Adaptive Robust Regression by Using a Nonlinear Regression Program

Mortaza Jamshidian [*]

September 18, 1999

Abstract

Robust regression procedures have received considerable attention in mathematical statistics literature. They, however, have not received nearly as much attention by practitioners performing data analysis. A contributing factor to this may be the lack of availability of these procedures in commonly used statistical software. In this paper we propose algorithms for obtaining parameter estimates and their asymptotic standard errors when fitting regression models to data assuming normal/independent errors. The algorithms proposed can be implemented in the commonly available nonlinear regression programs. We review a number of previously proposed algorithms. As we discuss, these require special code and are difficult to implement in a nonlinear regression program. Methods of implementing the proposed algorithms in SAS-NLIN is discussed. Specifically, the two applications of regression with the t and the slash family errors are discussed in detail. SAS NLIN and S-plus instructions are given for these two examples. Minor modification of these instructions can solve other problems at hand.

Key words: EM algorithm; GEM algorithm; Linear regression; Iterative reweighting; Normal/independent family; t distribution; SAS NLIN; Slash family, S-plus.

[*]Mortaza Jamshidian is Assistant Professor, Department of Statistics, University of Central Florida, Orlando, FL 32816-2370 (E-mail: `morij@pegasus.cc.ucf.edu`). The author would like to thank the referees for their helpful comments that led to a better presentation.

1 Introduction

It is well-known that statistical inference based on the classical normal theory assumption is sensitive to outliers and errors that come from a high-tailed distribution. Despite this fact and the considerable attention that robust estimation has received in the mathematical statistics literature, most practitioners continue to analyze their data based on the normal model. Lange, Little, and Taylor (1989) attribute the slow adaption of robust procedures to "the bewildering choice of alternative procedures and a lack of published applications to real data." Because data analysts often take the convenient route of using the available software to analyze their data and avoid the more costly alternative of writing special codes, we believe that the lack of wide availability of robust estimation procedures in the more popular statistical software also is a contributing factor to this slow adaption. To this end, in this paper we attempt to show how the widely available nonlinear regression programs may be used for adaptive robust regression.

Robust regression methods attempt to fit regression models to data so that the fit is less sensitive to the behavior on the tails of the errors and is reasonably stable when the errors come from a high-tailed distribution. M-estimation (Huber 1981), a generalized version of maximum likelihood (ML), is a popular robust procedure used in regression analysis. Lange and Sinsheimer (1993) argue for using adaptive robust regression methods that are based on ML estimation. Adaptive methods provide an attractive compromise between classical normal theory methods and modern robust methods (Lange and Sinsheimer 1993, Hogg 1974). In adaptive estimation, the distribution of the errors depends on tuning parameters that are estimated or set to achieve robust inference for statistical modeling of data sets involving errors with longer than normal tails.

The best studied adaptive methods are based on the distributional families of normal/independent (N/I) type (Lange and Sinsheimer 1993, Dempster, Laird, and Rubin 1977, 1980). Consider the regression model

$$y_i = f_i(\theta) + e_i \quad i = 1, \cdots, n$$

where y_i are the observed dependent variables, f_i are the theoretical response functions with a known form generally depending on a set of explanatory variables $\mathbf{x}_i$ and indexed by a set of parameters θ, and e_i denote the errors. In adaptive robust regression the classical normal theory assumption of $e_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$ is commonly replaced by the assumption that e_i are identically and independently distributed (iid) with a distribution from a N/I family. More specifically, let $e_i = r_i/\sqrt{u_i}$, where $r_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$ and $u_i \overset{iid}{\sim} U$ with U being a positive random variable. Then y_i are said to be N/I with location $f_i(\theta)$ and scale σ^2. Often the distribution of U depends on a single tuning parameter ν, disjoint from θ and σ^2. It is this class of the N/I family that we consider here.

A popular example from the N/I family, is the Student's t distribution. For this $U \sim \chi^2_\nu/\nu$, a fraction ν of the χ^2 with ν degrees of freedom. Another example is the slash distribution for which U has the density $h(u) = \nu u^{\nu-1}$ on $[0, 1]$ for $\nu > 0$. In both of these examples ν serves as a tuning parameter. Other examples include the contaminated normal, the power exponential, and the stable distribution (Lange and Sinsheimer 1993). A referee has also pointed out that another common method for dealing with outliers is to assume the e_i are generated from a mixture of two normal distributions, both with mean 0. One of the normals has variance σ^2 while the other has variance $k\sigma^2$ for some large k. This method, discussed in Verdinelli and Wasserman (1991), is also a member of the N/I family with the distribution of U being a dichotomous.

We focus on computational algorithms to obtain $\hat{\gamma} = (\hat{\theta}, \hat{\sigma^2}, \hat{\nu})$, the ML estimate of $\gamma = (\theta, \sigma^2, \nu)$ for the t and the slash families of distributions. Moreover, we give a method for obtaining $\widehat{\text{acov}}(\hat{\gamma})$, an estimate of the the asymptotic covariance matrix of $\hat{\gamma}$. Of course, the asymptotic standard error of parameters in $\hat{\gamma}$ can be obtained by taking the square root of the diagonal elements of $\widehat{\text{acov}}(\hat{\gamma})$. As we will see in Section 2, a number of algorithms have been proposed for both parameter and standard error estimation. These algorithms either have restrictions or to implement them requires special code (Jamshidian 1997). The algorithms that we propose here differ from the previously proposed algorithms in that they can be implemented in most available nonlinear regression programs, they produce standard errors, and have no restrictions.

In Section 2 we give a brief review of the previously proposed algorithms. In Section 3 we discuss parameter estimation. In Section 4 we discuss methods of obtaining standard errors. Finally, in Section 5 we discuss implementation of the proposed algorithms in a nonlinear regression program and give examples. For our examples we will use SAS NLIN. S-plus implementations are also given in the Appendix. The methods discussed, however, can be used in any other nonlinear regression package that has an iterative reweighting option, such as BMDP-3R or SPSS-NLR.

2 Review of algorithms

Dempster et al. (1980) proposed a method of iteratively reweighted least squares to estimate $\zeta = (\theta, \sigma^2)$ when $f_i(\theta)$ are linear, e_i are N/I, and ν is fixed. Among the N/I distributions, the t distribution has received special attention. Rubin (1983) gave details of implementing the Dempster et al. (1980) algorithm for the special case of the t distribution. Lange et al. (1989) described EM, Fisher-scoring, and quasi-Newton methods for ML estimation of the t model. In an attempt to accelerate EM, Liu and Rubin (1995) proposed a modified version of EM, called ECME, to estimate the regression model with t errors. Both the Lange et al. (1989) and the Liu and Rubin (1995) algorithms can handle nonlinear $f_i(\theta)$ and estimate ν as a free parameter. Finally, Lange and Sinsheimer (1993) described an EM algorithm for estimating the regression model with general N/I errors.

Except for the algorithm of Dempster et al. (1980) which is restricted to t with a fixed degree of freedom ν and linear $f_i(\theta)$, to use all of the other algorithms mentioned requires special code. Moreover Lange et al. (1989) and Lange and Sinsheimer (1993) proposed methods to obtain $\widehat{\text{acov}}(\hat{\gamma})$. As we will see in Section 4 these methods also require special code.

Jamshidian (1997) described a method for obtaining $\hat{\gamma}$ and $\widehat{\text{acov}}(\hat{\gamma})$ for the t model using BMDP-LE, a maximum likelihood program. BMDP-LE is not as widely available as nonlinear regression programs. Moreover, as explained by Jamshidian (1997), convergence of the maximization algorithm used in BMDP-LE is very sensitive to starting values. Jamshidian (1997) also described a method of using a nonlinear regression program for parameter estimation of the t model with a fixed degree of freedom ν. In addition to being restricted to t with a fixed-ν, his method does not produce standard errors.

3 Parameter Estimation

Let y_i have a N/I distribution with location $f_i(\theta)$, scale σ^2, and a tuning parameter ν, and let $\ell_i(\gamma)$ denote the log-density of y_i. We seek $\hat{\gamma}$ by maximizing the log-likelihood $\sum_{i=1}^{n} \ell_i(\gamma)$ with respect to γ. To do this we propose a generalized EM (GEM) algorithm (Dempster et al. 1977).

Since GEM is closely related to it, we begin by describing the EM algorithm of Dempster et al. (1977). Given a family of densities $g(\mathbf{y}|\gamma)$ and the observed values $\mathbf{y} = (y_1, \cdots, y_n)^T$ the EM algorithm begins by introducing a second family of densities $\tilde{g}(\mathbf{z}|\gamma)$, related to the first by the requirement that there is a function $\mathbf{h}$ such that $\mathbf{y} = \mathbf{h}(\mathbf{z})$ has density $g(\mathbf{y}|\gamma)$ if $\mathbf{z}$ has density $\tilde{g}(\mathbf{z}|\gamma)$. The vector $\mathbf{z}$ is usually called the *complete data* and $\mathbf{y}$ is called the *observed or incomplete data.* The algorithm begins by defining the Q-function $Q(\gamma', \gamma) = E\{\log \tilde{g}(\mathbf{z}|\gamma')|\mathbf{y}, \gamma\}$. This is called the E-step. Next the vector $\tilde{\gamma}$ that maximizes $Q(\gamma', \gamma)$ with respect to γ' is found. This is called the M-step. Replacing γ by $\tilde{\gamma}$ and repeating the E- and M- steps produces a sequence of values of γ that under appropriate conditions (Wu 1983, Dempster et al. 1977) converges to $\hat{\gamma}$. If maximization of Q in the M-step is replaced by the weaker condition of obtaining a $\tilde{\gamma}$ such that $Q(\tilde{\gamma}, \gamma) > Q(\gamma, \gamma)$, then the resulting algorithm is called GEM algorithm. Dempster et al. (1977) have shown that, like the EM algorithm, the GEM algorithm, under certain regularity conditions that are often satisfied for the problems discussed here, is globally convergent – that is, it will converge to a stationary point $\hat{\gamma}$ from almost any starting value.

Lange and Sinsheimer (1993) developed an EM algorithm to obtain $\hat{\gamma}$.

Their development emphasizes the comparison principle introduced by Dutter and Huber (Huber 1981). To derive their algorithm, we let $\mathbf{z}^T = (\mathbf{y}^T, \mathbf{u}^T)$ be the complete data, where $\mathbf{u} = (u_1, \cdots, u_n)^T$. Then it can be shown that $Q(\gamma', \gamma) = \sum_{i=1}^n Q_{1i}(\zeta', \gamma) + \sum_{i=1}^n Q_{2i}(\nu', \gamma)$, with

$$\begin{aligned} Q_{1i}(\zeta', \gamma) &= -\frac{1}{2}\left[E^*(u_i){\delta^2}_i(\zeta') + \log({\sigma^2}')\right] \\ Q_{2i}(\nu', \gamma) &= \frac{1}{2}E^*[\log(u_i)] + E^*\left\{\log[h_{\nu'}(u_i)]\right\}, \end{aligned} \tag{1}$$

where $E^*(\cdot)$ denotes the conditional expectation $E(\cdot|\mathbf{y}, \gamma)$, $\zeta' = (\theta', {\sigma^2}')$, ${\delta^2}_i(\zeta) = [y_i - f_i(\theta)]^2/\sigma^2$, and $h_\nu(u)$ denotes the density of U. Then given a set of starting values γ, to obtain $\hat{\gamma}$, the EM algorithm proceeds as follows:

Step 1. Solve the weighted regression problem

$$\min_{\theta'} S(\theta') \equiv \sum_{i=1}^n w_i(\gamma)[y_i - f_i(\theta')]^2$$

with weights $w_i(\gamma) = E^*(u_i)$ to obtain $\tilde{\theta}$.

Step 2. Update σ^2 using $\tilde{\sigma^2} = S(\tilde{\theta})/n$.

Step 3. Maximize $Q_2(\nu', \gamma) = \sum_{i=1}^n Q_{2i}(\nu', \gamma)$ with respect to ν' to obtain $\tilde{\nu}$.

Step 4. Check convergence. If convergence is not achieved, replace $\gamma = (\theta, \sigma^2, \nu)$ by $\tilde{\gamma} = (\tilde{\theta}, \tilde{\sigma^2}, \tilde{\nu})$ and go to Step 1.

Steps 1–3 make-up the M-step of the EM algorithm. The E-step consists of computing $w_i(\gamma) = E^*(u_i)$ and $E^*\left\{\log[h_{\nu'}(u_i)]\right\}$. The former is needed in Step 1 and the latter is needed in Step 3.

When $f_i(\theta)$ is nonlinear in θ, Step 1 is iterative. Step 3 is also generally iterative. These iterative steps within the iterative EM algorithm make it difficult, and often prohibitively difficult, to implement EM in a nonlinear regression program.

To avoid iteration in Step 1, we propose obtaining a point $\tilde{\theta}$ such that $S(\tilde{\theta}) < S(\theta)$. We then use this $\tilde{\theta}$ in Step 2 to obtain $\tilde{\sigma^2}$. The resulting algorithm is a GEM algorithm. To obtain such $\tilde{\theta}$ we propose using one step of the Gauss-Newton algorithm with step-halving. That is, we let

$$\tilde{\theta} = \theta + \left(\frac{1}{2}\right)^m \left[\frac{d\mathbf{f}^T}{d\theta} W \frac{d\mathbf{f}}{d\theta}\right]^{-1} \frac{d\mathbf{f}^T}{d\theta} W[\mathbf{y} - \mathbf{f}(\theta)], \tag{2}$$

where $\mathbf{f}(\theta) = [f_1(\theta), \cdots, f_n(\theta)]^T$, $d\mathbf{f}/d\theta$ denotes the Jacobian matrix of $\mathbf{f}(\theta)$, and W is a diagonal matrix with diagonal elements $w_i(\gamma)$. There is an $m \geq 0$ for which $S(\tilde{\theta}) < S(\theta)$. We choose the smallest such integer m. The choice of a Gauss-Newton step is made because of its often effectiveness in decreasing $S(\theta)$ and its wide availability in statistical software packages such as SAS, BMDP, SPSS, and S-plus. There are problems in which the Gauss-Newton Steps are not effective. These are mainly the problems referred to as large residual problems (see e.g., Seber and Wild, 1988, page 627). For such problems one may use other available options such as Levenberg-Marquardt method.

As noted, Step 3 is also generally iterative. This is because $Q_2(\nu', \gamma)$ can be a general nonlinear function of ν' and so its maximization may require iterative methods. Since $Q_2(\nu', \gamma)$ is model-specific, depending on the choice of U, we cannot give a general method of avoiding iterations in Step 3, as we did for Step 1. We will give specifics for the two examples of t and slash families. It turns out that for the slash family $\tilde{\nu}$ can be obtained in one step (Lange and Sinsheimer, 1993). Obtaining $\tilde{\nu}$ for the t, however, requires iteration. In Section 3.1 we give a one step method of obtaining $\tilde{\nu}$ to a good approximation and describe application of the GEM algorithm to the t model. In Section 3.2 details of the application of the GEM algorithm to the slash family are given.

3.1 A GEM algorithm for the t family

Assume $y_i \overset{iid}{\sim} t$ with location $f_i(\theta)$, scale σ^2, and degrees of freedom ν. Lange et al. (1989) show that

$$w_i(\gamma) = \frac{\nu + 1}{\nu + {\delta^2}_i(\zeta)}.$$

These weights are used in Step 1 of the GEM algorithm to obtain $\tilde{\theta}$.

The quantity $Q_{2i}(\nu', \gamma)$, needed in Step 3, is

$$Q_{2i}(\nu', \gamma) = -\frac{v_i(\gamma)}{2} + \left\{ \frac{\nu'}{2} [\log(\nu'/2) + v_i(\gamma) - w_i(\gamma)] - \log \Gamma(\nu'/2) \right\},$$

where $v_i(\gamma) \equiv E^*[\log(u_i)]$ and

$$v_i(\gamma) = \Psi[(\nu + 1)/2] + \log[2w_i(\gamma)/(\nu + 1)]$$

with $\Gamma(\cdot)$ and $\Psi(\cdot)$ denote the gamma and the digamma functions. Thus maximization of $Q_2(\nu', \gamma)$, required in Step 3, for the t model requires solving the equation

$$\log(\tilde{\nu}/2) - \Psi(\tilde{\nu}/2) + C = 0 \tag{3}$$

for $\tilde{\nu}$, where $C = 1 + (1/n) \sum_{i=1}^{n} [v_i(\gamma) - w_i(\gamma)]$. To solve (3), an iterative method is required. To avoid iterations we propose utilizing the approximation

$$\log(\tilde{\nu}/2) - \Psi(\tilde{\nu}/2) \approx \frac{1}{\tilde{\nu}} + \frac{1}{3\tilde{\nu}^2}. \tag{4}$$

This approximation has a maximum absolute error of $1/(120\tilde{\nu}^4)$ which makes it accurate to at least two significant digits when $\tilde{\nu} > 1$. This accuracy is sufficient for most practical problems. Replacing (4) in (3) leads to solving a quadratic equation for $\tilde{\nu}$ with its positive solution being

$$\tilde{\nu} = \frac{-3 - \sqrt{9 - 12C}}{6C}. \tag{5}$$

It can be shown that $C < 0$, and hence $\tilde{\nu}$ in (5) is positive.

3.2 A GEM algorithm for the slash family

For the slash family, the weights $w_i(\gamma)$ required in Step 1 are not as simple as those for the t distribution. These are

$$w_i(\gamma) = \left[\frac{2\nu + 1}{\delta^2{}_i(\zeta)} \right] \left\{ \frac{\Gamma[\delta^2{}_i(\zeta)/2, \nu + 3/2]}{\Gamma[\delta^2{}_i(\zeta)/2, \nu + 1/2]} \right\}$$

where $\Gamma(x, \nu)$ is the value of the gamma cumulative distribution function with parameter ν at x (Lange and Sinsheimer 1993).

Replacing the density $h_\nu(u) = \nu u^{\nu-1}$ of U in (1), it can be shown that up to a value not depending on ν'

$$Q_{2i}(\nu', \gamma) = \log(\nu') + \nu' v_i(\gamma)$$

where again $v_i(\gamma) = E^*[\log(u_i)]$ and

$$v_i(\gamma) = \Psi(\nu + 1/2) - \log[\delta^2{}_i(\zeta)/2] + \frac{\partial\Gamma[\delta^2{}_i(\zeta)/2, \nu + 1/2]/\partial\nu}{\Gamma[\delta^2{}_i(\zeta)/2, \nu + 1/2]}.$$

Thus the value $\tilde{\nu}$ that maximizes $Q_2(\nu', \gamma)$ in Step 3 is

$$\tilde{\nu} = -n/\sum_{i=1}^{n} v_i(\gamma).$$

To compute $\partial\Gamma[\delta^2{}_i(\zeta)/2, \nu + 1/2]/\partial\nu$, we propose using the central difference approximation $\{\Gamma[\delta^2{}_i(\zeta)/2, \nu_1 + \epsilon] - \Gamma[\delta^2{}_i(\zeta)/2, \nu_1 - \epsilon]\}/(2\epsilon)$ with $\nu_1 = \nu + 1/2$ and $\epsilon = 10^{-5} \times \max(1, \nu)$.

4 Standard Error Estimation

Both Lange et al. (1989) and Lange and Sinsheimer (1993) proposed using the inverse of the expected information matrix $\mathcal{I}^{-1}(\hat{\gamma})$ to obtain $\widehat{\text{acov}}(\hat{\gamma})$. Lange and Sinsheimer (1993) gave computational formulas for $\mathcal{I}(\hat{\gamma})$ for the N/I family and Lange et al. (1989) gave that for the special case of the t family. Like their parameter estimation algorithms, to use these computation formulae requires special code.

Consider $J(\gamma) = [d\ell(\gamma)/d\gamma)]^T[d\ell(\gamma)/d\gamma)]$, where $\ell(\gamma) = [\ell_1(\gamma), \cdots, \ell_n(\gamma)]^T$ and $d\ell(\gamma)/d\gamma$ denotes the Jacobian of $\ell(\gamma)$. $J(\hat{\gamma})$ is referred to as the empirical information. We propose using $J^{-1}(\hat{\gamma})$ to obtain $\widehat{\text{acov}}(\hat{\gamma})$. Our proposal to use $J^{-1}(\hat{\gamma})$ is motivated by the following facts: (i) $J(\hat{\gamma})$ is asymptotically equivalent to $\mathcal{I}(\hat{\gamma})$ in the sense that $[J(\hat{\gamma}) - \mathcal{I}(\hat{\gamma})]/n \to 0$ as $n \to \infty$ whenever the central limit theorem applies to the sequence $[\partial\ell_i(\gamma)/\partial\gamma)]^T[\partial\ell_i(\gamma)/\partial\gamma)]$. (ii) $J(\hat{\gamma})$ arises naturally in the context of the Gauss-Newton method, used in Step 1 of our GEM algorithm. (iii) Computation of $J(\hat{\gamma})$, as we will see shortly, is closely related to the EM Q-function.

To compute $J(\hat{\gamma})$ requires formulae for the partial derivatives $\partial \ell_i(\gamma)/\partial\gamma$. These formulas are related to the EM Q-function by the Fisher (1925) identity $\partial \ell_i(\gamma)/\partial\gamma = \dot{Q}_i(\gamma,\gamma)$, where $\dot{Q}_i(\gamma',\gamma)$ denotes the derivative of $Q_i(\gamma',\gamma)$ with respect to its first argument. Using this identity we obtain

$$\frac{\partial \ell_i(\gamma)}{\partial\theta} = \frac{1}{\sigma^2}\left\{E^*(u_i)\left[y_i - f_i(\theta)\right]\right\}\left[\frac{\partial f_i(\theta)}{\partial\theta}\right] \tag{6}$$

$$\frac{\partial \ell_i(\gamma)}{\partial\sigma^2} = \frac{1}{2\sigma^2}\left[E^*(u_i){\delta^2}_i(\zeta) - 1\right] \tag{7}$$

$$\frac{\partial \ell_i(\gamma)}{\partial\nu} = \dot{Q}_{2i}(\nu,\gamma). \tag{8}$$

Equations (6)-(8) depend on $w_i(\gamma) = E^*(u_i)$ and $\partial f_i(\theta)/\partial\theta$ both of which are used in Step 1 of our GEM algorithm. Also the components required to compute $\dot{Q}_{2i}(\nu,\gamma)$ are often those that are used in Step 3. These make obtaining $\widehat{\text{acov}}(\hat{\gamma})$ by using $J(\hat{\gamma})$ attractive since its required computational components are available from the parameter estimation part.

5 Using a nonlinear regression program

In this section we discuss implementation of the GEM algorithm and computation of $\widehat{\text{acov}}(\hat{\gamma})$, discussed in Sections 3 and 4, in a nonlinear regression program. We will use SAS NLIN. But, as mentioned earlier, the techniques that will be discussed can be used in most nonlinear regression programs that have an iteratively reweighting option.

5.1 Parameter estimation

A set of starting values are required to start the GEM iteration process. We have used ordinary least squares estimates for θ, its corresponding residual mean squares for σ^2, and $\nu = 4$. This choice of starting values has worked well in the examples reported here and in most other examples that we have run.

To obtain $\tilde{\theta}$ in Step 1, we proposed using the Gauss-Newton step (2). SAS-NLIN uses (2) by default, starting with $m = 0$. It continues adding 1 to m as long as $S(\tilde{\theta}) \geq S(\theta)$. This process is called step-halving. For the GEM algorithm it is required that the weights W remain constant throughout the step-halving process. Unfortunately when variable weights are used, as is the case here, SAS and most other nonlinear regression programs vary W in the step-halving process. To remedy this problem, like in many other iteratively reweighted least squares applications, we turn off the step-halving. When $f_i(\theta)$ is linear in θ, step halving is not required and in effect turning it off does not alter the GEM algorithm. However if $f_i(\theta)$ is nonlinear, to turn off the step halving makes the resulting algorithm not to be in general globally convergent. So in these cases it's advisable to check the iteration process for convergence. If nonconvergence results, one may try different starting values. We have not had any problems with nonconvergence, however. In our experience the starting values described above have almost always resulted in convergence.

In Step 2 the value of the weighted sum of squares $S(\tilde{\theta})$ is required. This value is generally available in nonlinear regression programs. In SAS, it is assigned to a variable named `_SSE_`.

As mentioned, the required computations in Step 3 depend on the choice of the N/I distribution. In Sections 3.1 and 3.2 we discussed methods of obtaining $\tilde{\nu}$ for the t and the slash family. In both cases this requires computation of $\sum_{i=1}^{n} v_i(\gamma)$ and in the case of t the quantity $\sum_{i=1}^{n} w_i(\gamma)$ is additionally needed. The methods for this computation are discussed in more details in Section 5.3.

5.2 Standard error estimation

In nonlinear regression programs commonly

$$\widehat{\text{acov}}(\hat{\gamma}) = RMS\{[dF(\gamma)/d\gamma]^T[dF(\gamma)/d\gamma]\}^{-1}, \tag{9}$$

where RMS is the residual mean squares, $F(\gamma) = [F_1(\gamma), \cdots, F_n(\gamma)]^T$ is the response function, and $dF(\gamma)/d\gamma$ is the Jacobian of F. This and the discussion of Section 4 suggest using the following method to obtain $J^{-1}(\hat{\gamma})$ in a nonlinear regression run: Set (i) starting values to $\hat{\gamma}$, (ii) the response function $F(\gamma)$ to $\ell(\gamma)$, (iii) the dependent variable equal to $\ell(\gamma)$, and (iv)

$RMS = 1$. Since both the response variable and the response function are set to the same value the Gauss-Newton step will be zero in the first iteration and thus the starting parameter values $\hat{\gamma}$ will not change. In fact since there is no change in the parameter estimates the program will stop after the first iteration. But, this trick will result in the program producing the appropriate standard errors in the process.

Instructions (ii) and (iii), given above, require specification of $\ell_i(\gamma)$ that might not be available in the EM setting. Note, however, that only the derivatives of $\ell_i(\gamma)$ are needed in this run. Since, by the Fisher's identity, $\partial \ell_i(\gamma)/\partial \gamma = \dot{Q}_i(\gamma, \gamma)$, we can use $Q_i(\gamma, \gamma) = Q_{1i}(\zeta, \gamma) + Q_{2i}(\nu, \gamma)$ in place of $\ell_i(\gamma)$ in (ii) and (iii). This avoids the need for specification of $\ell_i(\gamma)$. Finally, setting $RMS = 1$ is a standard practice in many iteratively reweighted least squares procedures. This is done in SAS using the PROC NLIN option "SIGSQ=1".

In the following subsection we give examples of regression with t and slash errors. In both cases our SAS instructions can be used to run any t or slash model with minimal modifications. The main modifications needed are to change the response function and its derivatives. The derivatives are not required if Version 6.12 of SAS is used. In this version SAS computes derivatives automatically, if they are not specified. This feature is also available for example in BMDP-3R.

5.3 Examples

Example 1: A t model example

Here we give SAS NLIN instructions to fit a linear regression model to the stack-loss data presented by Brownlee (1965). This data is often used for testing robust estimation procedures. Lange et al. (1989) used it to discuss linear regression with t errors. The stack loss data consists of measurements on the oxidation of ammonia to nitric acid during the operation of a plant. The data were collected over 21 consecutive days and the variables measured were air flow to the plant (AIR), cooling water inlet temperature (TEMP), acid concentration (ACID), and percent of ammonia lost (LOSS). The linear regression of LOSS on the variables AIRFLOW, TEMP, and ACID is considered.

An annotated SAS instructions is given in Figure 1. Instructions in [1] read in the data from a file named "stack.dat" and add a dummy case to the end of our data set. The dummy case is used in Step 3 of the GEM algorithm to signal the end of the data when passes are made through the data. Instead of using the last two lines of instructions in [1], one can also manually add a dummy case to the end of the data file. The values for the dummy case can be chosen arbitrarily.

In [2], we start the procedure NLIN. The `NOHALVE` option turns off the step-halving. The `CONVERGEPARM` sets the convergence criterion to be based on parameter estimates. The convergence criterion value of .0001 used gives about three to four digits of accuracy in the parameter estimates. The `MAXITER` option determines the maximum number of iterations allowed. If the algorithm does not converge within the default value of 50 a higher value should be set for `MAXITER`. Line two of [2] specifies the name of parameters and their starting values.

The first line in [3] sets the starting value for σ^2. The second line computes the update $\tilde{\sigma}^2$. This is Step 2 of the GEM algorithm. We have coded it before Step 1, since the initial value of σ^2 is needed in Step 1.

[4] implements Step 1 of the GEM algorithm, as the program passes through the $n = 21$ cases. The values of the response function $f_i(\theta)$, the dependent variable y_i, and weights $w_i(\gamma)$ are assigned to `F`, `Z`, and `W`, respectively. The variable C, required to obtain $\tilde{\nu}$, is accumulated at line 5 of [4]. Finally, the partial derivatives $\partial f_i(\theta)/\partial\theta$ are assigned to variables `DFXX`, where `XX` is replaced by a parameter name. $\partial f_i(\theta)/\partial\nu = 0$ and this is set in `DFNU`.

[5] implements Step 3 of the GEM algorithm. This step is executed when the program is operating on the added dummy case (case 22 here). Both the response function and the dependent variable are set to $\tilde{\nu}$, given by (5) and the weight is set to 1. This simply causes the value $\tilde{\nu}$ to be assigned to the variable `NU` before the start of the next iteration. The partial derivatives of

Figure 1: SAS instruction for fitting a linear regression model with t errors to the stack-loss data.

```
DATA;
   INFILE ”stack.dat” END=LAST;
   INPUT LOSS AIR TEMP ACID;                                  [1]
   IF LAST THEN DO;
      OUTPUT;  LOSS=.;  END;   OUTPUT;

PROC NLIN NOHALVE CONVERGEPARM=.0001 MAXITER=100;             [2]
   PARMS T1=-40 T2=.7 T3=1 T4=-.1 NU=4;

   IF _ITER_=-1 THEN SIGH=8;                                  [3]
      ELSE SIGH=_SSE_/21;

   IF _OBS_ ¡= 21 THEN DO;
      F=T1+T2*AIR+T3*TEMP+T4*ACID;
      Z=LOSS; R=Z-F;  DEL=(R*R)/SIGH;
      NU1=NU+1; W=NU1/(NU+DEL); V=DIGAMMA(NU1/2)+LOG(2*W/NU1) [4]
      C+(V-W+1)/21;
      DFT1=1; DFT2=AIR; DFT2=AIR; DFT3=TEMP; DFT4=ACID; DFNU=0;
      END;

   ELSE DO;
      Z=(-3-SQRT(9-12*C))/(6*C);
      NU=Z; F=NU; W=1;                                        [5]
      DFT1=0; DFT2=0; DFT3=0; DFT4=0; DFNU=1;
      END;

   MODEL Z=F; _WEIGHT_=W;
   DER.T1=DFT1; DER.T2=DFT2; DER.T3=DFT3; DER.T4=DFT4; DER.NU=DFNU;   [6]
   OUTPUT PARMS= T1H T2H T3H T4H NUH;
   ID SIGH V W DEL DFT1 DFT2 DFT3 DFT4;
```

the response function for this case are obvious and are stated on line 4 of [5].

Finally in [6] the `MODEL` statement is used to specify the overall model, putting together the model for the first $n = 21$ cases and the dummy case 22. The weights `W` are assigned to the SAS variable `_WEIGHT_`, and the derivatives are assigned to the SAS variables `DER.XX` where again `XX` is replaced by the name of the parameters. The `OUTPUT` statement instructs SAS to add the values of $\hat{\theta}$ and $\hat{\nu}$ to a SAS data file. Similarly the `ID` statement instructs SAS to write the variables whose names are included in that statement to a SAS data file. These values are then used to compute standard errors at convergence.

Instructions [1]–[6] will produce parameter estimates and a set of standard errors. While the parameter estimates are correct, the standard errors that will be printed are not. Instructions [7]–[9], shown in Figure 2, use the method discussed in Section 4 to produce the correct asymptotic standard errors.

The `SET` statement in the data step [7] brings the SAS data set which contains the variables in the `OUTPUT` and `ID` statements of [6] into our disposal. The second line of [7] deletes the dummy case from the SAS data set.

In [8] PROC NLIN is started and the option "`SIGSQ=1`" is set. The `PARMS` statement sets the starting values for the parameters to 0. It is really lines 3 and 4 of [8] that set the initial values to $\hat{\gamma}$ obtained from the instructions [1]–[6] of Figure 1.

Finally [9] defines Q_1, Q_2 the dependent and the response variables, as discussed in Section 4. The partial derivatives of the response for this run are given in [10]. These are based on the formulas (6)–(8).

To use the instructions [1]–[10] in solving other problems, one only needs to modify the statements underlined typeset Figures 1 and 2. These are mainly the variable names, the parameter names and their initial values, the response function, its derivatives if required, and the values 21 and 22 that are n and $n + 1$ for this example.

We ran instructions [1]-[10] in SAS. SAS required 57 iterations to meet the convergence criterion. The parameter estimates obtained were $\hat{\theta}_1 = -38.32$,

Figure 2: SAS instructions to produce standard errors for the stack-loss data example

```
DATA; SET;
   IF _N_=22 THEN DELETE;  [7]

PROC NLIN SIGSQ=1;
  PARMS T1=0 T2=0 T3=0 T4=0 NU=0 SIG=0;
  IF _ITER_=-1 THEN DO;                                          [8]
  T1=T1H; T2=T2H; T3=T3H; T4=T4H; NU=NUH; SIG=SIGH; END;

  F=T1+T2*AIR+T3*TEMP+T4*ACID;
  R=LOSS-F; DEL=(R*R)/SIG;
  Q1=-.5*(W*DEL+LOG(SIG));
  Q2=(NU/2)*(LOG(NU/2)+V-W)-LGAMMA(NU/2);  [9]
  Z=Q1+Q2;
  MODEL Z=Q1+Q2;

  WRS=W*R/SIG; DER.T1=WRS*DFT1; DER.T2=WRS*DFT2;
  DER.T3=WRS*DFT3; DER.T4=WRS*DFT4;                                    [10]
  DER.SIG=(W*DEL-1)/(2*SIG); DER.NU=.5*(LOG(NU/2)-DIGAMMA(NU/2)+V-W+1);
RUN;
```

$\hat{\theta}_2 = .8517$, $\hat{\theta}_3 = .4945$, $\hat{\theta}_4 = -.0734$, $\hat{\nu} = 1.186$, and $\hat{\sigma^2} = .9431$. All of the parameter estimates agree with those reported by Lange et al. (1989) to at least two significant digits. We will discuss the obtained standard error estimates in Section 5.4.

Example 2: A slash model example

Here we consider a data set that relates soil concentration of an agrochemical x_i to the growth response measured as log weight y_i of the plant lepidium. This data was analyzed by Racine-Poon (1988), where she considered the response function

$$f_i(\theta) = \begin{cases} \theta_1 / \left(1 + e^{\theta_2 + \theta_3 \ln(x_i) + \theta_4 [\ln(x_i)]^2}\right) & \text{if } x_i > 0 \\ \theta_1 & \text{if } x_i = 0 \end{cases} \tag{10}$$

Lange and Sinsheimer (1993) pointed out that this data set has two rather mild outliers among the 42 points observed. They used the slash model to fit the data. Figure 3 shows SAS NLIN instructions to obtain parameter estimates and their standard errors for this problem. These instructions are similar to those in Figures 1 and 2. The main differences are: (i) We have not included derivatives of the response functions in this example. As noted, SAS NLIN version 6.12 produces these automatically, if they are not specified. (ii) The model specific formulas for $w_i(\gamma)$, $v_i(\gamma)$, $\tilde{\nu}$, and $Q_2(\nu', \gamma)$ are specified as defined in Section 3.2. Again one can use these instructions to solve other problems by making appropriate changes to the underlined statements.

For this example SAS NLIN required 50 iterations to meet the convergence criterion. The resulting parameter estimates were $\hat{\theta}_1 = 2.367$, $\hat{\theta}_2 = -4.282$, $\hat{\theta}_3 = 2.660$, $\hat{\theta}_4 = -.4339$, $\hat{\nu} = 1.522$, and $\hat{\sigma^2} = .00223$. All the parameter estimates agree with those reported by Lange and Sinsheimer (1993). The standard errors estimates will be discussed in the next section.

5.4 Standard error estimates

In Section 4 we proposed using the empirical information to obtain asymptotic standard error estimates. This was motivated by the fact that the empirical information is asymptotically equivalent to the often used expected information. A simulation study is of interest to examine the behavior of

Figure 3: SAS instruction for fitting the slash model to the lepidium data.

```
DATA;
  INFILE "soil.dat" END=LAST;
  INPUT X Y;
  IF LAST THEN DO;
    OUTPUT; X=.; END;  OUTPUT;
PROC NLIN NOHALVE CONVERGEPARM=.0001 MAXITER=100;
  PARMS T1=2 T2=-4 T3=3 T4=-.4 NU=4;
  IF _ITER_=-1 THEN SIGH=.005;
    ELSE SIGH=_SSE_/42;
  IF _OBS_ <= 42 THEN DO;
    IF X=0 THEN F=T1;
    ELSE F=T1/(1+EXP(T2+T3*LOG(X)+T4*LOG(X)*LOG(X)));
    Z=LOG(Y); R=LOG(Y)-F;  DEL=(R*R)/SIGH; NU1=NU+1/2;
    W=((2*NU+1)/DEL)*PROBGAM(DEL/2,NU+3/2)/PROBGAM(DEL/2,NU1);
    HH=MAX(10**(-5)*NU1,10**(-5)); NUPH=NU1+HH; NUMH=NU1-HH;
    D=(PROBGAM(DEL/2,NUPH)-PROBGAM(DEL/2,NUMH))/(2*HH);
    V= DIGAMMA(NU1)-LOG(DEL/2)+D/PROBGAM(DEL/2,NU1);
    C+V; END;
  ELSE DO;
    Z=-42/C; NU=Z; F=NU; W=1;  END;
  MODEL Z=F; _WEIGHT_=W;
  OUTPUT PARMS=T1H T2H T3H T4H NUH;
  ID SIGH W V DEL;
DATA; SET;
  IF _N_=43 THEN DELETE;
PROC NLIN SIGSQ=1;
  PARMS T1=0 T2=0 T3=0 T4=0 NU=0 SIG=0;
  IF _ITER_=-1 THEN DO;
    T1=T1H; T2=T2H; T3=T3H; T4=T4H; NU=NUH; SIG=SIGH; END;
  IF X=0 THEN F=T1;
  ELSE F=T1/(1+EXP(T2+T3*LOG(X)+T4*LOG(X)*LOG(X)));
  R=LOG(Y)-F; DEL=(R*R)/SIG;
  Q1=-.5*(W*DEL+LOG(SIG));  Q2=LOG(NU)+NU*V;  Z=Q1+Q2;
  MODEL Z=Q1+Q2;
RUN;
```

both of these estimates in small samples in the context here. In this section we make a limited comparison of the information based standard error estimates to those obtained by bootstrap. We use the two examples given in the previous section and two new examples with artificial data and moderate sized sample sizes of 210. We refer to the new examples as Examples 3 and 4.

The data for Example 3 was generated using the model of Example 1 with population parameters $\theta = (-38, .9, .5, -.1)$, $\sigma^2 = 1$, and $\nu = 1$. The observations in the independent variables were replicated 10 times to generate 210 cases, and t noises were added to $f(\theta)$ to obtain y. The data for Example 4 was generated using the model of Example 2 with population parameters $\theta = (2, -4, 3, -.4)$, $\nu = 1.5$, and $\sigma = .01$. Again the X values were replicated 5 times to obtain 210 cases, and then slash distributed noises were added to $f(\theta)$ to obtain y.

Using SAS-NLIN, the parameter estimates obtained for Example 3 were $\hat{\theta} = (-40.2, 0.91, 0.47, -0.08)$, $\nu = .93$, and $\sigma^2 = 1.01$ and those for Example 4 were $\hat{\theta} = (2.37, -4.28, 2.66, -.43)$, $\nu = 1.52$, and $\sigma^2 = .002$. These estimate the population parameters well.

Table 1 shows a comparison of standard errors obtained for the four Examples. The expected information estimates for Examples 1 and 2 were obtained from Lange and Sinsheimer (1995) and Lange et al. (1989) and these were not computed for Examples 3 and 4. The empirical information estimates were obtained using SAS runs as described in Section 5.3. The bootstrap estimates were obtained using S-plus. They were based on bootstrap sample sizes of 150. For Examples 1 and 2 the expected and empirical informations estimates of asymptotic standard errors of the parameters agree to approximately one significant digit with the exception that for $\hat{\nu}$ in Example 2 the expected information estimate is .2 as compared to the empirical information estimate of 1.2 . Since the sample sizes are small, one does not expect great agreement between the information matrix estimates and the bootstrap estimates, but overall the empirical information estimates are closer to the

Table 1: Comparison of standard errors obtained via the expected information, Emprical information, and Bootstrap.

Method	Parameters					
	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\nu}$	$\hat{\sigma}^2$
			Example 1			
Expected Information	4.7	.054	.147	.063		
Empirical Information	5.6	.077	.198	.069	.79	.76
Bootstrap	9.9	.18	.38	.13	.88	.71
			Example 2			
Expected Information	.018	.357	.300	.0646	.234	.00048
Empirical Information	.019	.406	.341	.074	1.19	.0014
Bootstrap	.027	.755	.661	.146	.716	.00093
			Example 3			
Empirical Information	1.69	.019	.052	.022	.216	.078
Bootstrap	2.01	.020	.044	.026	.298	.121
			Example 4			
Empirical Information	.009	.203	.171	.0367	.288	.0003
Bootstrap	.0126	.228	.191	.041	.389	.0006

bootstrap estimates than the expected information estimates. In examples 3 and 4 where the sample sizes are larger, the bootstrap estimates and the empirical information estimates agree fairly well.

These limited examples give some comfort to using empirical information estimates of standard errors when the sample sizes are moderate to large. For small sample sizes the empirical information seem to produce slightly better estimates than the expected information but overall both of these seem to underestimate standard errors. Thus, in case of small sample sizes one may use the S-Plus scripts given in the Appendix to obtain bootstrap estimates of standard errors.

6 Appendix

In this section we give two S-plus scripts that will produce parameter and standard error estimates for Examples 1 and 2. By modifying the first 8 lines of the first script and the first 13 lines of the second script to specify information about a problem at hand these script can be used to solve other regression problems involving t and slash errors. The scripts were tested on S-Plus Student Edition 4.5.

Two S-plus functions are also given to compute the digamma function and the derivative of the gamma density with respect to ν. These need to be included and are called by the scripts.

An S-Plus script for Example 1.

```
VNAMES <- c("LOSS", "AIR", "TEMP", "ACID")
DEP <- c("LOSS")
PNAMES <- c("T1", "T2", "T3", "T4", "NU", "SIG")
initials <- c(-40, 0.7, 1, -0.1, 4, 8)
FN <- "T1 + T2 * AIR + T3 * TEMP + T4 * ACID"
FNW <- formula(sqrt(W) * Y ~ sqrt(W) * (T1 + T2 * AIR + T3 * TEMP + T4 * ACID))
MAXIT <- 250
TOL <- 1e-006
data <- data.frame(stack)
for(i in 1:length(VNAMES)) {
assign(VNAMES[i], data[, i])
}
assign("Y", eval(parse(text = DEP)))
nparm <- length(initials)
n <- length(eval(parse(text = VNAMES[1])))
it <- 0
updates <- initials
while(it < MAXIT) {
   it <- it + 1
   SIG <- initials[nparm]
   NU <- initials[(nparm - 1)]
   for(i in 1:nparm) {
     assign(PNAMES[i], initials[i])
   }
   fn <- eval(parse(text = FN))
   del <- ((Y - fn)^2)/SIG
   W <- (NU + 1)/(NU + del)
   for(i in 1:(nparm - 2)) {
      param(data, PNAMES[i]) <- initials[i]
   }
   nlout <- nls(FNW, data)
   for(i in 1:(nparm - 2)) {
      updates[i] <- nlout$parameter[i]
   }
   rss <- sum(nlout$residuals^2)
   updates[nparm] <- rss/n
   v <- digamma((NU + 1)/2) + log((2 * W)/(NU + 1))
   C <- 1 + mean(v - W)
   updates[(nparm - 1)] <- (-3 - sqrt(9 - 12 * C))/(6 * C)
   tmp <- abs(initials) * (abs(initials) > 1) + (abs(initials) < 1)
   relerr <- max(abs(initials - updates)/tmp)
   initials <- updates
   if(relerr < TOL)
      break
}
tmp <- eval(deriv(parse(text=FN),PNAMES[1:(nparm-2)]))
tmp <- attr(tmp, "gradient")
J <- (W*(Y-fn)/SIG)*tmp
tmp <- .5*(log(NU/2)+v-W+1-digamma(NU/2))
J <- cbind(J,tmp)
tmp <- .5*(W*del-1)/SIG
J <- cbind(J,tmp)
J <- diag(ginverse(t(J)%*%J))
Estimates <- format.default(updates,digits=6,scientific=c(-3,3))
Std.Errors <- format.default(sqrt(J),,digits=4,scientific=c(-3,3))
tmp <- cbind(PNAMES,"   ",Estimates,"  ",Std.Errors)
print.table(tmp)
```

An S-Plus script for Example 2.

```
VNAMES <- c("X", "LOGY")
DEP <- c("LOGY")
PNAMES <- c("T1", "T2", "T3", "T4", "NU", "SIG")
initials <- c(2, -4, 3, -0.4, 4, 0.005)
data <- data.frame(soil)
data[2] <- log(data[2])
xp <- (X>0)*1
x0 <- (X==0)*1
FN <- "xp*T1/(1+exp(T2+T3*log(X+x0)+T4*((log(X+x0)^2))))+x0*T1"
FNW <- formula(sqrt(W) * Y ~ sqrt(W) *(xp*T1/(1+exp(T2+T3*log(X+x0)
               +T4*((log(X+x0)^2))))+x0*T1) )
MAXIT <- 200
TOL <- 1e-006
for(i in 1:length(VNAMES)) {
```

```
assign(VNAMES[i], data[, i])
}
assign("Y", eval(parse(text = DEP)))
nparm <- length(initials)
n <- length(eval(parse(text = VNAMES[1])))
it <- 0
updates <- initials
while(it < MAXIT) {
   it <- it + 1
   SIG <- initials[nparm]
   NU <- initials[(nparm - 1)]
   for(i in 1:nparm) {
      assign(PNAMES[i], initials[i])
   }
   fn <- eval(parse(text = FN))
   del <- ((Y - fn)^2)/SIG
   W <- ((2*NU+1)/del)*(pgamma(del/2,NU+1.5)/pgamma(del/2,NU+.5))
   for(i in 1:(nparm - 2)) {
      param(data, PNAMES[i]) <- initials[i]
   }
   nlout <- nls(FNW, data)
   for(i in 1:(nparm - 2)) {
      updates[i] <- nlout$parameter[i]
   }
   rss <- sum(nlout$residuals^2)
   updates[nparm] <- rss/n
   v <- digamma(NU+.5)-log(del/2)+dpgamma(del/2,NU+.5)/pgamma(del/2,NU+.5)
   updates[(nparm - 1)] <- -1/mean(v)
   tmp <- abs(initials) * (abs(initials) > 1) + (abs(initials) < 1)
   relerr <- max(abs(initials - updates)/tmp)
   print(c(it,relerr))
   initials <- updates
   if(relerr < TOL)
      break
}
tmp <- eval(deriv(parse(text=FN),PNAMES[1:(nparm-2)]))
tmp <- attr(tmp, "gradient")
J <- (W*(Y-fn)/SIG)*tmp
tmp <- 1/NU+v
J <- cbind(J,tmp)
tmp <- .5*(W*del-1)/SIG
J <- cbind(J,tmp)
J <- diag(ginverse(t(J)%*%J))
```

```
Estimates <- format.default(updates,digits=6,scientific=c(-3,3))
Std.Errors <- format.default(sqrt(J),,digits=4,scientific=c(-3,3))
tmp <- cbind(PNAMES,"   ",Estimates,"  ",Std.Errors)
print.table(tmp)
```

```
"digamma"<-
function(x)
{
   eta <- 10^(-4)
   h <- eta * max(1, abs(x))
   lgm2 <- lgamma(x - 2 * h)
   lgm1 <- lgamma(x - h)
   lg2 <- lgamma(x + 2 * h)
   lg1 <- lgamma(x + h)
   y <- (lgm2 - 8 * lgm1 + 8 * lg1 - lg2)/(12 * h)
   return(y)
}
```

```
"dpgamma"<-
function(x,a)
{
   eta <- 10^(-4)
   h <- eta * max(1, abs(a))
   lgm2 <- pgamma(x,a - 2 * h)
   lgm1 <- pgamma(x,a - h)
   lg2 <- pgamma(x,a + 2 * h)
   lg1 <- pgamma(x,a + h)
   y <- (lgm2 - 8 * lgm1 + 8 * lg1 - lg2)/(12 * h)
   return(y)
}
```

References

Brownlee, K. A. (1965), *Statistical Theory and Methodology in Science and Engineering* (2nd ed.), Wiley, New York.

Dempster, A. P., Laird, N.M., and Rubin, D.B. (1977), "Maximum Likelihood Estimation From Incomplete Data Via the EM Algorithm (with Discussion)", *Journal of the Royal Statistical Society, Ser. B,* **39**, 1–38.

——— (1980) "Iteratively Reweighted Least Squares for Linear Regression When Errors are Normal/Independent Distributed," *in Multivariate Analysis V,* ed. P. R. Krishnaiah, North Holland, Amsterdam, pp. 35–57.

Fisher, R. A. (1925), "Theory of Statistical Estimation," *Proc. Camb. Phil. Soc.*, **22**, 700-725.

Hogg, R. V. (1974), "Adaptive Robust Procedures: A Partial Review and Some Suggestions for Future Applications and Theory (with Discussion)," *JASA,* **69**, 909-926.

Huber, P. J. (1981), *Robust Statistics,* John Wiley, New York.

Jamshidian, M. (1997), "t-Distribution Modeling Using the Available Statistical Software," *Computational Statistics and Data Analysis,* **25**, 181–216.

Lange, K. L., Little, R. J. A., and Taylor, J. M. G. (1989), "Robust Statistical Modeling Using the t Distribution," *JASA,* **84**, 881–890.

Lange, K. L. and Sinsheimer, J. S. (1993), "Normal/Independent Distributions and Their Applications in Robust Regression," *Journal of Computational and Graphical Statistics,* **2**, 175–198.

Liu, C. and Rubin, D. B. (1995), "ML Estimation of the t Distribution Using EM and Its Extensions, ECM and ECME," *Statistica Sinica*, **5** 19–39.

Racine-Poon, A. (1988), "A Bayesian Approach to Nonlinear Calibration Problems," *JASA,* **83**, 650–656.

Rubin, D. B. (1983), "Iteratively Reweighted Least Squares," In *Encyclopedia of Statistical Sciences*, Vol 4, John Wiley, New York, pp. 272–275).

Seber, G. A. F. and Wild, C. J. (1989), *Nonlinear Regression,* John Wiley, New York.

Verdinelli, I and Wasserman, L. (1991), "Baysian Analysis of Outlier Problems Using the Gibbs Sampler," *Statistics and Computing,* **1**, 105–117.

Wu, C. J. F. (1983), "On the Convergence Properties of the EM Algorithm," *The Annals of Statistics,* **11**, 95–103.