Maximum Likelihood with R[*]

Background and reference: Section A.6.3 from Appendix A in *Structural Equation Models*

```
>
> ###############################################################################
> # Question 2: Normal random sample
> ###############################################################################
>
> rm(list=ls()); options(scipen=999) # Option to suppress scientific notation
> # Read normal data
> normaldata = scan("http://www.utstat.toronto.edu/~brunner/data/legal/normal.data.txt")
Read 200 items
> normaldata
  [1] 116.3 112.6 107.4  88.4 143.6 113.6  85.6  97.2 121.3 114.8  93.3 116.1
 [13]  93.8  93.6  98.3  95.5  97.3 135.0  95.5 126.0 107.4 113.6 102.3  98.3
 [25]  99.5  92.8  78.4 116.7  96.6 103.0  89.3  68.8  93.8 117.7 113.8 105.2
 [37]  97.8  90.8  79.0  87.8 107.7 102.1 102.2  76.7  85.2 103.7  92.3  97.4
 [49]  83.5  87.3 113.7  94.7 113.6  99.7 131.3  65.6 105.2  87.9 105.2 110.4
 [61] 113.9 119.5  69.6 107.9  98.1  99.0  65.1  82.8  86.0 104.4  95.5  86.2
 [73] 100.1 131.5 110.0  96.0  87.4 132.1  97.0  95.3 114.1 106.9  99.1  78.3
 [85]  93.3  95.8 105.8  86.7 104.0 113.3 117.2  80.5  81.8  82.9  96.2  81.0
 [97] 130.5 122.4 118.0  98.7  83.2 114.9 104.3  98.8 115.7  94.3 101.6 126.3
[109] 101.1  94.3  97.7  97.8 126.6 110.5 103.3  82.5  88.3  91.8  88.1  95.7
[121]  97.1 108.0  98.7 118.8  95.8  94.8  84.4 126.7 101.3  72.8  96.1 105.1
[133]  89.4 126.0  86.0 106.5 119.0  85.8  76.6  91.7  96.8  97.4  89.9 130.4
[145]  63.9  93.0  92.0  99.6 103.3  95.8 105.7 103.5  94.6 124.3 104.6 105.7
[157]  89.4 110.6 108.6 106.5 121.0 119.7  76.7 104.8  93.4 114.4  82.3  97.7
[169]  94.1  95.6 113.6 106.5  88.2 109.2 113.9  95.0 130.9  99.8 120.7  86.8
[181]  82.7 100.4  85.4 117.0  69.2  91.8  99.8  98.7  91.7 118.4 104.8 105.0
[193]  81.6  96.7  91.3  93.3  96.3  96.7 124.1 126.8
>
> ###############################################################################
> # (a) Find the maximum likelihood estimates of mu and sigma-squared numerically.
> ###############################################################################
>
> mloglike = function(theta,x) # theta is (mu,sigmasq)
+     {
+     mu = theta[1]; sigmasq = theta[2]
+     n = length(x)
+     value = n/2*log(sigmasq) + n/2*log(2*pi) + sum((x-mu)^2)/(2*sigmasq)
+     return(value)
+     } # End function mloglike
>
> # Test the function -- easy in this case
> mloglike(c(90,220),normaldata)
[1] 871.4047
> -sum(dnorm(normaldata,90,sqrt(220),log=TRUE)) # Get log of density as an option
[1] 871.4047
>
> # help(optim) # Minimize a function
> startvals = c(0,1) # Of course the sample mean and variance would be better.
>
> normalsearch = optim(par=startvals, fn=mloglike, x=normaldata,
+                      hessian=TRUE, lower=c(-Inf,0), method='L-BFGS-B')
```

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

```
> normalsearch
$par
[1] 100.4805 216.3154

$value
[1] 821.4629

$counts
function gradient 
      33       33 

$convergence
[1] 0

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

$hessian
                [,1]            [,2]
[1,]  0.924575658701 -0.000000198952
[2,] -0.000000198952  0.002137142019

> 
> # If you really have no idea what the parameter values are, it's wise to try
> # several sets of starting values. Here's a really lucky guess.
> optim(par=c(mean(normaldata),var(normaldata)), fn=mloglike, x=normaldata,
+             hessian=TRUE, lower=c(-Inf,0), method='L-BFGS-B')
$par
[1] 100.4805 216.3186

$value
[1] 821.4629

$counts
function gradient 
       4        4 

$convergence
[1] 0

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

$hessian
          [,1]        [,2]
[1,] 0.9245623 0.000000000
[2,] 0.0000000 0.002137057
```

```
> # We will stay with the first solution.
>
> ##############################################################################
> # Compare the answer to your closed-form solution.
> ##############################################################################
>
> muhat = mean(normaldata)
> n = length(normaldata); sigmasqhat = (n-1)*var(normaldata)/n
> c(muhat,sigmasqhat) # The exact MLE
[1] 100.4805 216.3185
>
> ##############################################################################
> # (b) Show that the minus log likelihood is indeed minimized at the MLE for
> #     this data set.
> ##############################################################################
>
> # If both eigenvalues of the Hessian are positive, it's concave up.
> H = normalsearch$hessian
> eigen(H)$values
[1] 0.924575659 0.002137142
>
> ##############################################################################
> # (c) Calculate the estimated asymptotic covariance matrix of the MLEs.
> ##############################################################################
>
> Vhat = solve(H); Vhat # Solve returns the inverse.
             [,1]           [,2]
[1,] 1.0815772518   0.0001006868
[2,] 0.0001006868 467.9146220382
>
> ##############################################################################
> # (d) Give a ``better" estimated asymptotic covariance matrix based on your
> #     closed-form solution.
> ##############################################################################
>
> betterVhat = rbind(c(sigmasqhat/n, 0               ),
+                    c(0           , 2*sigmasqhat^2/n) )
> betterVhat
         [,1]     [,2]
[1,] 1.081592   0.0000
[2,] 0.000000 467.9368
>
> ##############################################################################
> # (e) Calculate a large-sample 95\% confidence interval for sigma-squared.
> ##############################################################################
>
> se = sqrt(Vhat[2,2])
> sigmasqhat = normalsearch$par[2] # Replacing the exact answer. It's almost
                                   # the same.
> lower95 = sigmasqhat - 1.96*se; upper95 = sigmasqhat + 1.96*se
> c(lower95,upper95)
[1] 173.9180 258.7129
```

```
>
>
> ##########################################################################
> # (f) Test H0: mu = 103 with
>     # (i) Z test
>     # (ii) Likelihood ratio chi-squared test. Compare the closed-form version.
>     # (iii) Wald chi-squared test.
> # Give the test statistic and the p-value for each test.
> ##########################################################################
>
> ##########################################################################
> # (i) Z test
> ##########################################################################
> se2 = sqrt(Vhat[1,1]); Z = (muhat-103)/se2
> pval = 2 * (1-pnorm(abs(Z)))
> c(Z,pval)
[1] -2.42262163  0.01540897
>
> ##########################################################################
> # (ii) Likelihood ratio chi-squared test
> ##########################################################################
>
> # LR test could be done in closed form, but do it numerically
> # Function for constrained minimization
> mloglike0 = function(theta,x) # theta is just sigmasq
+     {
+     mu = 103; sigmasq = theta
+     n = length(x)
+     value = n/2*log(sigmasq) + n/2*log(2*pi) + sum((x-mu)^2)/(2*sigmasq)
+     return(value)
+     } # End function mloglike0
>
> # Estimate. Don't ask for the Hessian.
> restricted1 = optim(par=sigmasqhat, fn=mloglike0, x=normaldata,
+                 lower=0, method='L-BFGS-B')
> restricted1
$par
[1] 222.6663

$value
[1] 824.3552

$counts
function gradient
       5        5

$convergence
[1] 0

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

>
>
> Gsq = 2 * (restricted1$value - normalsearch$value)
> pval = 1-pchisq(Gsq,1)
> c(Gsq,pval)
[1] 5.78454876 0.01616765
```

```
> Gsq = 2 * (restricted1$value - normalsearch$value)
> pval = 1-pchisq(Gsq,1)
> c(Gsq,pval)
[1] 5.78454876 0.01616765

>
> ###############################################################################
> # Compare the closed-form version.
> ###############################################################################
>
> sigmasqhat0 = sum( (normaldata-103)^2 )/n
> n * ( log(sigmasqhat0) - log(sigmasqhat) ) # G-squared in closed form
[1] 5.787343
>
> ###############################################################################
> # (iii) Wald chi-squared test.
> ###############################################################################
>
> # Use my function for the Wald test
> source("http://www.utstat.toronto.edu/~brunner/Rfunctions/Wtest.txt")
> Wtest # To see the function definition

function(L,Tn,Vn,h=0) # H0: L theta = h
# Tn is estimated theta, usually a vector.
# Vn is the estimated asymptotic covariance matrix of Tn.
# For Wald tests based on numerical MLEs, Tn = theta-hat,
# and Vn is the inverse of the Hessian of the minus log
# likelihood.
     {
     Wtest = numeric(3)
     names(Wtest) = c("W","df","p-value")
     r = dim(L)[1]
     W = t(L%*%Tn-h) %*% solve(L%*%Vn%*%t(L)) %*%
          (L%*%Tn-h)
     W = as.numeric(W)
     pval = 1-pchisq(W,r)
     Wtest[1] = W; Wtest[2] = r; Wtest[3] = pval
     Wtest
     }
>
> LL = cbind(1,0); hh = 103 # For testing H0: L theta = h
> thetahat = normalsearch$par; thetahat
[1] 100.4805 216.3154
> Wtest(LL,thetahat,Vhat,hh)
         W         df    p-value
5.86888878 1.00000000 0.01541078
>
> Z^2 # For one-df tests, Wald test statistic is Z-squared
[1] 5.869096
```

```
>
>
> ##############################################################################
> # (g) Coefficient of variation
> ##############################################################################
>
> # Note that (g) part (i) is paper and pencil
>
> ##############################################################################
> # (ii) Point estimate of coefficient of variation
> ##############################################################################
>
> cvhat = sqrt(sigmasqhat)/muhat;  cvhat
[1] 0.1463733
>
> ##############################################################################
> # (iii) CI for coefficient of variation
> ##############################################################################
>
> # Get standard error using delta method.
>
> gdot = cbind( -sqrt(sigmasqhat)/muhat^2, 1/( 2*muhat*sqrt(sigmasqhat) ) )
> # That's gdot evaluated at the MLE.
> # cbind makes it a 1x2 matrix.
> gdot
             [,1]         [,2]
[1,] -0.001456734 0.0003383331
>
> se_cvhat = sqrt( as.numeric( gdot %*% Vhat %*% t(gdot) ) )
> # as.numeric makes it a real number
> se_cvhat
[1] 0.007473749
> # 95% confidence interval
> c(cvhat - 1.96*se_cvhat, cvhat + 1.96*se_cvhat)
[1] 0.1317248 0.1610219
>
```