Calculating the Jacobean and Residuals of a Nonlinear Regression Model

Phil Spector
Statistical Computing Facility
UC Berkeley

1 The Model and Data

Consider the following nonlinear model, used to determine the effectiveness of different doses of a drug:

$$y = -\frac{b_0}{1 + (x/b_2)^{b_1}} \tag{1}$$

Here y represents the response to the drug and x represents the dose. Thus there is one independent variable (x), and three parameters (b_0, b_1, and b_2). The following table provides a small data set for the example:

x	y	x	y
0.1	0.1701	0.9	0.3201
0.2	0.2009	1.0	0.4140
0.3	0.2709	1.1	0.3677
0.4	0.2648	1.2	0.3476
0.5	0.3013	1.3	0.3656
0.6	0.4278	1.4	0.3879
0.7	0.3466	1.5	0.3649
0.8	0.2663		

2 Jacobean and Residuals

To calculate the Jacobean, we need to find the first partial derivatives of the residual from the nonlinear regression with respect to each of the three parameters. This can be easily done using `mathematica` (See Figure 1).

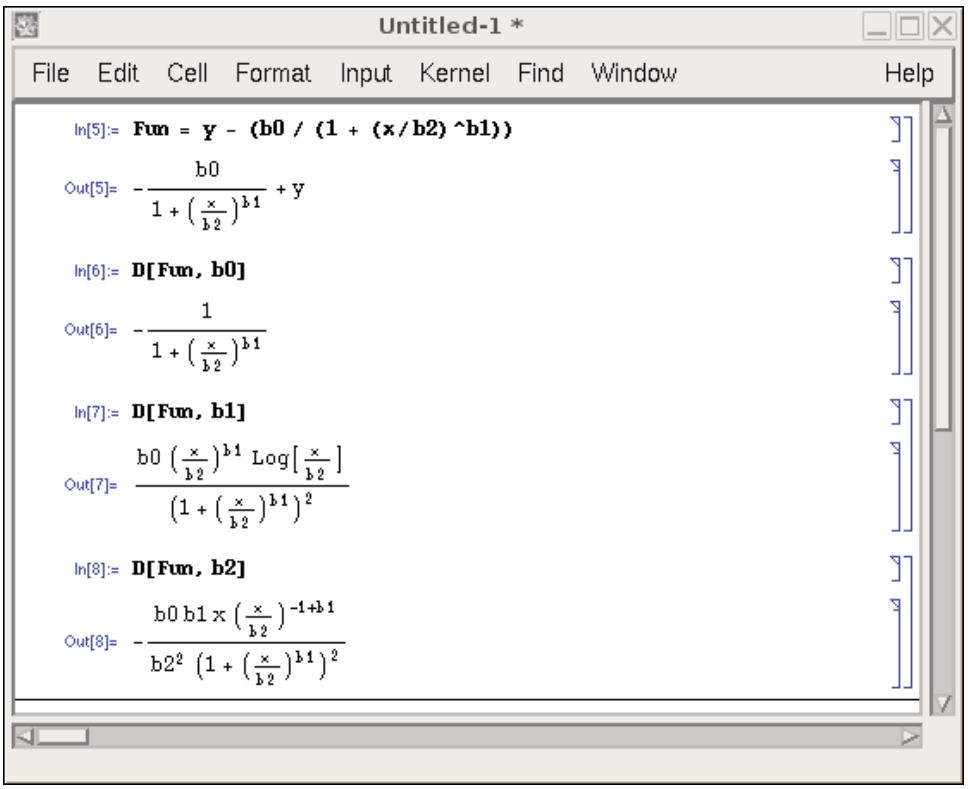

Figure 1: Mathematica Session to find Derivatives

Since there are 15 observations and three parameters, the Jacobean matrix will have dimensions 15×3. Each row of this matrix will represent the three partial derivatives evaluated at one individual's x values, i.e.

$$J = \begin{bmatrix} -\frac{1}{(1+(x_1/b_2)^{b_1})} & \frac{b_0(x_1/b_2)^{b_1}\log(x_1/b_2)}{(1+(x_1/b_2)^{b_1})^2} & -\frac{b_0 b_1 x_1 (x_1/b_2)^{(-1+b_1)}}{b_2^2(1+(x_1/b_2)^{b_1})^2} \\ \vdots & \vdots & \vdots \\ -\frac{1}{(1+(x_{15}/b_2)^{b_1})} & \frac{b_0(x_{15}/b_2)^{b_1}\log(x_{15}/b_2)}{(1+(x_{15}/b_2)^{b_1})^2} & -\frac{b_0 b_1 x_{15} (x_{15}/b_2)^{(-1+b_1)}}{b_2^2(1+(x_{15}/b_2)^{b_1})^2} \end{bmatrix} \tag{2}$$

The residuals are computed similarly:

$$f = \begin{bmatrix} y_1 - \left(\frac{b_0}{(1+(x_1/b_2)^{b_1})}\right) \\ \vdots \\ y_{15} - \left(\frac{b_0}{(1+(x_{15}/b_2)^{b_1})}\right) \end{bmatrix} \tag{3}$$

3 Iteratively Finding the Solution

The Gauss-Newton method uses the following formula to iteratively arrive at a set of parameter values that minimizes the sum of squared residuals:

$$\beta_{i+1} = \beta_i - (J(x, \beta_i)' J(x, \beta_i))^{-1} J(x, \beta_i)' f(x, y, \beta_i) \tag{4}$$

where β_i represents the vector (b_0, b_1, b_2) at the i-th iteration, $J(\cdot)$ is defined by equation 2 and $f(\cdot)$ is defined by equation 3. The process requires an initial value β_0 to get started.

3.1 Starting Values

One method for finding starting values is to perform a grid search. In R, the `expand.grid()` function makes this very easy. Assume that `x` contains the independent variable, and `y` the response variable, as shown in the earlier table. First, a function is defined which will calculate the objective function for the problem, that is, the sum of squared residuals from the non-linear fit

```
> ssrfun = function(b)sum((y-(b[1]/(1+(x/b[3])^b[2])))^2)
```

Next, construct a grid around some reasonable values, and evaluate the function at each of the grid points:

```
> points = expand.grid(b0=seq(0,1,l=10),b1=seq(-1,1,l=10),b2=seq(0,1,l=10))
> result = apply(points,1,ssrfun)
```

Finally, find the minimum value in `result`, along with the corresponding values from `points`:

```
> which(result == min(result))
205
205
> points[205,]
           b0 b1        b2
205 0.4444444 -1 0.2222222
```

3.2 Iterative Evaluation

To carry out the iterative evaluation in R, we need functions to calculate the Jacobean and residuals, given a set of parameter values:

```
> getjac = function(b){
+ cbind(-1/(1+(x/b[3])^b[2]),
+ (b[1]*(x/b[3])^b[2]*log(x/b[1])) / (1+(x/b[3])^b[2])^2,
+ -(b[1]*b[2]*x*(x/b[3])^(-1+b[2]))/(b[3]^2*(1+(x/b[3])^b[2]))^2)
+ }
> getres =  function(b)y-(b[1]/(1+(x/b[3])^b[2]))
```

Due to the similarity of the updating calculation to that used in linear regression, the `lm` function is used to calculate the necessary updates:

```
> b = c(.4,-1.0,.2)
> bold = c(0,0,0)
>
> while(sum(abs(b-bold)) > 1e-8){
+         bold = b
+         jac = getjac(b)
+         oldres = getres(b)
+         adj = lm(oldres ~ jac - 1)$coef
+         b = b - adj
+         res = sum(getres(b)^2)
+ }
>
> print(b)
      jac1       jac2       jac3
 0.4222882 -0.9774553  0.1741624
```

(In practice, it must be verified that the residual sum of squares is actually decreasing with each iterative adjustment. If not, step halving should be performed, *i.e.* try using one-half of the adjustment, then one-quarter and so on until the sum of squared residuals decreases.)

4 Verification using the `nls()` function in R

As a final check on the computation, we can fit the same model using the `nls()` function in R:

```
> nls(y~b[1]/(1 + (x/b[3])^b[2]),start=list(b=c(.4,-1.0,.2)))
Nonlinear regression model
  model:  y ~ b[1]/(1 + (x/b[3])^b[2])
   data:  parent.frame()
        b1         b2         b3
 0.4222878 -0.9774575  0.1741619
 residual sum-of-squares:  0.02341823
```