STA 410/2102, Spring 2003 — Assignment #2

Due at **start** of class on March 18. Worth 18% of the final mark.

Note that this assignment is to be done by each student individually. You may discuss it in general terms with other students, but the work you hand in should be your own.

In this assignment, you will write programs to find maximum likelihood estimates for the parameters of a logistic distribution, along with standard errors for these estimates.

The logistic distribution has a location parameter, μ, and a width parameter, ω. Its density function is

$$f(x) \;=\; \left[\omega\,(1+\exp((x-\mu)/\omega))\,(1+\exp(-(x-\mu)/\omega))\right]^{-1}$$

This is symmetrical around μ, and has a shape similar to that of the normal density function, but with somewhat heavier tails.

Suppose we have observed data x_1, x_2, ..., x_n, which we model as coming independently from a logistic distribution with some values for the parameters μ and ω. We wish to find the maximum likelihood estimates for these parameters, and also an estimate of the standard error for these estimates, obtained from the observed information matrix.

Part I:

For the first part of the assignment, you should assume that the value for ω is known to be one. You therefore need only find the maximum likelihood estimate for μ.

You should write a function that tries to find the MLE for μ given the data vector, x, and an initial guess for μ. The function should find the MLE using Newton-Raphson iteration, starting at this initial guess. The number of Newton-Raphson iterations to perform should be another argument of the function (you needn't try to determine automatically how many iterations are needed). The value returned by this function should be a list with two elements, called `mle` and `std.err`, which are the MLE found and the standard error for it. The standard error should be the square root of minus the reciprocal of the second derivative of the log likelihood, at the MLE. The function should also have a `debug` argument (defaulting to FALSE), which when set to TRUE causes the value of μ and the log likelihood to be printed at every iteration, so that you can see what is happening.

You should write your function in good style, using other functions (eg, for evaluating derivatives of the log likelihood) as appropriate. You should hand in a printout of your program, along with some tests of how well it works, and a brief discussion of when it seems to work, and of how many iterations it requires to get a good estimate when it does work. You should also propose a method of finding a good starting value for μ from the data.

Part II:

In this part, you will estimate both μ and ω from the data. You should again write a function that finds the MLE, using Newton-Raphson iteration, starting from initial values for μ and ω. The result returned should be a list with elements called `mle` and `std.err`, which will both be vectors of length two. The standard errors for the two parameters will be the square roots of minus the diagonal elements of the inverse of the matrix of second derivatives of the log likelihood.

You should hand in program printouts, tests, and discussion as for Part I.

Part III:

Finally, you should try solving Part I and Part II using R's built-in `nlm` function, which you will have to provide with a function to evaluate the log likelihood. (For this part, you just need to find the MLE, not standard errors.) Discuss how well `nlm` works, and how fast it is compared to your functions when the data vector is large. (The `system.time` function will be useful for this; look at the "user" time, or at the "elapsed" time, if your system doesn't support "user" time.)

Notes:

In order to solve Parts I and II, you will need to find the first and second derivatives of the log likelihood. This can be done by hand, but if you wish, you can use the symbolic differentiation facilities of R or another package (eg, maple or mathematica). The expresssions you use for these derivatives should be simplified to a reasonable extent, however, which these packages may not do. You should use the expressions for the derivatives directly in your functions. (Ie, your functions should not have to find these derivatives themselves, or do other sorts of operations that they would not have had to do if you had found the derivatives by hand.)

It is very likely that your initial expressions for the derivatives will be wrong. You should test them out by comparing the results to those found by taking small differences of the log likelihood. To check second derivatives, you can take small differences of the first derivatives (once you've verified that they are computed correctly).

You should test the functions (yours and `nlm`) using a wide variety of starting points, in order to get a good picture of when they work and when they don't. You should use the following data set in your tests (for both Parts I and II):

1.56 5.20 1.49 7.52 3.43 5.96 5.16 7.62 3.93 4.22

You should also use other data sets, as appropriate. You can generate n points from a logistic distribution with the following R statements:

```
u <- runif(n)
x <- mu + omega*log(u/(1-u))
```