Maximum Likelihood for Censored Data with R*

```
> rm(list=ls()); options(scipen=999)
> wdata = read.table("http://www.utstat.utoronto.ca/~brunner/data/legal/Weibull.data2.txt")
> head(wdata)
  Time Uncensored
1 1.60          0
2 0.60          0
3 3.03          1
4 2.90          0
5 3.60          1
6 2.76          1
> summary(wdata)
      Time          Uncensored    
 Min.   :0.010   Min.   :0.0000  
 1st Qu.:1.680   1st Qu.:0.0000  
 Median :3.200   Median :1.0000  
 Mean   :3.257   Mean   :0.5236  
 3rd Qu.:4.675   3rd Qu.:1.0000  
 Max.   :8.230   Max.   :1.0000  
> attach(wdata) # Now Time (failure time) and Uncensored (delta) are available by name.
> 
> # Find MLE numerically
```

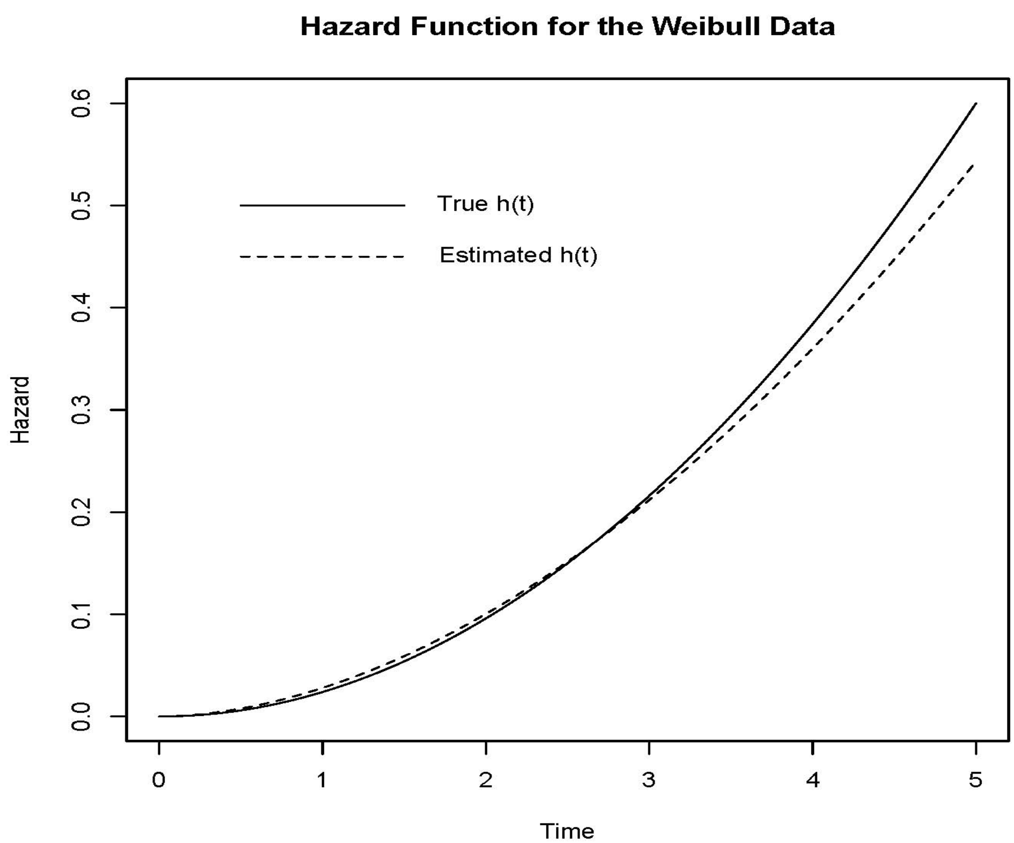
$$f(t|\alpha, \lambda) = \alpha\lambda(\lambda t)^{\alpha-1} \exp\{-(\lambda t)^{\alpha}\}$$

```
> 
> mloglike = function(theta,t,delta)
+     { # Minus log likelihood function
+     alpha = theta[1]; lambda = theta[2]
+     # logf and logS will be of length n
+     logf = log(alpha)+log(lambda)+(alpha-1)*log(lambda*t) + -(lambda*t)^alpha
+     logS = -(lambda*t)^alpha
+     value = -sum(logf*delta) - sum(logS*(1-delta))
+     return(value)
+     } # End of function mloglike
> 
> # Testing
> mloglike(c(3,0.2),t=Time,delta=Uncensored)
[1] 321.3091
> 
> yes = Time[Uncensored==1]; no = Time[Uncensored==0]
> -sum(dweibull(yes,shape=3, scale=5,log=TRUE)) - sum(pweibull(no, shape=3, scale=5, 
lower.tail = FALSE, log.p =TRUE))
[1] 321.3091
> 
> 
> 
> ###############################################################################
> # Find MLE
> ###############################################################################
> 
> startvals = c(1,1/2) # I tried a few values
> 
> search1 = optim(par=startvals, fn=mloglike, t=Time,delta=Uncensored,
+                 hessian=TRUE, lower=c(0,0), method='L-BFGS-B')
```

* This document is free and open source. See last page for copyright information.

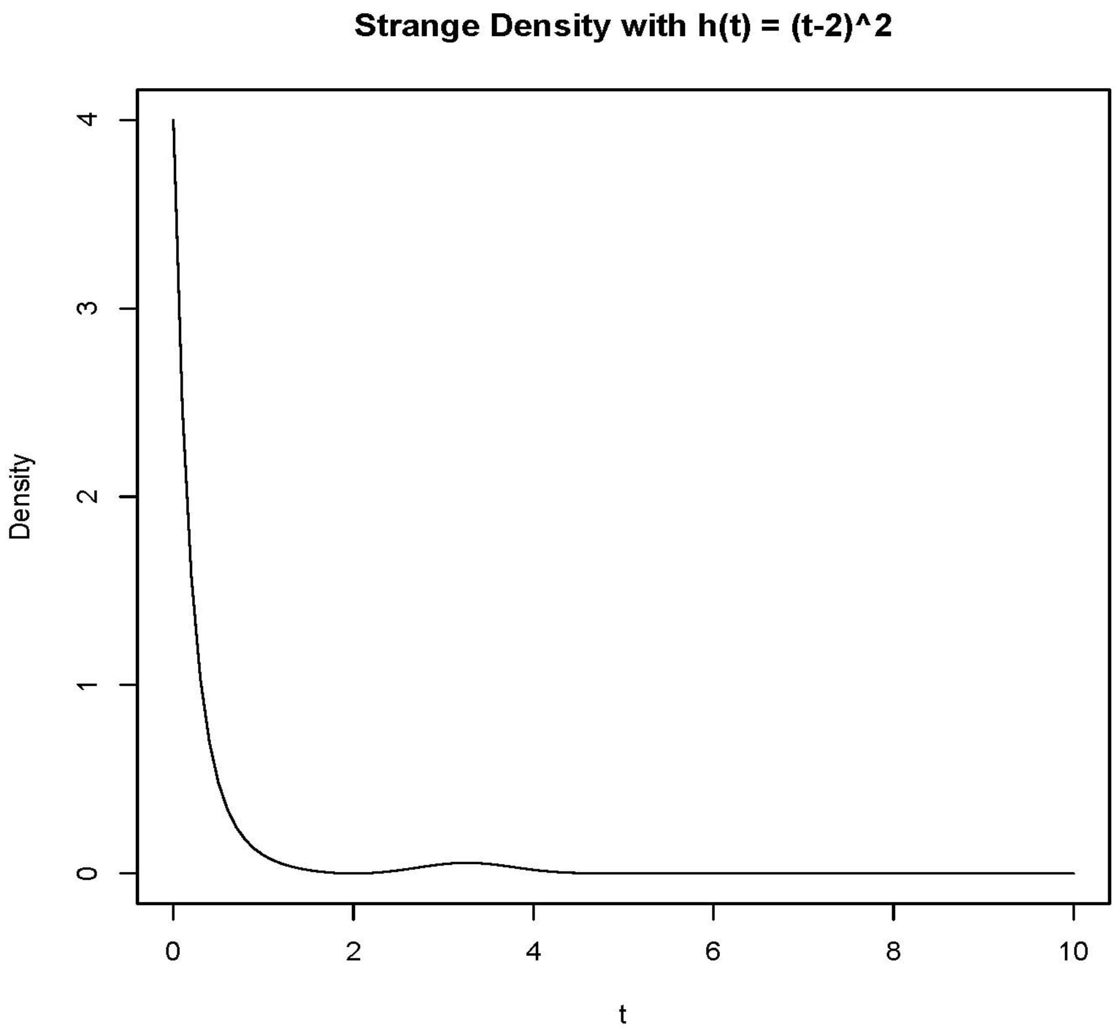
```
> search1
$par
[1] 2.8417618 0.1968182

$value
[1] 320.7533

$counts
function gradient
      14       14

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

$hessian
         [,1]        [,2]
[1,] 30.99060     1.94865
[2,]  1.94865 30020.31664

>
> # If both eigenvalues of the Hessian are positive, minus LL is concave up.
> H = search1$hessian
> eigen(H)$values
[1] 30020.31677    30.99047
>
> alphahat = search1$par[1]; lambdahat = search1$par[2]
>
> # These data were simulated, so I know the true parameter values.
> truealpha = 3; truelambda = 1/5
> # Compare:
> c(alphahat,truealpha)
[1] 2.841762 3.000000
> c(lambdahat,truelambda)
[1] 0.1968182 0.2000000

>
> ################################################################################
> # Calculate the estimated asymptotic covariance matrix of the MLEs.
> ################################################################################
>
> Vhat = solve(H); Vhat # Solve returns the inverse.
                [,1]            [,2]
[1,]  0.032267982000 -0.000002094548
[2,] -0.000002094548  0.000033310911
>
> ################################################################################
> # Point estimate and confidence interval for the median
> # Median = log(2)^(1/alpha) / lambda
> ################################################################################
>
> # Point estimate of median
> medhat = 1/lambdahat * log(2)^(1/alphahat); medhat
[1] 4.466034
> # Compare the truth
> truemedian = log(2)^(1/truealpha) / truelambda; truemedian
[1] 4.424985
>
> median(Time) # Sample median is way off, because it ignores censoring
[1] 3.2
>
```

```
> # Confidence interval for median
> # Need gdot
> # D[b^(1/a),a] works in Wolfram Alpha, as a check on hand calculation.
>
> gdot = cbind( - log(2)^(1/alphahat)*log(log(2))/(lambdahat*alphahat^2),
+               - log(2)^(1/alphahat)/lambdahat^2 )
> v_medhat = as.numeric( gdot %*% Vhat %*% t(gdot) ); se_medhat = sqrt(v_medhat)
> lower95 = medhat - 1.96*se_medhat; upper95 = medhat + 1.96*se_medhat
> c(lower95,upper95)
[1] 4.199471 4.732597
>
> ##############################################################################
> # Plot hazard function h(t) = alpha*lambda^alpha * t^(alpha-1)
> ##############################################################################
>
> x = seq(from=0,to=5,length=101)
> esthazard = alphahat*lambdahat^alphahat * x^(alphahat-1)
> truehazard = truealpha*truelambda^truealpha * x^(truealpha-1)
>
> plot(x,truehazard,type='l',xlab='Time',ylab='Hazard',
+ main='Hazard Function for the Weibull Data')
> lines(x,esthazard,lty=2)
> # Annotate the plot (Make the legend)
> x1 = c(0.5,1.5); y1 = c(0.5,0.5)
> lines(x1,y1,lty=1)
> text(2,0.5,'True h(t)')
> x2 = c(0.5,1.5); y2 = c(0.45,0.45)
> lines(x2,y2,lty=2)
> text(2.2,0.45,'Estimated h(t)')
```



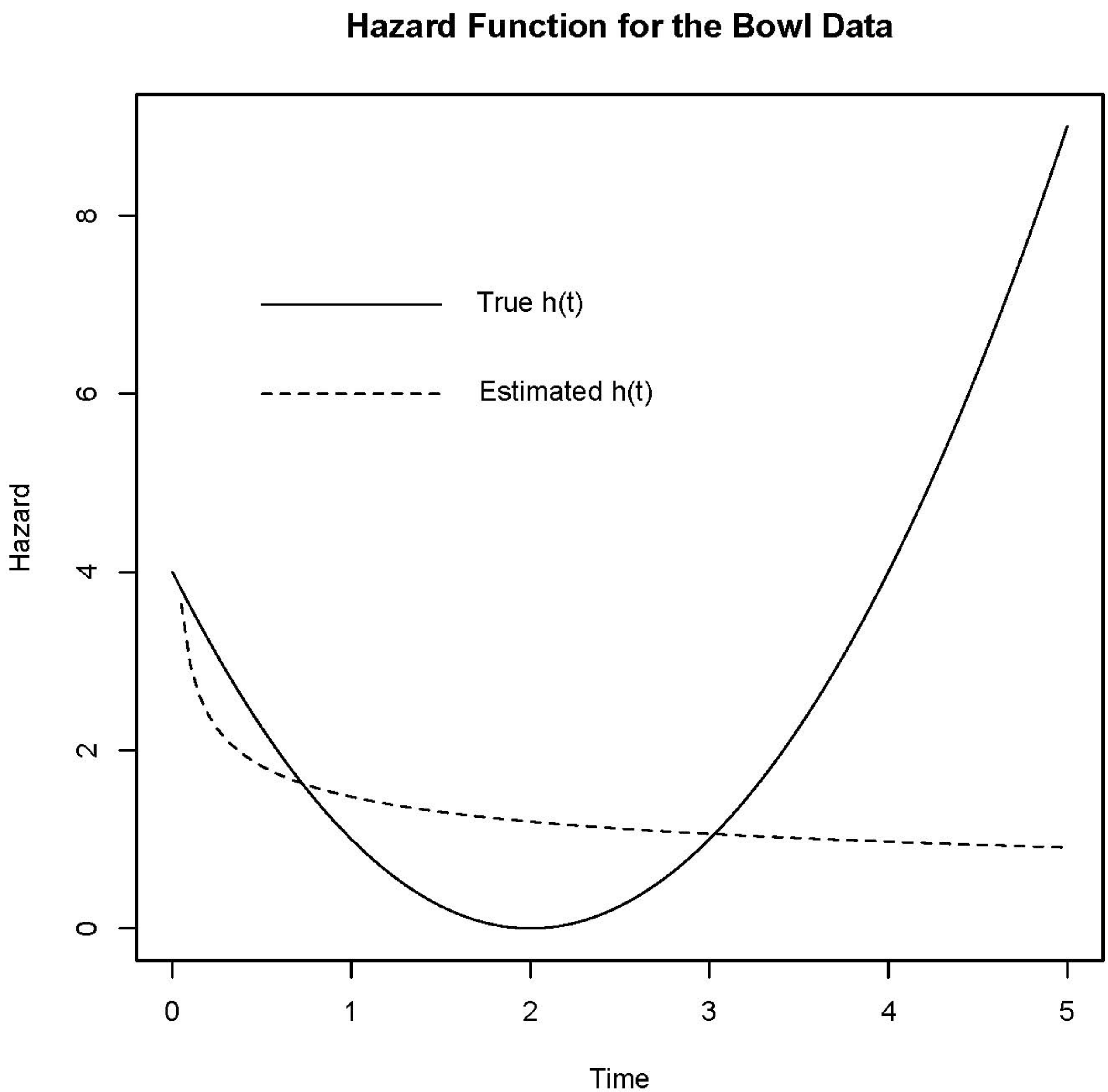
```
> ###############################################################################
> # Okay, that was nice. Try Gumbel model on log transformed data.
> ###############################################################################
>
> gmll = function(theta,y,delta)
+     { # Gumbel minus log likelihood
+     mu = theta[1]; sigma = theta[2]
+     z = (y-mu)/sigma
+     # logf and logS will be of length n
+     logf = -log(sigma) + z - exp(z)
+     logS = -exp(z)
+     value = -sum(logf*delta) - sum(logS*(1-delta))
+     return(value)
+     } # End of function gmll
>
> logt = log(Time)
> summary(logt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-4.6050  0.5188  1.1630  0.8815  1.5420  2.1080
> c(mean(logt),sd(logt))
[1] 0.8814772 1.0297158
>
> gmll(c(1.6,1/3),y=logt,delta=Uncensored)
[1] 138.0478
>
> begin = c( mean(logt),sd(logt) )
> search2 = optim(par=begin, fn=gmll, y=logt,delta=Uncensored,
+                 hessian=TRUE, lower=c(-Inf,0), method='L-BFGS-B')
> search2
$par
[1] 1.6254767 0.3518959

$value
[1] 137.2245

$counts
function gradient
      12       12

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

$hessian
            [,1]        [,2]
[1,] 1162.876296    3.057568
[2,]    3.057568 2021.146865

>
> muhat = search2$par[1]; sigmahat = search2$par[2]
> c(muhat,sigmahat)
[1] 1.6254767 0.3518959
>
> truemu = -log(truelambda); truesigma = 1/truealpha
> c(truemu,truesigma)
[1] 1.6094379 0.3333333
>
> # The invariance principle says the MLE of a function is that function of the MLE.
> c(-log(lambdahat), 1/alphahat) # Compare muhat and sigmahat
[1] 1.6254746 0.3518944
```

```
> # What if the (Weibull) model is wrong?
> t = seq(from=0,to=10,length=101)
> Density = exp(-8/3)* (t-2)^2 * exp(-(t-2)^3/3)
> plot(t,Density,type='l',main='Strange Density with h(t) = (t-2)^2')
```




```
>
> bowldat =
read.table("http://www.utstat.toronto.edu/~brunner/data/legal/bowlhaz.data.txt")
> head(bowldat); summary(bowldat); attach(bowldat)
        Time Uncensored
1 0.72275893          1
2 0.12004774          0
3 0.53171197          1
4 0.05997346          1
5 0.35008144          1
6 0.65936362          1
      Time             Uncensored
 Min.   :0.000144   Min.   :0.000
 1st Qu.:0.062032   1st Qu.:1.000
 Median :0.158124   Median :1.000
 Mean   :0.352768   Mean   :0.862
 3rd Qu.:0.376218   3rd Qu.:1.000
 Max.   :3.722165   Max.   :1.000
```

```
> # Weibull minus log likelihood again
> mloglike = function(theta,t,delta)
+     { # Minus log likelihood function for Weibull
+     alpha = theta[1]; lambda = theta[2]
+     # logf and logS will be of length n
+     logf = log(alpha)+log(lambda)+(alpha-1)*log(lambda*t) + -(lambda*t)^alpha
+     logS = -(lambda*t)^alpha
+     value = -sum(logf*delta) - sum(logS*(1-delta))
+     return(value)
+     } # End of function mloglike
>
> ################################################################################
> # Find MLE
> ################################################################################
>
> startvals = c(1,1/2)
> search = optim(par=startvals, fn=mloglike, t=Time,delta=Uncensored,
+                     hessian=TRUE, lower=c(0,0), method='L-BFGS-B')
> search
$par
[1] 0.699065 2.914821

$value
[1] -13.33825

$counts
function gradient
      13       13

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

$hessian
           [,1]     [,2]
[1,] 1586.58497 38.24125
[2,]   38.24125 24.79069

>
> alphahat = search$par[1]; lambdahat = search$par[2]
>
> ################################################################################
> # Plot hazard function h(t) = alpha*lambda^alpha * t^(alpha-1)
> ################################################################################
>
> x = seq(from=0,to=5,length=101)
> esthazard = alphahat*lambdahat^alphahat * x^(alphahat-1)
> truehazard = (x-2)^2
>
> plot(x,truehazard,type='l',xlab='Time',ylab='Hazard', main='Hazard Function for the Bowl
Data')
> lines(x,esthazard,lty=2)
> # Annotate the plot (Make the legend)
> x1 = c(0.5,1.5); y1 = c(7,7)
> lines(x1,y1,lty=1)
> text(2,7,'True h(t)')
> x2 = c(0.5,1.5); y2 = c(6,6)
> lines(x2,y2,lty=2)
> text(2.2,6,'Estimated h(t)')
```




```
> # This is bad, but the worst part is outside the range of the data: Max = 3.72
> # 75th percentile is 0.38
> # From HW4, S(t) = exp(-1/3 ((t-2)^3 + 8)
> # At t=2, where hazard starts increasing,
> exp(-8/3)
[1] 0.06948345
> # So for 93% of the distribution, the hazard function is decreasing.
```

```
> # Try estimating the survival function
> x = seq(from=0,to=5,length=101)
> Shat = exp(-(lambdahat*x)^alphahat)
> trueS = exp( -1/3*((x-2)^3 + 8) )
> tstring = 'Survival Function for the Bowl Data'
> plot(x,trueS,type='l',xlab='Time',ylab='Survival',ylim=c(0,1), main=tstring)
> lines(x,Shat,lty=2)
> # Annotate the plot (Make the legend)
> x1 = c(2.5,3.5); y1 = c(0.8,0.8)
> lines(x1,y1,lty=1)
> text(4,0.8,'True S(t)')
> x2 = x1; y2 = c(0.7,0.7)
> lines(x2,y2,lty=2)
> text(4.2,0.7,'Estimated S(t)')
```