nreps samples of size n from Weibull
#
#------------------------------------------------------------
  {

    n     <- dim(x)[1]
    nreps <- dim(x)[2]

    zz <- .Fortran("hw6",
                   as.double(x),
                   as.integer(nreps),
                   as.integer(n),
                   as.integer(n),
              mhat=as.double(rep(0,nreps)),
            mtilde=as.double(rep(0,nreps)),
               ier=as.integer(0))

    return(list(mhat=zz$mhat,mtilde=zz$mtilde,ier=zz$ier))

  }


densweib <- function(zz)
#------------------------------------------------------------
#
#  Superimpose pairs of plots of density estimates from 1000 sample medians
#  and MLE's of medians for all combinations of alpha=1,2,3,4 and
#  n=10,20,50,100
#
#------------------------------------------------------------
  {

    par(mfrow=c(4,4),mar=c(2,2,2,2))

    alpha <- c(1,2,3,4)
    nn <- c(10,20,50,100)

    jj <- 0
    for(i in 1:4) {
      for(j in 1:4) {
        jj <- jj+1
        plot(c(.5,1.5),c(0,15),type='n',main=paste('alpha = ',alpha[i],
             ', n= ',nn[j]))

        lines(density(zz$mhat[,jj],n=101,from=.5,to=1.5,width=.25),lty=1)
        lines(density(zz$mtilde[,jj],n=101,from=.5,to=1.5,width=.05),lty=2)
        legend(1.0,10,c('mhat','mtilde'),bty='n',lty=1:2,font=2)

      }

    }

    invisible()

  }


mhatweib <- function(nreps,n,alpha)  # get sample medians for reps samples
```

```
{
  w  <- matrix(rweib(nreps*n,alpha),n,nreps)
  w  <- matrix(sort(w+100*col(w)),n,nreps)-100*col(w)
  i1 <- floor((n+1)/2)
  i2 <- floor((n+2)/2)
  return((w[i1,]+w[i2,])/2)
}



slowloops <- function(reps)
#------------------------------------------------------------
#
#  This routine does the simulation without using Fortran.
#  It is included so that one can time how slow doing the
#  project would be without using Fortran
#
#------------------------------------------------------------
{

  print(unix('date'))
  alpha <- 1:4
  n     <- c(10,20,50,100)

  mhat  <- mtilde <- matrix(0,reps,16)

  jj <- 0
  for(i in 1:4) {
    for(j in 1:4) {

      print(i)
      print(j)

      jj <- jj+1
      xx <- matrix(rweib(reps*n[j],alpha[i]),n[j],reps)

      for(k in 1:reps) {

        mhat[k,jj]   <- median(xx[,k])

        zz <- weibmle(xx[,k])
        if(zz$ier==1) print(k)

        mtilde[k,jj] <- zz$mtilde

      }

    }

  }

  print(unix('date'))
  return(list(mhat=mhat,mtilde=mtilde))

}

weibmle <- function(x, maxit=100, eps=.0001)
#----------------------------------------------
{

  n <- length(x)

  yk <- log(-log(1-((1:n)-.5)/n))
  xk <- log(sort(x))

  aold <- max(.5,sum(xk*yk)/sum(xk*xk))
  bold <- log(aold)

  lx <- log(x)
  s1 <- sum(lx)

  for(l in 1:maxit) {

    if(sum(is.na(aold*lx))!=0) print(lx)

    xa <- exp(aold*lx)
    s2 <- sum(xa*lx)
    s3 <- sum(xa*lx*lx)

    del <- (n+aold*(s1-s2))/(aold*(s1-s2-aold*s3))
```

```
    if(abs(del) < eps*abs(bold)) { return(list(mtilde=log(2)^(1/aold),ier=0)) }

    bold <- bold-del
    aold <- exp(bold)

  }


  return(list(mtilde=log(2)^(1/aold),ier=1))

}
```