Statistics 580

Maximum Likelihood Estimation

Introduction

Let $\mathbf{y} = (y_1, y_2, \ldots, y_n)'$ be a vector of iid, random variables from one of a family of distributions on $\Re^n$ and indexed by a p-dimensional parameter $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)'$ where $\boldsymbol{\theta} \,\epsilon\, \Omega \subset \Re^p$ and $p \leq n$. Denote the distribution function of $\mathbf{y}$ by $F(\mathbf{y}|\boldsymbol{\theta})$ and assume that the density function $f(\mathbf{y}|\boldsymbol{\theta})$ exists. Then the likelihood function of $\boldsymbol{\theta}$ is given by

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} f(y_i|\boldsymbol{\theta})\,.$$

In practice, the natural logarithm of the likelihood function, called the log-likelihood function and denoted by

$$\ell(\boldsymbol{\theta}) = \log\, L(\boldsymbol{\theta}) = \sum_{i=1}^{n} log(f(y_i|\boldsymbol{\theta}))\,,$$

is used since it is found to be easier to manipulate algebraically. Let the p partial derivatives of the log-likelihood form the $p \times 1$ vector

$$\mathbf{u}(\boldsymbol{\theta}) = \frac{\partial\, \ell\,(\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \begin{pmatrix} \frac{\partial \ell}{\partial \theta_1} \\ \vdots \\ \frac{\partial \ell}{\partial \theta_p} \end{pmatrix}.$$

The vector $\mathbf{u}(\boldsymbol{\theta})$ is called the *score vector* of the log-likelihood function. The moments of $\mathbf{u}(\boldsymbol{\theta})$ satisfy two important identities. First, the expectation of $\mathbf{u}(\boldsymbol{\theta})$ with respect to $\mathbf{y}$ is equal to zero, and second, the variance of $\mathbf{u}(\boldsymbol{\theta})$ is the negative of the second derivative of $\ell(\boldsymbol{\theta})$, i.e.,

$$\mathrm{Var}(\mathbf{u}(\boldsymbol{\theta})) = -E\left\{\mathbf{u}(\boldsymbol{\theta})\, \mathbf{u}\,(\boldsymbol{\theta})^T\right\} = -E\left\{\left(\frac{\partial^2\, \ell(\boldsymbol{\theta})}{\partial\, \theta_j\, \partial \theta_k}\right)\right\}.$$

The $p \times p$ matrix on the right hand side is called the *expected Fisher information* matrix and usually denoted by $\mathcal{I}(\boldsymbol{\theta})$. The expectation here is taken over the distribution of $\mathbf{y}$ at a fixed value of $\boldsymbol{\theta}$. Under conditions which allow the operations of integration with respect to $\mathbf{y}$ and differentiation with respect to $\boldsymbol{\theta}$ to be interchanged, the maximum likelihood estimate of $\boldsymbol{\theta}$ is given by the solution $\hat{\boldsymbol{\theta}}$ to the p equations

$$\mathbf{u}(\hat{\boldsymbol{\theta}}) = \mathbf{0}$$

and under some regularity conditions, the distribution of $\hat{\boldsymbol{\theta}}$ is asymptotically normal with mean $\boldsymbol{\theta}$ and variance-covariance matrix given by the $p \times p$ matrix $\mathcal{I}(\boldsymbol{\theta})^{-1}$ i.e., the inverse of the *expected information* matrix. The $p \times p$ matrix

$$\mathbf{I}(\boldsymbol{\theta}) = -\left\{\frac{\partial^2\, \ell\,(\boldsymbol{\theta})}{\partial\, \theta_j\, \partial\, \theta_k}\right\}$$

is called the *observed information* matrix. In practice, since the true value of $\boldsymbol{\theta}$ is not known, these two matrices are estimated by substituting the estimated value $\hat{\boldsymbol{\theta}}$ to give $\mathcal{I}(\hat{\boldsymbol{\theta}})$ and $\mathbf{I}(\hat{\boldsymbol{\theta}})$, respectively. Asymptotically, these four forms of the information matrix can be shown to be equivalent.

From a computational standpoint, the above quantities are related to those computed to solve an optimization problem as follows: $-\ell(\boldsymbol{\theta})$ corresponds to the *objective function* to be minimized, $\mathbf{u}(\boldsymbol{\theta})$ represents the *gradient vector*, the vector of first-order partial derivatives, usually denoted by $\mathbf{g}$, and $\mathbf{I}(\boldsymbol{\theta})$, corresponds to the negative of the Hessian matrix $H(\boldsymbol{\theta})$, the matrix of second-order derivatives of the objective function, respectively. In the MLE problem, the Hessian matrix is used to determine whether the minimum of the objective function $-\ell(\boldsymbol{\theta})$ is achieved by the solution $\hat{\boldsymbol{\theta}}$ to the equations $\mathbf{u}(\boldsymbol{\theta}) = 0$, i.e., whether $\hat{\boldsymbol{\theta}}$ is a stationery point of $\ell(\boldsymbol{\theta})$. If this is the case, then $\hat{\boldsymbol{\theta}}$ is the maximum likelihood estimate of $\boldsymbol{\theta}$ and the asymptotic covariance matrix of $\hat{\boldsymbol{\theta}}$ is given by the inverse of the negative of the Hessian matrix evaluated at $\hat{\boldsymbol{\theta}}$, which is the same as $\mathbf{I}(\hat{\boldsymbol{\theta}})$, the observed information matrix evaluated at $\hat{\boldsymbol{\theta}}$.

Sometimes it is easier to use the observed information matrix $\mathbf{I}(\hat{\boldsymbol{\theta}})$ for estimating the asymptotic covariance matrix of $\hat{\boldsymbol{\theta}}$, since if $\mathcal{I}(\hat{\boldsymbol{\theta}})$ were to be used then the expectation of $\mathbf{I}(\hat{\boldsymbol{\theta}})$ needs to be evaluated analytically. However, if computing the derivatives of $\ell(\boldsymbol{\theta})$ in closed form is difficult or if the optimization procedure does not produce an estimate of the Hessian as a byproduct, estimates of the derivatives obtained using finite difference methods may be substituted for $\mathbf{I}(\hat{\boldsymbol{\theta}})$.

Newton-Raphson method is widely used for function optimization. Recall that the iterative formula for finding a maximum or a minimum of f($\mathbf{x}$) was given by

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} - \mathbf{H}_{(i)}^{-1}\mathbf{g}_{(i)},$$

where $\mathbf{H}_{(i)}$ is the Hessian $\mathbf{f}''(\mathbf{x}_{(\mathbf{i})})$ and $\mathbf{g}_{(i)}$ is the gradient vector $\mathbf{f}'(\mathbf{x}_{(\mathbf{i})})$ of $f(\mathbf{x})$ at the i^{th} iteration. The Newton-Raphson method requires that the starting values be sufficiently close to the solution to ensure convergence. Under this condition the Newton-Raphson iteration converges quadratically to at least a local optimum. When Newton-Raphson method is applied to the problem of maximizing the likelihood function the i^{th} iteration is given by

$$\hat{\boldsymbol{\theta}}_{(i+1)} = \hat{\boldsymbol{\theta}}_{(i)} - H(\hat{\boldsymbol{\theta}}_{(i)})^{-1}\,\mathbf{u}(\hat{\boldsymbol{\theta}}_{(i)}).$$

We shall continue to use the Hessian matrix notation here instead of replacing it with $-I(\boldsymbol{\theta})$. Observe that the Hessian needs to be computed and inverted at every step of the iteration. In difficult cases when the Hessian cannot be evaluated in closed form, it may be substituted by a discrete estimate obtained using finite difference methods as mentioned above . In either case, computation of the Hessian may end up being a substantially large computational burden. When the expected information matrix $\mathcal{I}(\boldsymbol{\theta})$ can be derived analytically without too much difficulty, i.e., the expectation can be expressed as closed form expressions for the elements of $\mathcal{I}(\boldsymbol{\theta})$, and hence $\mathcal{I}(\boldsymbol{\theta})^{-1}$, it may be substituted in the above iteration to obtain the modified iteration

$$\hat{\boldsymbol{\theta}}_{(i+1)} = \hat{\boldsymbol{\theta}}_{(i)} + \mathcal{I}(\hat{\boldsymbol{\theta}}_{(i)})^{-1}\,\mathbf{u}(\hat{\boldsymbol{\theta}}_{(i)}).$$

This saves on the computation of $H(\hat{\boldsymbol{\theta}})$ because functions of the data $\mathbf{y}$ are not involved in the compuitation of $\mathcal{I}(\hat{\boldsymbol{\theta}}_{(i)})$ as they are with the computation of $H(\hat{\boldsymbol{\theta}})$. This provides a sufficiently accurate Hessian to correctly orient the direction to the maximum. This procedure is called the *method of scoring*, and can be as effective as Newton-Raphson for obtaining the maximum likelihood estimates iteratively.

Example 1:

Let $y_1, \ldots, y_n$ be a random sample from $N(\mu, \sigma^2)$, $-\infty < \mu < \infty$, $0 < \sigma^2 < \infty$. Then

$$\begin{aligned} L(\mu, \sigma^2)|\mathbf{y}) &= \prod_{i=1}^{n} \frac{1}{(2\pi)^{1/2}\sigma} \exp\left[-(y_i-\mu)^2/2\sigma^2\right] \\ &= \frac{1}{(2\pi)^{n/2}\sigma^n} \exp\left[-\sum(y_i-\mu)^2/2\sigma^2\right] \end{aligned}$$

giving

$$\ell(\mu, \sigma^2) = \log L(\mu, \sigma^2|\mathbf{y}) = -n\log\sigma \; - \sum(y_i-\mu)^2/2\sigma^2 + \text{constant}.$$

The first partial derivatives are:

$$\begin{aligned} \frac{\partial \ell}{\partial \mu} &= \frac{2\Sigma(y_i-\mu)}{2\sigma^2} \\ \frac{\partial \ell}{\partial \sigma^2} &= \frac{\Sigma(y_i-\mu)^2}{2\sigma^4} - \frac{n}{2\sigma^2} \end{aligned}$$

Setting them to zero,

$$\sum y_i - n\,\mu = 0$$

and

$$\sum(y_i-\mu)^2/2\sigma^4 - n/2\sigma^2 = 0,$$

give the m.l.e.'s $\hat{\mu} = \bar{y}$, and $\hat{\sigma}^2 = \sum(y_i-\bar{y})^2/n$, respectively.

The observed information matrix $I(\boldsymbol{\theta})$ is:

$$\begin{aligned} I(\boldsymbol{\theta}) &= -\left\{\frac{\partial^2 \ell}{\partial\theta_j \partial\theta_k}\right\} \\ &= -\begin{bmatrix} \frac{\Sigma(-1)}{\sigma^2} & \frac{-\Sigma(y_i-\mu)}{\sigma^4} \\ \frac{-\Sigma(y_i-\mu)}{\sigma^4} & \frac{-\Sigma(y_i-\mu)^2}{\sigma^6} + \frac{n}{2\sigma^4} \end{bmatrix} \\ &= \begin{bmatrix} \frac{n}{\sigma^2} & \frac{\Sigma(y_i-\mu)}{\sigma^4} \\ \frac{\Sigma(y_i-\mu)}{\sigma^4} & \frac{\Sigma(y_i-\mu)^2}{\sigma^6} - \frac{n}{2\sigma^4} \end{bmatrix} \end{aligned}$$

and the expected information matrix $\mathcal{I}(\boldsymbol{\theta})$ is:

$$\begin{aligned} \mathcal{I}(\boldsymbol{\theta}) &= \begin{bmatrix} \frac{n}{\sigma^2} & 0 \\ 0 & \frac{n\sigma^2}{\sigma^6} - \frac{n}{2\sigma^4} \end{bmatrix} = \begin{bmatrix} \frac{n}{\sigma^2} & 0 \\ 0 & \frac{n}{2\sigma^4} \end{bmatrix} \\ \mathcal{I}(\boldsymbol{\theta})^{-1} &= \begin{bmatrix} \frac{\sigma^2}{n} & 0 \\ 0 & \frac{2\sigma^4}{n} \end{bmatrix} \end{aligned} \qquad \square$$

Example 2:

In this example, Fisher scoring is used to obtain the m.l.e. of θ using a sample of size n from the Cauchy distribution with density:

$$f(x) = \frac{1}{\pi}\frac{1}{1+(x-\theta)^2}, \qquad -\infty < x < \infty\,.$$

The likelihood function is

$$L(\theta) = \left(\frac{1}{\pi}\right)^n \prod_{i=1}^{n} \frac{1}{1+(x_i-\theta)^2}$$

and the log-likelihood is

$$\log L(\theta) = -\sum \log\left(1+(x_i-\theta)^2\right) + \text{constant}$$

giving

$$\frac{\partial \log L}{\partial \theta} = \sum_{i=1}^{n} \frac{2(x_i-\theta)}{1+(x_i-\theta)^2} = \mathbf{u}(\boldsymbol{\theta})\,.$$

Information $\mathcal{I}(\theta)$ is

$$\mathcal{I}(\theta) = \frac{\partial^2 \log L}{\partial \theta^2} = n/2$$

and therefore

$$\theta_{i+1} = \theta_i + \frac{2u(\theta_i)}{n}$$

is the iteration formula required for the scoring method. □

Example 3:

The following example is given by Rao (1973). The problem is to estimate the gene frequencies of blood antigens A and B from observed frequencies of four blood groups in a sample. Denote the gene frequencies of A and B by θ_1 and θ_2 respectively, and the expected probabilities of the four blood group O, A, B and AB by π_1, π_2, π_3, and π_4 . These probabilities are functions of θ_1 and θ_2 and are given as follows:

$$\begin{aligned}
\pi_1(\theta_1,\theta_2) &= 2\theta_1\,\theta_2\\
\pi_2(\theta_1,\theta_2) &= \theta_1(2-\theta_1-2\theta_2)\\
\pi_3(\theta_1,\theta_2) &= \theta_2(2-\theta_2-2\theta_1)\\
\pi_4(\theta_1,\theta_2) &= (1-\theta_1-\theta_2)^2
\end{aligned}$$

The joint distribution of the observed frequencies y_1, y_2, y_3, y_4 is a multinomial with $n = y_1+y_2+y_3+y_4$.

$$f(y_1,y_2,y_3,y_4) = \frac{n!}{y_1!y_2!y_3!y_4!}\,\pi_1^{y_1}\,\pi_2^{y_2}\,\pi_3^{y_3}\,\pi_4^{y_4}\,.$$

It follows that the log-likelihood can be written as

$$\log L(\boldsymbol{\theta}) = y_1 \log \pi_1 + y_2 \log \pi_2 + y_3 \log \pi_3 + y_4 \log \pi_4 + \text{constant}\,.$$

The likelihood estimates are solutions to

$$\begin{aligned} \frac{\partial \log L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \left(\frac{\partial \boldsymbol{\pi}}{\partial \boldsymbol{\theta}}\right)^T \cdot \left(\frac{\partial \log L(\boldsymbol{\theta})}{\partial \boldsymbol{\pi}}\right) = 0 \\ \text{i.e., } \ \mathbf{u}(\boldsymbol{\theta}) &= \sum_{j=1}^{4} (y_j/\pi_j) \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right) = 0, \end{aligned}$$

where $\boldsymbol{\pi} = (\pi_1,\, \pi_2,\, \pi_3,\, \pi_4)^T$, $\boldsymbol{\theta} = (\theta_1,\, \theta_2)^T$, and

$$\frac{\partial \pi_1}{\partial \boldsymbol{\theta}} = \begin{pmatrix} 2\,\theta_2 \\ 2\,\theta_1 \end{pmatrix},\ \frac{\partial \pi_2}{\partial \boldsymbol{\theta}} = \begin{pmatrix} 2\,(1-\theta_1-\theta_2) \\ -2\theta_1 \end{pmatrix},\ \frac{\partial \pi_3}{\partial \boldsymbol{\theta}} = \begin{pmatrix} -2\theta_2 \\ 2(1-\theta_1-\theta_2) \end{pmatrix},\ \frac{\partial \pi_4}{\partial \boldsymbol{\theta}} = \begin{pmatrix} -2(1-\theta_1-\theta_2) \\ -2(1-\theta_1-\theta_2) \end{pmatrix}.$$

The Hessian of the log-likelihood function is the 2×2 matrix:

$$\begin{aligned} H(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{\partial \log L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right) &= \sum_{j=1}^{4} \left[(y_j/\pi_j) \frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right) + \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right) \cdot \frac{\partial}{\partial \boldsymbol{\theta}} (y_j/\pi_j) \right] \\ &= \sum_{j=1}^{4} \left[(y_j/\pi_j)\, H_j(\boldsymbol{\theta}) - \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right) (y_j/\pi_j^2) \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right)^T \right] \\ &= \sum_{j=1}^{4} y_j \left[(1/\pi_j)\, H_j(\boldsymbol{\theta}) - (1/\pi_j^2) \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right) \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right)^T \right] \end{aligned}$$

where $H_j(\boldsymbol{\theta}) = \left(\frac{\partial^2 \pi_j}{\partial \theta_h \partial \theta_k}\right)$ is the 2×2 Hessian of $\pi_j,\ j = 1, 2, 3, 4$. These are easily computed by differentiation of the gradient vectors above:

$$H_1 = \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix},\ H_2 = \begin{pmatrix} -2 & -2 \\ -2 & 0 \end{pmatrix},\ H_3 = \begin{pmatrix} 0 & -2 \\ -2 & -2 \end{pmatrix},\ H_4 = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}.$$

The information matrix is then given by:

$$\begin{aligned} \mathcal{I}(\boldsymbol{\theta}) = E\left\{ -\frac{\partial^2 \log(\boldsymbol{\theta})}{\partial \theta_j \partial \theta_k} \right\} &= -\sum_{j=1}^{4} n\,\pi_j \left[(1/\pi_j) H_j(\boldsymbol{\theta}) - (1/\pi_j^2) \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right) \left(\frac{\partial \pi_j)}{\partial \boldsymbol{\theta}}\right)^T \right] \\ &= n \sum_{j=1}^{4} (1/\pi_j) \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right) \left(\frac{\partial \pi_j}{\partial \boldsymbol{\theta}}\right)^T \end{aligned}$$

Thus elements of $\mathcal{I}(\boldsymbol{\theta})$ can be expressed as closed form functions of $\boldsymbol{\theta}$. In his discussion, Rao gives the data as $y_1 = 17, y_2 = 182, y_3 = 60, y_4 = 176$ and $n = y_1 + y_2 + y_3 + y_4 = 435$.

The first step in using iterative methods for computing m.l.e.'s is to obtain starting values for θ_1 and θ_2. Rao suggests some methods for providing good estimates. A simple choice is to set $\pi_1 = y_1/n$, $\pi_2 = y_2/n$ and solve for θ_1 and θ_2. This gives $\theta_{(0)} = (0.263, 0.074)'$. Three methods are used below for computing m.l.e.'s; *method of scoring, Newton-Raphson with analytical derivatives* and, *Newton-Raphson with numerical derivatives*, and results of several iterations are tabulated:

Table 1: Convergence of iterative methods for computing maximum likelihood estimates.
(a) Method of Scoring

Iteration No.	θ_1	θ_2	$\mathcal{I}_{11}$	$\mathcal{I}_{12}$	$\mathcal{I}_{22}$	$\log L(\boldsymbol{\theta})$
0	.263000	.074000	2.98150	28.7419	2.43674	-494.67693
1	.164397	.092893	9.00457	23.2801	2.47599	-492.52572
2	.264444	.093168	9.00419	23.2171	2.47662	-492.53532

(b) Newton's Method with Hessian computed directly using analytical derivatives

Iteration No.	θ_1	θ_2	H_{11}	H_{12}	H_{22}	$\log L(\boldsymbol{\theta})$
0	.263000	.074000	-3872.33	-15181.07	-1006.17	-494.67693
1	.265556	.089487	-3869.46	-10794.02	-1059.73	-492.60313
2	.264480	.093037	-3899.85	-10083.28	-1067.56	-492.53540
3	.264444	.093169	-3900.90	-10058.49	-1067.86	-492.53532

(c) Newton's Method with Hessian computed numerically using finite differences

Iteration No.	θ_1	θ_2	$\hat{H}_{11}$	$\hat{H}_{12}$	$\hat{H}_{22}$	$\log L(\boldsymbol{\theta})$
0	.263000	.074000	-3823.34	-14907.05	-1010.90	-494.67693
1	.265494	.089776	-3824.23	-10548.51	-1065.84	-492.59283
2	.264440	.093118	-3853.20	-9896.60	-1073.19	-492.52533
3	.264444	.093170	-3853.29	-9887.19	-1073.37	-492.53532

```
"p"<-function(k, t1, t2)
{
        switch(k,
           2 * t1 * t2,
           t1 * (2 - t1 - 2 * t2),
           t2 * (2 - t2 - 2 * t1),
           (1 - t1 - t2)^2)
}
```

Computes π_k

```
-----------------------------------------------------------------
"u"<-function(k, t1, t2)
{
        switch(k,
           c(2 * t2, 2 * t1),
           c(2 * (1 - t1 - t2), -2 *t1),
           c(-2 * t2, 2 * (1 - t1 - t2)),
           c(-2 * (1 - t1 - t2), -2 * (1 - t1 - t2)))
}
```

Computes $\underset{\sim}{u}_k = \frac{\partial \pi_k}{\partial \underset{\sim}{\theta}}$

```
-----------------------------------------------------------------
"gradient"<-function(y, t1, t2)
{
        ii <- c(0, 0)
        for(j in 1:4) {
           c <- 1/p(j, t1, t2)
           uj <- u(j, t1, t2)
           ii <- ii + y[j] * c * uj
        }
        ii
}
```

Computes $\frac{\partial \ell}{\partial \underset{\sim}{\theta}} = \sum_{j=1}^{4} (y_j / \pi_j) \underset{\sim}{u}_j$

```
-----------------------------------------------------------------
"hessian"<-function(y, t1, t2)
{
        ii <- matrix(0, 2, 2)
        for(j in 1:4) {
           c <- 1/p(j, t1, t2)
           uj <- u(j, t1, t2)
           hj <- h[[j]]
           ii <- ii + y[j] * (c * hj - c^2 * outer(uj, uj, "*"))
        }
        ii
}
```

Computes $H(\underset{\sim}{\theta})$

$$= \sum_{j=1}^{4} y_j \left[(1/\pi_j) H_j(\underset{\sim}{\theta}) - (1/\pi_j^2) \underset{\sim}{u}_j \underset{\sim}{u}_j^T \right]$$

where $H_1 = \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix}$, $H_2 = \begin{pmatrix} -2 & -2 \\ -2 & 0 \end{pmatrix}$, $H_3 = \begin{pmatrix} 0 & -2 \\ -2 & -2 \end{pmatrix}$, $H_4 = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}$.

```
-----------------------------------------------------------------
"inf"<-function(t1, t2)
{
        ii <- matrix(0, 2, 2)
        for(j in 1:4) {
           c <- 1/p(j, t1, t2)
           uj <- u(j, t1, t2)
           ii <- ii + c * outer(uj,uj, "*")
        }
        ii
}
```

Computes $\mathcal{I}(\underset{\sim}{\theta}) = n \sum_{j=1}^{4} (1/\pi_j) \underset{\sim}{u}_j \underset{\sim}{u}_j^T$

```
-----------------------------------------------------------------
"logl"<-function(y, t1, t2)
{
        t(y) %*% log(unlist(lapply(1:4,
           p, t1, t2)))
}

-----------------------------------------------------------------
"hhat"<-function(f, y, t1, t2)
{
        ii <- matrix(0, 2, 2)
        h1 <- 0.01 * t1
        h2 <- 0.01 * t2
        ii[1, 1] <- ((f(y, t1 + h1 + h1,t2) - f(y, t1 + h1, t2)) -
           (f(y, t1 + h1, t2) - f(y, t1, t2)))/(h1 * h1)
        ii[1, 2] <- ((f(y, t1 + h1, t2 + h2) - f(y, t1 + h1, t2)) -
           (f(y, t1, t2 + h2) - f(y, t1, t2)))/(h1 * h2)
        ii[2, 1] <- ((f(y, t1 + h1, t2 + h2) - f(y, t1, t2 + h2)) -
           - (f(y, t1 + h1, t2) - f(y, t1, t2)))/(h1 * h2)
        ii[2, 2] <- ((f(y, t1, t2 + h2 + h2) - f(y, t1, t2 + h2)) -
           (f(y, t1, t2 + h2) - f(y, t1, t2)))/(h2 * h2)
        ii
}
-----------------------------------------------------------------
```

Computes estimate of H using finite difference method. See below *

$$* \quad \hat{H}_{ij} = \left\{ \left[f(\underset{\sim}{\theta} + h_i \underset{\sim}{e}_i + h_j \underset{\sim}{e}_j) - f(\underset{\sim}{\theta} + h_i \underset{\sim}{e}_i) \right] - \left[f(\underset{\sim}{\theta} + h_j \underset{\sim}{e}_j) - f(\underset{\sim}{\theta}) \right] \right\} / h_i h_j .$$

$$h_i = .01\, \theta_i \qquad h_j = .01\, \theta_j$$

```
"newton"<-function(y, t1, t2)
{
        last <- 0
        repeat {
           m <- hessian(y, t1, t2)
           theta <- c(t1, t2) - solve(m) %*% gradient(y, t1, t2)
           l <- logl(y, t1, t2)
           cat(t1, t2, m[1, 1], m[1, 2], m[2, 2], l, fill = T)
           t1 <- theta[1]
           t2 <- theta[2]
           if(abs(last - l)/abs(l) <0.0001)
            break
           last <- l
        }
        l
}
```

Newton-Raphson iteration using observed information matrix: $I(\underset{\sim}{\theta})$

```
"newtest"<-function(y, t1, t2)
{
        last <- 0
        repeat {
           m <- hhat(logl, y, t1, t2)
           theta <- c(t1, t2) - solve(m) %*% gradient(y, t1, t2)
           l <- logl(y, t1, t2)
           cat(t1, t2, m[1, 1], m[1, 2], m[2, 2], l, fill = T)
           t1 <- theta[1]
           t2 <- theta[2]
           if(abs(last - l)/abs(l) <0.0001)
            break
           last <- l
        }
}
```

Newton-Raphson iteration using H estimated using finite differences: $\hat{H}$

```
"scoring"<-function(y, t1, t2)
{
        last <- 0
        repeat {
           m <- inf(t1, t2)
           theta <- c(t1, t2) + (1/435) * solve(m) %*%
                 gradient(y, t1, t2)
           l <- logl(y, t1, t2)
           cat(t1, t2, m[1, 1], m[1, 2], m[2, 2], l, fill = T)
           t1 <- theta[1]
           t2 <- theta[2]
           if(abs(last - l)/abs(l) <0.0001)
            break
           last <- l
        }
        l
}
```

Newton-Raphson iteration using expected information matrix: $\mathcal{I}(\underset{\sim}{\theta})$

"method of scoring"

4.2.6.1 A multinomial problem

A classic example of maximum likelihood estimation is due to Fisher (1925) and arises in a genetics problem. Consider a multinomial observation $X = (m_1, m_2, m_3, m_4)$ with class probabilities given by

$$\begin{aligned} p_1 &= (2+\theta)/4, \\ p_2 &= (1-\theta)/4, \\ p_3 &= (1-\theta)/4, \\ p_4 &= \theta/4, \end{aligned} \tag{4.2.3}$$

where $0 < \theta < 1$. The sample size is $n = \sum m_i$. The parameter θ is to be estimated from the observed frequencies (1997, 906, 904, 32) from a sample of size 3839. The log-likelihood function and its derivatives are given by

$$\ell(\theta) = m_1 \log(2+\theta) + (m_2+m_3)\log(1-\theta) + m_4 \log\theta, \tag{4.2.4}$$

$$\dot{\ell}(\theta) = \frac{m_1}{2+\theta} - \frac{m_2+m_3}{1-\theta} + \frac{m_4}{\theta}, \tag{4.2.5}$$

and

$$\ddot{\ell}(\theta) = -\left\{\frac{m_1}{(2+\theta)^2} + \frac{m_2+m_3}{(1-\theta)^2} + \frac{m_4}{\theta^2}\right\}. \tag{4.2.6}$$

Expression (4.2.5) can be rewritten as a rational function, the numerator of which is quadratic in θ. One of the roots is negative, so the other root is the one we seek. (Note that even though the score function is defined for $\theta < 0$, the log-likelihood function is not.) Although this equation can be solved explicitly, we shall use it to illustrate the iterative methods discussed above.

Because we have an explicit and relatively simple expression (4.2.6) for the derivative of the score function, Newton-Raphson is a logical candidate for the iterative method. For comparison, we shall also show how the method of scoring performs. Choosing a starting value is not difficult in this case; an unbiased estimator for θ is given by $\tilde{\theta} = (m_1 - m_2 - m_3 + m_4)/n = 0.05704611$. We shall start with the convergence criterion of a relative change in $\hat{\theta}$ of less than 10^{-6}, with the view that either the tolerance or the criterion may need to be changed if convergence is not achieved within a few iterations. Table 4.2.5 shows the results of applying the two methods. Using Newton-Raphson, the derivative of the score function at $\hat{\theta}$ is -27519.22288. This should be compared to the negative of the Fisher information at $\hat{\theta}$, which is 29336.52362. The standard errors from the two methods are 0.006028 and 0.005838, respectively.

Once again, the sensitivity of Newton-Raphson to choice of starting values is illustrated in Table 4.2.6, which shows what happens to Newton's method and to the scoring method with $\hat{\theta}_0 = 0.5$. One might be led to such a choice by simply noting that θ must be in $(0,1)$ and by taking

	Newton-Raphson		Scoring	
i	$\hat{\theta}_i$	$\dot{\ell}(\hat{\theta})$	$\hat{\theta}_i$	$\dot{\ell}(\hat{\theta})$
0	0.05704611	-387.74068038	0.05704611	-387.74068038
1	0.02562679	376.95646890	0.03698326	-33.88267279
2	0.03300085	80.19367817	0.03579085	-2.15720180
3	0.03552250	5.24850707	0.03571717	-0.13386352
4	0.03571138	0.02527096	0.03571260	-0.00829335
5	0.03571230	0.00000059	0.03571232	-0.00051375
6	0.03571230	-0.00000000	0.03571230	-0.00003183

TABLE 4.2.5 *Two methods for maximum likelihood estimation in a multinomial problem. The scoring method converges more rapidly at the start, but Newton-Raphson's quadratic convergence takes over in the last few iterations.*

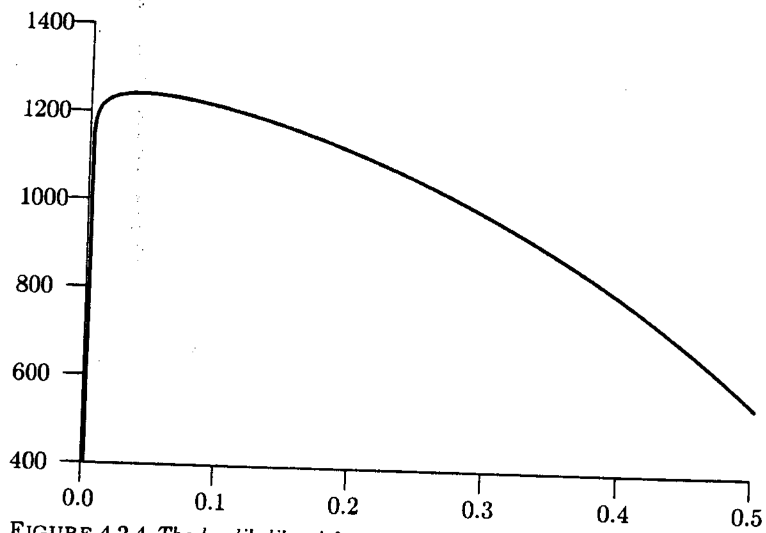

FIGURE 4.2.4 *The log-likelihood function $\ell(\theta)$ for the multinomial example. Note that starting values very close to zero or much larger than 0.1 are unreasonable and likely to cause difficulty for Newton methods.*

the midpoint of that interval. This "easy way out" of the starting-value problem leads to disaster for Newton's method, which converges to the wrong root! This difficulty is easily avoided by plotting the log likelihood before selecting a starting value, as we have done in Figure 4.2.4.

	Newton-Raphson		Scoring	
i	$\hat{\theta}_i$	$\dot{\ell}(\hat{\theta})$	$\hat{\theta}_i$	$\dot{\ell}(\hat{\theta})$
0	0.50000000	-2757.20000000	0.50000000	-2757.20000000
1	0.14134077	-948.94105740	0.05112008	-307.92073569
2	-0.06989831	-1114.89695350	0.03664009	-24.94396888
3	-0.19853655	-562.81128214	0.03576968	-1.57659006
4	-0.40797728	-109.58876500	0.03571586	-0.09778861
5	-0.46586259	-1.74894631	0.03571252	-0.00605820
6	-0.46681399	-0.00030477	0.03571232	-0.00037529
7	-0.46681415	-0.00000000	0.03571230	-0.00002325

TABLE 4.2.6 *Sensitivity of Newton's method to choice of starting values. Newton's method converges to the negative root of the likelihood equation. The scoring method converges to the correct root.*

4.2.6.2 Poisson regression

Thisted and Efron (1986) consider the problem of predicting the number of occurrences of rare words in a new passage of given length written by an author whose distribution of word frequencies is known. Thisted and Efron consider, in particular, whether a nine-stanza poem attributed to Shakespeare discovered in late 1985 by Gary Taylor, an American Shakespearean scholar, is consistent in patterns of word usage with that of the Bard. One approach to this question employs a Poisson regression model discussed at length by Cox and Hinkley (1974). The model discussed in this section has the drawback that fitted values of the Poisson parameter need not be positive; it has the virtue, however, of illustrating the solution of a single nonlinear equation, which is why we employ it here. A more suitable Poisson regression model is discussed in Section 4.3.5.2.

Let m_x be the number of words in the new passage used exactly x times in the known Shakespearean literature, and denote by M_j the jth "decade count," defined by $M_j = \sum_{x=10j}^{10j+9} m_x$. Under plausible assumptions, one can consider M_j to be a Poisson variate with mean $\alpha+\theta(X_j-\overline{X})$, where the X_j's are known constants with $\overline{X} = \sum X_j/n$, where α is a known constant which we shall take to equal $\overline{M} = \sum M_j/n = 11.8$, and where n is the number of decades being examined. Under the hypothesis of Shakespearean authorship, $\theta = 1$. We seek to estimate θ, and to obtain a standard error for the estimate as well. Table 4.2.7 gives the observed counts, and the regressors X_j derived from the Shakespearean literature, for the first $n = 10$ decades.

Let $x_j = X_j - \overline{X}$. Under the model given above, the log-likelihood

Using nlm() to maximize loglikelihood in the Rao example

```
% Set-up function to return negative loglikelihood

logl=function(t)
{
p1 = 2 * t[1] * t[2]
p2 = t[1] * (2 - t[1] - 2 * t[2])
p3 = t[2] * (2 - t[2] - 2 * t[1])
p4 = (1 - t[1] - t[2])^2
return( - (17 * log(p1) + 182 * log(p2) + 60 * log(p3) + 176 * log(p4)))
}

% Use nlm() to minimize negative loglikelihood

> nlm(logl,c(.263,.074),hessian=T,print.level=2)

iteration = 0
Step:
[1] 0 0
Parameter:
[1] 0.263 0.074
Function Value
[1] 494.6769
Gradient:
[1]  -25.48027 -237.68086

iteration = 1
Step:
[1] 0.002574994 0.024019643
Parameter:
[1] 0.26557499 0.09801964
Function Value
[1] 492.6586
Gradient:
[1]  9.635017 47.935126

iteration = 2
Step:
[1] -0.0007452647 -0.0040264713
Parameter:
[1] 0.26482973 0.09399317
Function Value
[1] 492.5393
Gradient:
[1] 2.386392 8.646591
```

```
iteration = 3
Step:
[1] -0.0002310889 -0.0008887750
Parameter:
[1] 0.2645986 0.0931044
Function Value
[1] 492.5354
Gradient:
[1]  0.5349342 -0.4784625

iteration = 4
Step:
[1] -4.312360e-05  4.180465e-05
Parameter:
[1] 0.2645555 0.0931462
Function Value
[1] 492.5353
Gradient:
[1]  0.4114774 -0.1036801

iteration = 5
Step:
[1] -8.846223e-05  2.977044e-05
Parameter:
[1] 0.26446705 0.09317597
Function Value
[1] 492.5353
Gradient:
[1] 0.09830296 0.10132959

iteration = 6
Step:
[1] -2.113552e-05 -4.583349e-06
Parameter:
[1] 0.26444592 0.09317139
Function Value
[1] 492.5353
Gradient:
[1] 0.01096532 0.03266155

iteration = 7
Step:
[1] -2.165882e-06 -2.800342e-06
Parameter:
[1] 0.26444375 0.09316859
Function Value
[1] 492.5353
Gradient:
[1] -0.0004739604  0.0021823894
```

```
iteration = 8
Parameter:
[1] 0.26444429 0.09316885
Function Value
[1] 492.5353
Gradient:
[1] -6.392156e-05  4.952284e-05

Relative gradient close to zero.
Current iterate is probably solution.

$minimum
[1] 492.5353

$estimate
[1] 0.26444429 0.09316885

$gradient
[1] -6.392156e-05  4.952284e-05

$hessian
         [,1]      [,2]
[1,] 3899.064  1068.172
[2,] 1068.172 10039.791

$code
[1] 1

$iterations
[1] 8

Warning messages:
1: NaNs produced in: log(x)
2: NaNs produced in: log(x)
3: NA/Inf replaced by maximum positive value
4: NaNs produced in: log(x)
5: NaNs produced in: log(x)
6: NA/Inf replaced by maximum positive value
7: NaNs produced in: log(x)
8: NaNs produced in: log(x)
9: NA/Inf replaced by maximum positive value
```

Generalized Linear Models: Logistic Regression

In generalized linear models (GLM's), a random variable y_i from a distribution that is a member of the *scaled exponential family* is modelled as a function of a dependent variable x_i. This family is of the form

$$f(y|\theta) = \exp\{[y\theta - b(\theta)]/a(\phi) + c(y,\theta)\}$$

where θ is called the natural (or canonical) parameter and ϕ is the scale parameter. Two useful properties of this family are

- $E(Y) = b'(\theta)$
- $Var(Y) = b''(\theta(\phi)$

The "linear" part of the GLM comes from the fact that some function $g()$ of the mean $E(Y_i|x_i)$ is modelled as a linear function $\mathbf{x}^T\boldsymbol{\beta}$ i.e.

$$g(E(Y_i|x_i) = \mathbf{x}^T\boldsymbol{\beta}$$

This function $g()$ is called the link function and is dependent on the distribution of y_i. The logistic regression model is an example of a GLM. Suppose that $(y_i|x_i)\, i = 1,\ldots,n$ represent a random sample from the Binomial distribution with parameters n_i and p_i i.e., $\text{Bin}(n_i,p_i)$. Then

$$\begin{aligned} f(y|p) &= \binom{n}{y} p^y (1-p)^{n-y} \\ &= \binom{n}{y}(1-p)^n(\frac{p}{1-p})^y \\ &= \exp\{y\log(\frac{p}{1-p}) + n\log(1-p) + \log\binom{n}{y}\} \end{aligned}$$

Thus the natural parameter is $\theta = \log(\frac{p}{1-p})$, $a(\phi) = 1$, $b(\theta) = -n\log(1-p)$, and $c(y,\phi) = \log\binom{n}{y}$.

Logistic Regression Model: Let$y_i|x_i \sim$ Binomial (n_i, p_i) Then the model is:

$$\log\frac{p_i}{1-p_i} = \beta_0 + \beta_1 x_i, \qquad i = 1,\ldots,n$$

It can be derived from this model that $p_i = \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}$ and thus $1 - p_i = \frac{1}{1+e^{\beta_0+\beta_1 x_i}}$

The likelihood of $\mathbf{p}$ is:

$$L(\mathbf{p}|\mathbf{y}) = \prod_{i=1}^{k}\begin{pmatrix} n_i \\ y_i \end{pmatrix} p_i^{y_i}(1-p_i)^{n_i-y_i}$$

and the log likelihood is:

$$\ell(\mathbf{p}) = \sum_{i=1}^{k}\left[\log\begin{pmatrix} n_i \\ y_i \end{pmatrix} + y_i\log p_i + (n_i - y_i)\log(1-p_i)\right]$$

where p_i are as defined above. There are two possible approaches for deriving the gradient and the Hessian. First, one can substitute p_i in ℓ above and obtain the log likelihood as a function of $\boldsymbol{\beta}$:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{k} \left[\log \begin{pmatrix} n_i \\ y_i \end{pmatrix} - n_i log(1 + \exp(\beta_0 + \beta_1 x_i)) + y_i(\beta_0 + \beta_1 x_i) \right]$$

and then calculate the gradient and the Hessian of $\ell(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ directly i.e. compute $\frac{\partial \ell}{\partial \beta_0}, \frac{\partial \ell}{\partial \beta_1}, \frac{\partial^2 \ell}{\partial \beta_0 \partial \beta_1}, \ldots$ etc., directly or use the chain rule $\frac{\partial \ell}{\partial \beta_j} = \frac{\partial \ell}{\partial p_i} \cdot \frac{\partial p_i}{\partial \beta_j}$, $j = 0, 1$ to calculate the partials from the original model as follows:

$$\log \tfrac{p_i}{1-p_i} = \beta_0 + \beta_1 x_i$$

$$\log p_i - \log(1 - p_i) = \beta_0 + \beta_1 x_i$$

$$\left(\tfrac{1}{p_i} + \tfrac{1}{1-p_i}\right) \partial p_i = \partial \beta_0 \quad \text{giving } \tfrac{\partial p_i}{\partial \beta_0} = p_i(1 - p_i)$$

$$\left(\tfrac{1}{p_i} + \tfrac{1}{1-p_i}\right) \partial p_i = x_i \partial \beta_1 \quad \text{giving } \tfrac{\partial p_i}{\partial \beta_1} = p_i(1 - p_i)$$

So, after substitution, it follows that:

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta_0} &= \sum_{i=1}^{k} \left\{ \frac{y_i}{p_i} - \frac{(n_i - y_i)}{1 - p_i} \right\} \frac{\partial p_i}{\partial \beta_0} \\
&= \sum_{i=1}^{k} \left\{ \frac{y_i}{p_i} - \frac{(n_i - y_i)}{1 - p_i} \right\} p_i(1 - p_i) \\
&= \sum_{i=1}^{k} (y_i - n_i p_i) \\
\frac{\partial \ell}{\partial \beta_1} &= \sum_{i=1}^{k} \left\{ \frac{y_i}{p_i} - \frac{(n_i - y_i)}{1 - p_i} \right\} p_i(1 - p_i) x_i \\
&= \sum_{i=1}^{k} x_i (y_i - n_i p_i) \\
\frac{\partial^2 \ell}{\partial \beta_0^2} &= -\sum_{i=1}^{k} n_i \frac{\partial p_i}{\partial \beta_0} = -\sum_{i=1}^{k} n_i p_i (1 - p_i) \\
\frac{\partial^2 \ell}{\partial \beta_0 \partial \beta_1} &= -\sum_{i=1}^{k} n_i p_i (1 - p_i) x_i \\
\frac{\partial^2 \ell}{\partial \beta_1^2} &= -\sum_{i=1}^{k} n_i x_i p_i (1 - p_i) x_i \\
&= -\sum_{i=1}^{k} n_i p_i (1 - p_i) x_i^2
\end{aligned}$$

In the following implementations of Newton-Raphson, a negative sign is inserted in front of the log likelihood, the gradient, and the hessian, as these routines are constructed for minimizing nonlinear functions.

Logistic Regression using an R function I wrote for implementing Newton-Raphson using analytical derivatives

```
Starting Values:  -3.9 0.64

Current Value of Log Likelihood:  6.278677

Current Estimates at Iteration  1 :
 Beta     =  -3.373257 0.6232912
 Gradient =  0.1055660 -0.1294950
 Hessian  =  6.178139 32.31268 32.31268 200.8371
Current Value of Log Likelihood:  5.754778

Current Estimates at Iteration  2 :
 Beta     =  -3.502322 0.6447013
 Gradient =  0.001067698 -0.01121716
 Hessian  =  6.009407 31.39635 31.39635 194.2825
Current Value of Log Likelihood:  5.746453

Current Estimates at Iteration  3 :
 Beta     =  -3.505401 0.6452565
 Gradient =  -3.942305e-07 -1.349876e-05
 Hessian  =  6.005219 31.37172 31.37172 194.0973
Current Value of Log Likelihood:  5.746449


Convergence Criterion of  1e-05  met for norm
of estimates.

Final Estimates Are:  -3.505402 0.6452569

Final Value of Log Likelihood:  5.746449

Value of Gradient at Convergence:
-3.942305e-07 -1.349876e-05

Value of Hessian at Convergence:

          [,1]      [,2]
[1,]  6.005219  31.37172
[2,] 31.371717 194.09727
```

Logistic Regression: Using the R function nlm() without specifying analytical derivatives. In this case nlm() will use numerical derivatives. First, define (the negative of) the log likelihood:

```
fun=function(b,xx,yy,nn)
{
        b0 = b[1]
        b1 = b[2]
        c=b0+b1*xx
        d=exp(c)
        p=d/(1+d)
        e=d/(1+d)^2
        f  = -sum(log(choose(nn,yy))-nn*log(1+d)+yy*c)
        return(f)
}
```

Data:

```
> x
[1] 10.2  7.7  5.1  3.8  2.6
> y
[1] 9 8 3 2 1
> n
[1] 10  9  6  8 10
```

> nlm(fun,c(-3.9,.64),hessian=T,print.level=2,xx=x,yy=y,nn=n)

iteration = 0

```
Step:
[1] 0 0
Parameter:
[1] -3.90  0.64
Function Value
[1] 6.278677
Gradient:
[1]  -2.504295 -13.933817
```

iteration = 1

```
Step:
[1] 0.01250237 0.06956279
Parameter:
[1] -3.8874976  0.7095628
Function Value
[1] 5.812131
Gradient:
[1] -0.2692995  0.3948808
```

```
iteration = 2
Step:
[1]  0.001281537 -0.001881983
Parameter:
[1] -3.8862161  0.7076808
Function Value
[1] 5.81129
Gradient:
[1] -0.3164593  0.1001753

iteration = 3
Step:
[1]  0.004598578 -0.002751733
Parameter:
[1] -3.881618  0.704929
Function Value
[1] 5.809923
Gradient:
[1] -0.3706596 -0.2551295

iteration = 4
Step:
[1]  0.013477313 -0.004877317
Parameter:
[1] -3.8681402  0.7000518
Function Value
[1] 5.806892
Gradient:
[1] -0.4379372 -0.7371624

iteration = 5
Step:
[1]  0.03893449 -0.01021976
Parameter:
[1] -3.829206  0.689832
Function Value
[1] 5.799401
Gradient:
[1] -0.5215346 -1.4545062

iteration = 6
Step:
[1]  0.08547996 -0.01754651
Parameter:
[1] -3.7437258  0.6722855
Function Value
[1] 5.784796
Gradient:
[1] -0.5641819 -2.1742337
```

iteration = 7

```
Step:
[1]  0.13604133 -0.02206089
Parameter:
[1] -3.6076844  0.6502246
Function Value
[1] 5.764175
Gradient:
[1] -0.4541274 -2.2419222
```

iteration = 8

```
Step:
[1]  0.10499324 -0.01132109
Parameter:
[1] -3.5026912  0.6389035
Function Value
[1] 5.749875
Gradient:
[1] -0.1843692 -1.1615320
```

iteration = 9

```
Step:
[1] 0.010799142 0.003049408
Parameter:
[1] -3.4918920  0.6419529
Function Value
[1] 5.746656
Gradient:
[1] -0.02263585 -0.21858969
```

iteration = 10

```
Step:
[1] -0.011132763  0.002875633
Parameter:
[1] -3.5030248  0.6448285
Function Value
[1] 5.746451
Gradient:
[1]  0.0008509039 -0.0084593736
```

iteration = 11

```
Step:
[1] -0.0022772925  0.0004153859
Parameter:
[1] -3.5053021  0.6452439
Function Value
[1] 5.746449
Gradient:
[1] 0.0002068596 0.0007312231
```

```
iteration = 12
Step:
[1] -9.381231e-05  1.163371e-05
Parameter:
[1] -3.5053959  0.6452556
Function Value
[1] 5.746449
Gradient:
[1] 8.464233e-06 4.623324e-05

iteration = 13
Parameter:
[1] -3.5054026  0.6452569
Function Value
[1] 5.746449
Gradient:
[1] -2.239637e-07 -1.805893e-06

Relative gradient close to zero.
Current iterate is probably solution.

$minimum
[1] 5.746449

$estimate
[1] -3.5054026  0.6452569

$gradient
[1] -2.239637e-07 -1.805893e-06

$hessian
          [,1]      [,2]
[1,]  6.005267  31.36796
[2,] 31.367958 194.02490

$code
[1] 1

$iterations
[1] 13
```

Note:

The convergence criterion was determined using the default parameter values for ndigit, gradtol, stepmax, gradtol, and iterlim.

Logistic Regression Example: Solution using user written R function for computing the Loglikelihood with "gradient" and "Hessian" attributes.

```
derfs4=function(b,xx,yy,nn)
{
        b0 = b[1]
        b1 = b[2]
        c=b0+b1*xx
        d=exp(c)
        p=d/(1+d)
        e=d/(1+d)^2
        f  = -sum(log(choose(nn,yy))-nn*log(1+d)+yy*c)
        attr(f,"gradient")=c(-sum(yy-nn*p),-sum(xx*(yy-nn*p)))
        attr(f,"hessian")=matrix(c(sum(nn*e),sum(nn*xx*e),sum(nn*xx*e),
                                                        sum(nn*xx^2*e)),2,2)
        return(f)
}

> x
[1] 10.2  7.7  5.1  3.8  2.6
> y
[1] 9 8 3 2 1
> n
[1] 10  9  6  8 10

> nlm(derfs4,c(-3.9,.64), hessian=T, iterlim=500, xx=x, yy=y, nn=n)
$minimum
[1] 5.746449

$estimate
[1] -3.5054619  0.6452664

$gradient
[1] -5.751195e-05 -1.252621e-05

$hessian
          [,1]      [,2]
[1,]  6.005193  31.36757
[2,] 31.367567 194.02219

$code
[1] 2

$iterations

[1] 338
```

Logistic Example: Solution using **nlm()** and symbolic derivatives obtained from **deriv3()**

The following statement returns a function **loglik(b, x , y, nn)** with “gradient” and “hessian” attributes

```
> loglik=deriv3(y~(nn*log(1+exp(b0+b1*x))  yy*(b0+b1*x)),c("b0","b1"),
 +   function(b,x,y,nn){})
```

Save the function in you working directory as a text file by the name **loglik.R**

```
> dump("loglik","loglik.R")
```

Now edit this file by inserting the hi-lited stuff:

```
loglik <-function (b, x, y, nn)
{
   b0 = b[1]
   b1 = b[2]
   .expr2 <- b0 + b1 * x
   .expr3 <- exp(.expr2)
   .expr4 <- 1 + .expr3
   .expr9 <- .expr3/.expr4
   .expr13 <- .expr4^2
   .expr17 <- .expr3 * x
   .expr18 <- .expr17/.expr4
   .value <- sum(nn * log(.expr4) - y * .expr2)
   .grad <- array(0, c(length(.value), 2), list(NULL, c("b0",
      "b1")))
   .hessian <- array(0, c(length(.value), 2, 2), list(NULL,
      c("b0", "b1"), c("b0", "b1")))
   .grad[, "b0"] <- sum(nn * .expr9 - y)
   .hessian[, "b0", "b0"] <- sum(nn * (.expr9 - .expr3 * .expr3/.expr13))
   .hessian[, "b0", "b1"] <- .hessian[, "b1", "b0"] <- sum(nn *
      (.expr18 - .expr3 * .expr17/.expr13))
   .grad[, "b1"] <- sum(nn * .expr18 - y * x)
   .hessian[, "b1", "b1"] <- sum(nn * (.expr17 * x/.expr4 - .expr17 *
      .expr17/.expr13))
   attr(.value, "gradient") <- .grad
   attr(.value, "hessian") <- .hessian
   .value
}
```

```
> xx
[1] 10.2  7.7  5.1  3.8  2.6
> yy
[1] 9 8 3 2 1
> nn
[1] 10  9  6  8 10
>
```

The number of iterations recquired for convergence by nlm() is determined by the quantities specified for the parameters ndigit, gradtol, stepmax, steptol, and iterlim. Here, using the defaults it required 338 iterations to converge.

```
> nlm(loglik,c(-3.9,.64), hessian=T, iterlim=500 ,x=xx, y=yy, nn=nn)
$minimum
[1] 18.87678

$estimate
[1] -3.5054619  0.6452664

$gradient
[1] -5.751168e-05 -1.252451e-05

$hessian
          [,1]      [,2]
[1,]  6.005383  31.35395
[2,] 31.353953 193.75950

$code
[1] 2

$iterations
[1] 338
```

Concentrated or Profile Likelihood Function

The most commonly used simplification in maximum likelihood estimation is the use of a *concentrated* or *profile* likelihood function. The parameter vector is first partitioned into two subsets $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (say, of dimensions p and q, respectively) and then the log-likelihood function is rewritten with two arguments, $\ell(\boldsymbol{\theta}) = \ell(\boldsymbol{\alpha},\ \boldsymbol{\beta})$. Now suppose that, for a given value of $\boldsymbol{\beta}$, the MLEs for the subset $\boldsymbol{\alpha}$ could be found as a function of $\boldsymbol{\beta}$; that is, $\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\alpha}}(\boldsymbol{\beta})$. Then the profile likelihood function is a function only of $\boldsymbol{\beta}$,

$$\ell_c(\boldsymbol{\beta}) = \ell\left[\hat{\boldsymbol{\alpha}}(\boldsymbol{\beta}),\ \boldsymbol{\beta}\right]. \tag{1}$$

Clearly, maximizing ℓ_c for $\boldsymbol{\beta}$ will maximize $\ell(\boldsymbol{\alpha},\ \boldsymbol{\beta})$ with respect to both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The main advantage is a reduction in the dimension of the optimization problem. The greatest simplifications occur when $\hat{\boldsymbol{\alpha}}$ does not functionally depend on $\boldsymbol{\beta}$.

For the first example, consider the basic problem $Y_i \sim$ IID$N(\mu,\ \sigma^2)$ for $i = 1, \ldots, n$. Then $\hat{\mu} = \bar{Y}$ and is independent of σ^2 and we have

$$\ell_c(\sigma) = -(n/2)\log\sigma^2 - \sum(Y_i - \bar{Y})^2/(2\sigma^2)$$

This needs to be maximized only with respect to σ^2, which simplifies the problem substantially, because it is a univariate problem. For another example, consider a modification of the simple linear regression model

$$y_i = \alpha_1 + \alpha_2 x_i^{\beta} + e_i, \qquad e_i \sim \text{ IID } N(0,\ \alpha_3). \tag{2}$$

Given $\beta = b$, the usual regression estimates can be found for $\alpha_1,\ \alpha_2,\ \alpha_3$, but these will all be explicit functions of b. In fact, $\ell_c(\beta)$ will depend only on an error sum of squares, since $\hat{\alpha}_3 = \text{SSE}/n$. Hence the concentrated likelihood function becomes simply

$$\ell_c(\beta) = \text{constant} - \frac{n}{2}\log\left(\frac{\text{SSE}(\beta)}{n}\right). \tag{3}$$

The gain is that the dimension of an unconstrained (or even constrained) search has been reduced from three (or four) dimensions to only one, and 1-dimensional searches are markedly simpler than those in any higher dimension.

Examples:

Bates and Watts (1988, p.41) gave an example of a rather simple nonlinear regression problem with two parameters:

$$y_i = \theta_1\left(1 - \exp\{-\theta_2 x_i\}\right) + e_i.$$

Given θ_2, the problem becomes regression through the origin. The estimator of θ_1 is simply

$$\hat{\theta}_1 = \sum_{i=1}^{n} z_i y_i \Big/ \sum_{i=1}^{n} z_i^2 \text{ where } z_i = 1 - \exp\{-\theta_2 x_i\},$$

and the concentrated likelihood function is as in (3) with

$$\text{SSE}(\theta_2) = \sum_{i=1}^{n}\left[y_i - \hat{\theta}_1(\theta_2)\right]^2.$$

Finding mle's of a two-parameter gamma distribution

$$f(y\,|\alpha,\ \beta) = \frac{1}{\beta^{\alpha}\,\Gamma(\alpha)}\, y^{\alpha-1}\exp\left(-y/\beta\right), \qquad \text{for } \alpha,\ \beta, \text{and, } y > 0$$

provides another example. For a sample size n, the log likelihood is

$$\ell(\alpha, \beta) = -n\alpha \log(\beta) - n \log \Gamma(\alpha) - (1-\alpha) \sum_{i=1}^{n} \log(y_i) - \sum_{i=1}^{n} \frac{y_i}{\beta}.$$

For a fixed value of α, the mle of β is easily found to be $\hat{\beta}(\alpha) = \sum_{i=1}^{n} y_i/n\alpha$. Thus the profile likelihood is

$$\ell_c(\alpha) = -n\alpha(\log(\sum_{i=1}^{n} y_i) - \log(n\alpha)) - n \log \Gamma(\alpha) - (1-\alpha) \sum_{i=1}^{n} \log(y_i) - n\alpha$$

A contour plot of the 2-dimensional log likelihood surface corresponding to the cardiac data, and the 1-dimensional plot of the profile likelihood are produced below.