Rose, Smith:

Symbolic Maximum Likelihood Estimation with Mathematica

Sonderforschungsbereich 386, Paper 141 (1999)

Online unter: http://epub.ub.uni-muenchen.de/

Projektpartner

Symbolic Maximum Likelihood Estimation with *Mathematica*

January 1999

COLIN ROSE
Theoretical Research Institute, Sydney, Australia

and MURRAY D. SMITH*
University of Sydney, Sydney, Australia

SUMMARY
Mathematica is a symbolic programming language that empowers the user to undertake complicated algebraic tasks. One such task is the derivation of maximum likelihood estimators, demonstrably an important topic in statistics at both the research and expository level. In this paper, a *Mathematica* package is provided that contains a function entitled **SuperLog**. This function utilises pattern-matching code that enhances *Mathematica'*s ability to simplify expressions involving the natural logarithm of a product of algebraic terms. This enhancement to *Mathematica*'s functionality can be of particular benefit for maximum likelihood estimation.

Keywords: Computer algebra systems; Estimate; Estimator; *Mathematica*; Symbolic maximum likelihood; Teaching.

* *Address for correspondence*: Department of Econometrics, University of Sydney, Sydney NSW, Australia 2006.

Email: ≪Murray.Smith@econ.usyd.edu.au≫

■ 1. Introduction

While statistical software has long been used for maximum likelihood (ML) estimation, the focus of attention has almost always been on obtaining ML estimates (a *numerical* problem), rather than on deriving ML estimators (a *symbolic* problem). Even after the introduction of powerful computer algebra systems, in the context of ML estimation, symbolic engines have largely only been used to solve numerical problems (see, for example, Currie, 1995, and Cook and Broemeling, 1995), or to derive large sample symbolic approximations to ML estimators (see Stafford and Andrews, 1993, and Stafford *et al.*, 1994). By contrast, we use a computer algebra system for the first time to derive *exact* symbolic ML estimators from first principles.

This paper shows how *Mathematica* (see Wolfram, 1996) can be extended to handle symbolic ML estimation, which is demonstrably an important topic in statistics, at both the research and expository level. The paper has five sections. Section 2 expands *Mathematica*'s programming language to handle symbolic ML estimation. Section 3 illustrates the approach with three simple expository examples, and is therefore well-suited for teaching purposes. Section 4 extends the analysis to more challenging material. Section 5 concludes. A *Mathematica* glossary and appendix follow.

■ 2. Extending *Mathematica*: the SMLE package

Consider the following simple problem: Let $(Y_1, ..., Y_n)$ denote a random sample of size n collected on $Y \sim \text{Poisson}(\theta)$, where parameter $\theta > 0$ is unknown - show that the mean of the random sample is the ML estimator of θ. We begin our answer in the usual way by inputting the likelihood function into *Mathematica*:

$$L = \prod_{i=1}^{n} \frac{e^{-\theta}\,\theta^{Y_i}}{Y_i!};$$

If we try to evaluate the log-likelihood:

```
Log[L]
```

$$\text{Log}\left[\prod_{i=1}^{n} \frac{E^{-\theta}\,\theta^{Y_i}}{Y_i!}\right]$$

... nothing happens, and for good reason! *Mathematica* will not normally 'convert' an expression such as **Log[a×b]** into **Log[a]+Log[b]** because symbols **a** and **b** could represent anything. *Prima facie*, it may appear then that a student armed with a pencil and paper can achieve much more than *Mathematica*. Of course, when deriving the log-likelihood, the student makes use of information which *Mathematica* does not 'know'; for example, the student knows that Y_i takes non-negative integer values, that n is a positive integer, that θ is positive and real-valued, that each term in the product is positive and real-valued etc… *Mathematica* assumes nothing about the symbols that have been input, so its inaction is perfectly reasonable.

Fortunately, ML problems have some common features which we can exploit: the contributions to the likelihood are all positive-valued, sample size is a positive integer, and parameters and variables are real-valued. These features can be added to *Mathematica*, not by providing explicit information about each symbol in turn, but rather by providing a pattern-matching function that assumes these common features. To do so, we load our small package entitled SMLE.m (see Appendix listing) into *Mathematica* in the usual way: [*Footnote 1*]

```
<< SMLE.m
```

This package is also available electronically by anonymous ftp at ≪ ftp://ftp.tri.org.au/SMLE.m≫. Moreover, this paper is available as a *Mathematica* notebook at ≪ftp://ftp.tri.org.au/SMLE.nb≫. We then 'activate' our **SuperLog** function as follows:

```
SuperLog[On]
```

```
SuperLog is now On.
```

If we now evaluate **Log[L]** again, we obtain a much more useful result:

```
logL = Log[L]
```

$$-n\,\theta - \sum_{i=1}^{n} \text{Log}[Y_i!] + \text{Log}[\theta] \sum_{i=1}^{n} Y_i$$

In essence, **SuperLog** modifies *Mathematica*'s **Log** function to mimic what our student does: it converts 'logs of products' into 'sums of logs'. It simplifies log-likelihood expressions of form:

$$\log\left(\prod_{i=1}^{n} f_i\right)$$

where *we know* that f_i is defined on the positive real line. To do this, **SuperLog** exploits the power of the *Mathematica* programming language by using pattern-matching code (apart from reserved symbols, the function accepts any indexing symbol, any symbol for sample size, and any functional form for the likelihood contributions). [*Footnote 2*] To return *Mathematica*'s **Log** function to its default status, simply enter: **SuperLog[Off]**.

■ 3. Simple Examples

In this section, we consider three expository examples:

⊕ *Example 1: Poisson*

We have already derived the log-likelihood for the Poisson. The score is:

```
score = ∂θ logL
```

$$-n + \frac{\sum_{i=1}^{n} Y_i}{\theta}$$

where *Mathematica*'s internal partial differential operator ∂_θ has been used. Setting the score to zero defines the ML estimator $\hat{\theta}$. The resulting equation can be easily solved using *Mathematica*'s **Solve** function. The ML estimator $\hat{\theta}$ is given as a replacement rule → for θ:

```
θ̂ = Solve[score == 0, θ]
```

$$\left\{\left\{\theta \to \frac{\sum_{i=1}^{n} Y_i}{n}\right\}\right\}$$

... where the second order conditions confirm that $\hat{\theta}$, the mean of the random sample, is indeed the ML estimator.

```
∂{θ,2} logL
```

$$-\frac{\sum_{i=1}^{n} Y_i}{\theta^2}$$

Finally, let us suppose that an observed random sample is (1,0,3,4):

```
data = {1, 0, 3, 4};
```

Then the ML estimate of θ is obtained by substitution of this data into the ML estimator $\hat{\theta}$: [*Footnote 3*]

```
θ̂ /. {n → 4, Yi :→ data[[i]]}
```

$$\{\{\theta \to 2\}\}$$

⊕ *Example 2: Exponential*

Suppose that the positive-valued continuous random variable Y is such that $Y \sim$ Exponential(θ), where parameter $\theta > 0$. For a random sample of size n on Y, we have the log-likelihood function as:

```
logL = Log[∏_{i=1}^{n} e^(-Yi/θ) / θ] // Apart
```

$$-n \, \text{Log}[\theta] - \frac{\sum_{i=1}^{n} Y_i}{\theta}$$

which has been further simplified by using the **Apart** function. The score is:

```
score = ∂θ logL
```

$$-\frac{n}{\theta} + \frac{\sum_{i=1}^{n} Y_i}{\theta^2}$$

First order conditions:

```
θ̂ = Solve[score == 0, θ]
```

$$\left\{\left\{\theta \to \frac{\sum_{i=1}^{n} Y_i}{n}\right\}\right\}$$

Second-order conditions evaluated at $\hat{\theta}$:

```
∂{θ,2} logL /. Flatten[θ̂]
```

$$-\frac{n^3}{\left(\sum_{i=1}^{n} Y_i\right)^2}$$

Hence, the hessian is strictly negative. The ML estimator is thus $\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} Y_i$.

⊕ *Example 3: Beta*

Let the proportion Y be a random variable defined over the unit interval of the real line, $0 < y < 1$. Furthermore, let the probability density function (pdf) of Y be given by:

```
f = θ y^(θ-1);
```

Thus $Y \sim \text{Beta}(\theta, 1)$, where parameter $\theta > 0$ is unknown. Clearly the distribution of Y is a special case of the standard two-parameter Beta distribution. For a random sample of size n on Y, the log-likelihood for θ is given by:

```
logL = Log[∏_{i=1}^{n} (f /. y → Yi)]
```

$$n\,\text{Log}[\theta] + (-1+\theta)\sum_{i=1}^{n}\text{Log}[Y_i]$$

The MLE of θ is derived as:

```
θ̂ = Solve[∂θ logL == 0, θ]
```

$$\left\{\left\{\theta \to -\frac{n}{\sum_{i=1}^{n}\text{Log}[Y_i]}\right\}\right\}$$

because on inspecting the hessian

```
∂{θ,2} logL
```

$$-\frac{n}{\theta^2}$$

we see that it is negative for all θ; thus the log-likelihood is globally concave in θ and the solution to the first-order condition corresponds to the unique maximum.

■ 4. Further Illustrations

In this section, we consider three examples in which *Mathematica* is used to derive the ML estimator. All three are commonly used when teaching ML estimation. In addition, we show how to use *Mathematica* to tackle other important issues concerning ML estimation such as: deriving the distribution of a ML estimator; proving concavity of a log-likelihood; and concentration of a log-likelihood.

⊕ *Example 3 (continued): Beta*

Continuing the Beta(θ, 1) model, consider the problem of determining the exact distribution of the ML estimator $\hat{\theta} = -n / \sum_{i=1}^{n} \log(Y_i)$. This can be attempted with *Mathematica* by using the moment generating function (mgf) method (Mittelhammer, 1996, sec. 3.5, provides a good description of this method). First, we derive the mgf of $\log(Y)$; that is, we find $E[\exp(t \log(Y))] = E[Y^t]$:

```
SetOptions[Integrate, GenerateConditions → False];
```

$$\text{mgf} = \int_0^1 y^t \, f \, dy$$

$$\frac{\theta}{t + \theta}$$

where, for simplicity, we have turned off the **Integrate** option which yields conditional output. Next, because $(Y_1, ..., Y_n)$ is a random sample of size n, the mgf of

$$\overline{\log Y} \;=\; -\frac{1}{n} \sum_{i=1}^{n} \log(Y_i) \;=\; \frac{1}{\hat{\theta}}$$

is, by the mgf theorem, equal to

$$(E[\exp(\tfrac{-t}{n} \log(Y))])^n \;=\; (E[Y^{-t/n}])^n.$$

Given our previous output, we only need to use a replacement rule to determine the mgf of $\overline{\log Y}$:

```
(mgf /. t → -t/n)^n // Simplify
```

$$\left(\frac{n\,\theta}{-t+n\,\theta}\right)^n$$

This expression can be recognised as the mgf of a random variable $W \sim \text{Gamma}\,(n,\ \frac{1}{n\theta})$, as we verify by comparing it to the following:

```
g = w^(a-1) e^(-w/b) / Gamma[a] b^a /. {a → n, b → 1/(n θ)};
```

```
∫_0^∞ e^(t w) g dw // PowerExpand
```

$$n^n\,\theta^n\,(-t+n\,\theta)^{-n}$$

Given that $\overline{\log Y}$ is Gamma distributed, and that $\hat{\theta} = \frac{1}{\overline{\log Y}}$, it follows that the ML estimator $\hat{\theta}$ has an inverted Gamma distribution with parameters n and $\frac{1}{n\theta}$. The pdf of $\hat{\theta} = q > 0$ is easily derived by transformation:

```
w = 1/q;      pdf = Abs[∂_q w] g
```

$$\frac{E^{-\frac{n\theta}{q}}\ \left(\frac{1}{q}\right)^{-1+n}\ \left(\frac{1}{n\,\theta}\right)^{-n}}{\text{Abs}[q]^2\ \text{Gamma}[n]}$$

If desired, this expression can be further simplified using a replacement rule, since the argument of the absolute value is always positive. The mean of the ML estimator is easily derived:

```
∫_0^∞ q pdf dq // FullSimplify
```

$$\frac{n\,\theta}{-1+n}$$

It is easy to see from this output that the ML estimator $\hat{\theta}$ is biased upwards.

⊕ ***Example 4: Normal Linear Regression Model***

A statistical model of considerable practical importance is the normal linear regression model. For purposes of illustration, we shall consider a simple case of this model, namely a regression model with a constant dummy and one regressor variable X. For a given value of $X = x$, the conditional distribution of the dependent variable Y is assumed to be:

$$Y \mid (X = x) \sim N(\beta_1 + \beta_2 x, \gamma),$$

where parameter $\theta = (\beta_1, \beta_2, \gamma)$ (we use γ to denote the variance parameter). Denote a random sample of T pairs on (Y, X) by $((Y_1, X_1), ..., (Y_T, X_T))$. We assume, conditional upon each $X_k = x_k$, that Y_i is independent of Y_j for all $i \neq j$ $(i, j, k = 1, ..., T)$. Under these assumptions, the log-likelihood is given by:

$$\text{logL} = \text{Log}\left[\prod_{k=1}^{T} \frac{e^{-\frac{(Y_k-\mu)^2}{2\gamma}}}{\sqrt{2\pi\gamma}} \;/.\; \mu \to \beta_1 + x_k \beta_2\right]$$

$$-\frac{1}{2\gamma}\left(T\gamma\,\text{Log}[2] + T\gamma\,\text{Log}[\pi] + T\gamma\,\text{Log}[\gamma] + T\beta_1^2 + \beta_2^2\sum_{k=1}^{T} x_k^2 + 2\beta_1\left(\beta_2\sum_{k=1}^{T} x_k - \sum_{k=1}^{T} Y_k\right) - 2\beta_2\sum_{k=1}^{T} x_k Y_k + \sum_{k=1}^{T} Y_k^2\right)$$

The score vector is given by:

$$\text{score} = \{\partial_{\beta_1}\text{logL}, \partial_{\beta_2}\text{logL}, \partial_{\gamma}\text{logL}\} \;\; // \text{Simplify}$$

$$\left\{\frac{-T\beta_1 - \beta_2\sum_{k=1}^{T} x_k + \sum_{k=1}^{T} Y_k}{\gamma},\right.$$

$$\left.\frac{-\beta_1\sum_{k=1}^{T} x_k - \beta_2\sum_{k=1}^{T} x_k^2 + \sum_{k=1}^{T} x_k Y_k}{\gamma}, \frac{1}{2\gamma^2}\left(-T\gamma + T\beta_1^2 + \beta_2^2\sum_{k=1}^{T} x_k^2 + 2\beta_1\left(\beta_2\sum_{k=1}^{T} x_k - \sum_{k=1}^{T} Y_k\right) - 2\beta_2\sum_{k=1}^{T} x_k Y_k + \sum_{k=1}^{T} Y_k^2\right)\right\}$$

The ML estimators of θ are derived by *Mathematica* as:

```
θ̂ = Solve[score == {0, 0, 0}, {β₁, β₂, γ}]
```

$$\left\{\left\{\gamma \to \left(-\sum_{k=1}^{T} x_k^2 \left(\sum_{k=1}^{T} Y_k\right)^2 + 2\left(\sum_{k=1}^{T} x_k\right)\left(\sum_{k=1}^{T} Y_k\right)\sum_{k=1}^{T} x_k\, Y_k - T\left(\sum_{k=1}^{T} x_k\, Y_k\right)^2 - \left(\sum_{k=1}^{T} x_k\right)^2 \sum_{k=1}^{T} Y_k^2 + T\left(\sum_{k=1}^{T} x_k^2\right)\sum_{k=1}^{T} Y_k^2\right) \Big/ \left(T\left(-\left(\sum_{k=1}^{T} x_k\right)^2 + T\sum_{k=1}^{T} x_k^2\right)\right),\right.$$

$$\beta_1 \to \frac{\left(\sum_{k=1}^{T} x_k^2\right)\sum_{k=1}^{T} Y_k - \left(\sum_{k=1}^{T} x_k\right)\sum_{k=1}^{T} x_k\, Y_k}{-\left(\sum_{k=1}^{T} x_k\right)^2 + T\sum_{k=1}^{T} x_k^2},$$

$$\left.\beta_2 \to \frac{\left(\sum_{k=1}^{T} x_k\right)\sum_{k=1}^{T} Y_k - T\sum_{k=1}^{T} x_k\, Y_k}{\left(\sum_{k=1}^{T} x_k\right)^2 - T\sum_{k=1}^{T} x_k^2}\right\}\right\}$$

The functional form given for the estimator is unfamiliar and imposing. However, if we iteratively solve the first-order conditions:

```
β̃₁ = Solve[score[[1]] == 0, β₁] // Simplify
```

$$\left\{\left\{\beta_1 \to \frac{-\beta_2 \sum_{k=1}^{T} x_k + \sum_{k=1}^{T} Y_k}{T}\right\}\right\}$$

```
β̃₂ = Solve[(score[[2]] /. β̃₁) == 0, β₂] // Simplify
```

$$\left\{\left\{\beta_2 \to \frac{\left(\sum_{k=1}^{T} x_k\right)\sum_{k=1}^{T} Y_k - T\sum_{k=1}^{T} x_k\, Y_k}{\left(\sum_{k=1}^{T} x_k\right)^2 - T\sum_{k=1}^{T} x_k^2}\right\}\right\}$$

... it is fairly easy to see that these results are just the well-known formulae for the estimators presented in most elementary texts:

$$\hat{\beta}_1 = \overline{Y} - \hat{\beta}_2\,\bar{x} \quad \text{and} \quad \hat{\beta}_2 = \frac{\sum_{k=1}^{T}(Y_k - \overline{Y})(x_k - \bar{x})}{\sum_{k=1}^{T}(x_k - \bar{x})^2},$$

Further analysis might include: verification of the equivalence between functional forms, and a formal check of the second-order conditions. Both exercises involve a mixture of work with *Mathematica*,and with pencil and paper.

For data collected on the variables, it is straightforward to compute ML estimates. For example, if the data is:

```
depY = {1, 0, 3, 4};      regX = {1, 2, 3, 4};
```

the ML estimates are:

```
θ̂ /. {T → 4, Y_k :↦ depY[[k]], x_k :↦ regX[[k]]}
```

$$\left\{\left\{\gamma \to \frac{7}{10},\ \beta_1 \to -1,\ \beta_2 \to \frac{6}{5}\right\}\right\}$$

⊕ ***Example 5: Gamma***

Let $(Y_1, ..., Y_T)$ denote a random sample of size T collected on a random variable $Y = y > 0$ which is Gamma distributed; that is, $Y \sim \text{Gamma}(\alpha, \beta)$, where parameter $\theta = (\alpha, \beta)$, with $\alpha > 0$ and $\beta > 0$. We wish to determine the ML estimator of θ. The log-likelihood is:

```
logL = Log[ ∏_{i=1}^{T} Y_i^(α-1) ℯ^(-Y_i/β) / (Gamma[α] β^α) ]  // Apart
```

$$-T\,\alpha\,\text{Log}[\beta] - T\,\text{Log}[\text{Gamma}[\alpha]] - \sum_{i=1}^{T}\text{Log}[Y_i] + \alpha\sum_{i=1}^{T}\text{Log}[Y_i] - \frac{\sum_{i=1}^{T} Y_i}{\beta}$$

For this problem the ML estimator of θ admits, only in part, a closed form solution. To see this, consider the score:

```
score = {∂_α logL, ∂_β logL}
```

$$\left\{-T\,\text{Log}[\beta] - T\,\text{PolyGamma}[0, \alpha] + \sum_{i=1}^{T}\text{Log}[Y_i],\ -\frac{T\,\alpha}{\beta} + \frac{\sum_{i=1}^{T} Y_i}{\beta^2}\right\}$$

where the **PolyGamma[0, α]** term is *Mathematica*'s notation for the digamma function (the first derivative of the natural logarithm of the gamma function). In order for *Mathematica* to derive the ML estimator in closed form, it must

successfully execute `Solve[score=={0,0},{α,β}]`. Unfortunately, this task cannot be performed in this case due to the presence of the **PolyGamma** function. Nevertheless, we can make progress by concentrating the log-likelihood in β. This is because:

```
β̂ = Solve[score[[2]] == 0, β]
```

$$\left\{\left\{\beta \to \frac{\sum_{i=1}^{T} Y_i}{T\,\alpha}\right\}\right\}$$

is in a closed form. *Mathematica* has given us $\hat{\beta}$ as a function of α, i.e. $\hat{\beta} = \hat{\beta}(\alpha)$. Hence, the ML estimator of β is:

$$\hat{\beta}\,(\hat{\alpha}) = \frac{1}{\hat{\alpha}\,T} \sum_{i=1}^{T} Y_i,$$

where $\hat{\alpha}$ denotes the ML estimator of α. Replacing β with $\hat{\beta}(\alpha)$ in the log-likelihood yields the concentrated log-likelihood:

```
ConlogL = logL /. Flatten[β̂]
```

$$-T\,\alpha - T\,\text{Log}[\text{Gamma}[\alpha]] - T\,\alpha\,\text{Log}\left[\frac{\sum_{i=1}^{T} Y_i}{T\,\alpha}\right] - \sum_{i=1}^{T} \text{Log}[Y_i] +$$

$$\alpha \sum_{i=1}^{T} \text{Log}[Y_i]$$

As in the original problem, maximisation of the concentrated log-likelihood with respect to α does not yield a closed form solution. Hence, the ML estimator is defined implicitly by:

$$\hat{\alpha} = \arg\max_{\alpha > 0} \textbf{ConlogL}\,.$$

For a specific set of data, numerical techniques are required to determine the ML estimate $\hat{\alpha}$.

With actual data, the issue of determining a suitable numerical algorithm to maximise the observed concentrated log-likelihood becomes important. *Mathematica*'s inbuilt optimiser, **FindMinimum**, has a suite of the more popular gradient method algorithms available to the user, including the Newton-Raphson algorithm. Of course, which of these algorithms, if any, is appropriate, depends very much on the particular situation at hand. In the

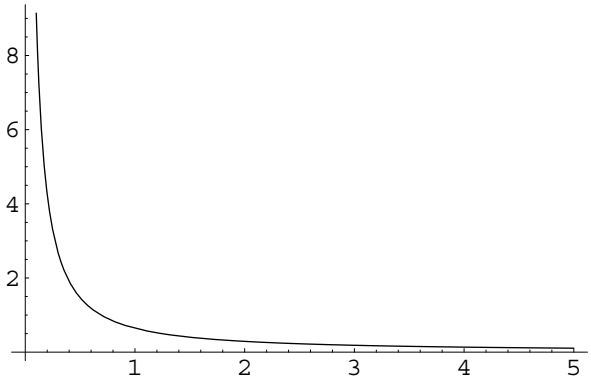
present case, further algebra with *Mathematica* enables us to uncover concavity in α:

```
∂{α,2} ConlogL // Factor
```

$$-\frac{T\,(-1+\alpha\,\text{PolyGamma}[1,\,\alpha])}{\alpha}$$

The sign of the derivative obviously depends on the quantity within round braces. A plot against various $\alpha > 0$ is a simple, informal device for determining its sign; for example:

```
Plot[-1 + α PolyGamma[1, α], {α, 0.1, 5}];
```



As the plot is everywhere positive, the hessian must be negative for all $\alpha > 0$. Thus the concentrated log-likelihood is concave in α, and we may conclude that the Newton-Raphson algorithm is well suited for estimating α.

■ 5. Conclusion

While computer software has long been used for ML estimation, the focus has almost always been on numerical problems rather than symbolic ones. In this paper, we have shown how the *Mathematica* computer algebra system can be used to derive *symbolic* ML estimators. This was done by modifying *Mathematica*'s **Log** command using our **SuperLog** function. This is especially useful when constructing the log-likelihood for a random sample consisting of independent random variables. We have shown how this new function enhances *Mathematica*'s functionality, and demonstrated its particular benefits for the teaching of ML estimation.

■ Acknowledgements

Revisions to this paper were undertaken while Smith was visiting the Sonderforschungsbereich 386, Institüt für Statistik, Ludwig-Maximilians-Universität, München, Deutschland. The content of the paper benefitted from presentations at the Worldwide *Mathematica* Conference (Chicago, 1998), the IMACS Conference on Applications of Computer Algebra (Prague, 1998), the RSS International Conference (Glasgow, 1998), and from discussions with seminar participants at the University of Sydney. We also thank the editor and referee for helpful comments. Smith acknowledges gratefully the support of the the Alexander von Humboldt-Stiftung, and the Australian Research Council (through VISLAB and MEMLab).

■ Footnotes

1. The code given in the SMLE.m package should be stored as a text file on disk under the name SMLE.m. It should be placed in any one of the directories which *Mathematica* searches when attempting to find an external file - this list of directories can be determined by entering **$Path**.

2. The SMLE.m package has been tested using both *Mathematica* version 3 and the forthcoming version 4 release, on platforms including Mac, Windows95, WinNT, SGI UNIX, and DEC UNIX.

3. There exist a number of ways to enter data into *Mathematica* objects. The method we use here is based on one of *Mathematica*'s replacement rules **:→**. The notation **data[[i]]** is *Mathematica* notation for the i^{th} element of the list **data**.

■ References

Cook, P., and Broemeling, L. (1995), "Bayesian statistics using *Mathematica*", *American Statistician* **49**, 70-76.

Currie, I. D. (1995), "Maximum likelihood estimation with *Mathematica"*, *Journal of the Royal Statistical Society (Applied Statistics)* **C44**, 379-394.

Mittelhammer, R. (1996), *Mathematical Statistics for Economics and Business*, New York: Springer.

Stafford, J., and Andrews, D. (1993), "A symbolic algorithm for studying adjustments to the profile likelihood", *Biometrika* **80**, 715-730.

Stafford, J., Andrews, D., and Wang, Y. (1994), "Symbolic computation: a unified approach to studying likelihood", *Statistics and Computing* **4**, 235-245.

Wolfram, S. (1996), *The Mathematica Book*, Cambridge: Cambridge University Press.

■ Glossary

For completeness, we provide a short summary of the use and syntax of a number of the *Mathematica* functions that appear in the paper. Further details of these, and all other functions, can be found in Wolfram (1996); see also the on-line help provided with the *Mathematica* software.

Simplication Rules

Functions **Apart**, **Factor**, **FullSimplify**, **PowerExpand**, and **Simplify** represent a portion of a suite of commands designed to perform various algebraic manipulations to a specified expression. The syntax of each of these commands is, for example, **Simplify[*expr*]** or **expr // Simplify**. The latter is the form which we generally prefer.

Parentheses

Parentheses in *Mathematica* have distinct interpretations, and are not interchangeable. Square braces **[ ]** are reserved for use with *Mathematica* function names; e.g., the logarithm function is **Log[ ]**. A list of elements is collected between curly braces **{ }**; vectors and matrices are built from lists. Round braces **( )** are the only parentheses which have the usual term-collective interpretation.

Partial Differentiation

∂_θ is *Mathematica*'s partial differential operator. Its syntax, for the partial derivative of an expression ***expr*** with respect to a symbol θ, is ∂_θ ***expr***. Syntax for the second-order partial derivative is $\partial_{\{\theta,2\}}$ ***expr***.

Replacement Rules

The ability to replace, in a given expression, a symbol with other symbols or numeric quantities is a vital component of the *Mathematica* programming language. For instance, ***expr*** **/.** $\theta \to \tau$, acts on ***expr*** by replacing symbol θ with τ wherever the former occurs in ***expr***. By contrast, ***expr*** **/.** $\theta :\to \tau$, acts on ***expr*** by replacing symbol θ with τ, but delaying the evaluation of τ until after the replacement is performed. This form of transformation is especially useful when ***expr*** is a symbolic sum, and we wish to evaluate it for a specific set of data.

Search Functions

Solve searches for solutions to an equation or set of equations. The command **Solve[*expr* == 0, *θ*]** attempts to find the values of ***θ*** for which ***expr* == 0**. If ***sol*** denotes the solution to an equation, the output from **Solve** is reported in the form of a list of replacement rules, for example, **{*θ* → *sol*}**. In the case of polynomial equations, *Mathematica* attempts to output all solutions, including complex-valued solutions. Sets of equations must be enclosed within a list, for example, **{*expr1* == 0, *expr2* == 0}**; similarly, if the solutions are sought in two or more variables, then those variable symbols must be placed in a list too, for example, **{*θ*1, *θ*2}**. Hence, **Solve[{*expr1* == 0, *expr2* == 0}, {*θ*1, *θ*2}]**.

■ Appendix

```
(*:Name:     SMLE                                                *)
(*:Authors:  Colin Rose and Murray D. Smith                      *)
(*:Version:  Mathematica v3, or v4, or later required            *)
(*:Legal:    Copyright 1999                                      *)
(*:Summary:  Symbolic Maximum Likelihood Estimation              *)

BeginPackage["SMLE`"]

SuperLog::usage =
    "SuperLog[On] activates the enhanced Log operator, so that
    Log[Product[__]] objects get converted into sums of logs.
    SuperLog[Off] switches the enhancement off."

Begin["`Private`"]

SuperLog[Q_] := Module[{erk,iii,nnn},

  Product[iii,{iii,nnn}]; (* pre-load Product *)
  Which[
   Q === On,
    Unprotect[Log]; Clear[Log];
    Log[Product[x_, {k_, a_, b_}]] :=
           Log[Product[Times[erk, x], {k, a, b}]] /. erk -> 1;
    Log[Product[HoldPattern[Times[x__]], {k_, a_, b_}]] := Simplify[
       Map[Sum[#, {k, a, b}]&,
             Plus@@Map[Expand[PowerExpand[Log[#]]]&, List[x]] ] //.
             Sum[u_. w_, {kk_,aa_,bb_}] :>
                       u Sum[w, {kk,aa,bb}] /; FreeQ[u, kk]==True];
    Protect[Log]; Print["SuperLog is now On."],

   Q === Off,
      Unprotect[Log]; Clear[Log]; Protect[Log]; Print["SuperLog is now Off."],

   True,
      Print["Carumbah!  Please use SuperLog[On] or SuperLog[Off]."]
      ]]
End[]
Protect[ SuperLog ];
EndPackage[]
```