Maximum Likelihood Programming in Stata

Marco R. Steenbergen
Department of Political Science
University of North Carolina, Chapel Hill

August 2003

Contents

1 Introduction

Maximum likelihood-based methods are now so common that most statistical software packages have "canned" routines for many of those methods. Thus, it is rare that you will have to program a maximum likelihood estimator yourself. However, if this need arises (for example, because you are developing a new method or want to modify an existing one), then Stata offers a user-friendly and flexible programming language for maximum likelihood estimation (MLE).

In this document, I describe the basic syntax elements that allow you to write and execute MLE routines in Stata (Versions 7 and 8). I do not intend to offer an in-depth discussion of all of Stata's many features—for this you should consult Gould and Sribney (1999) and Gould, Pitblado, and Sribney (2003). My objective is merely to provide you with enough tools that you can write a simple MLE program and implement it.

2 Features

Stata has many nice features, including: (1) quick convergence (under most circumstances) using the Newton-Raphson algorithm (Stata 8 also offers quasi-Newton algorithms); (2) a conservative approach to declaring convergence, which leads to more trustworthy estimates; (3) simplifying features that allow implementing MLE with a minimum of calculus; (4) robust variance estimation; (5) Wald and likelihood ratio test procedures; (6) a search routine that chooses improved starting values; (7) estimation under linear constraints; and (8) post-estimation commands. These features make Stata one of the easiest MLE programs to work with.

3 Syntactic Structure

Programming and executing MLE routines in Stata requires a specific sequence of commands. These may be part of an `ado` file, or they can be entered interactively. The following shows the sequence of commands and explains their meaning. Optional commands are indicated by an asterisk.

1. Program instructions: The program specifies the parameters and log-likelihood function. This is done in general terms, so that the commands can be used in any application where they are relevant. (The program may be kept in a separate `ado` file.)

2. `ml model`:[1] This command specifies the model that is to be estimated (i.e., dependent variable and predictors), as well as the MLE program that should be run and the way in which it should be run. This command is application-specific: it specifies the model in terms of the particular set of variables that is loaded into memory.

[1]In this document, I indicate Stata commands in `print type`.

3. `ml check`*: This command checks the program syntax for mistakes. While optional, it is extremely useful for debugging MLE routines. Beginning programmers are advised to use this command.

4. `ml search`*: This optional command causes Stata to search for better starting values for the numerical optimization algorithm.

5. `ml maximize`: This command starts the execution of the estimation commands and generates the output.

6. `ml graph`*: This is an optional command that produces a graph showing the iteration path of the numerical optimization algorithm. I recommend using this command so that one can monitor convergence.

3.1 Program Instructions

In most cases, writing and MLE program requires only a couple of lines of syntax. At least, this is the case if (1) the log-likelihood function meets the linear form restriction—i.e., the observations are independent—and (2) Stata derives the first and second (partial) derivatives numerically (these derivatives are needed for the Newton-Raphson algorithm). In this document, I assume that these two conditions are met.[2]

The program is started by entering

```
program define name
```

where *name* is any name up to 32 characters in length. (It is preferable to choose a descriptive name.) The user may abbreviate `program define` to `pr de`. To end the program, one should type

```
end
```

In between these keywords, the user has to declare the parameters and the log-likelihood function. First, the log-likelihood function and its parameters have to be labeled. This is done through the command `args` (which is an abbreviation for the computer term "arguments"). Next, the log-likelihood function has to be defined; this is done using the `quietly replace` command.[3] In addition to these specifications, it is often useful to declare the program version, especially if you are planning to make changes to the program over time.

[2]That is, the maximization method is `lf` (see the discussion of `ml model` below). Please note that `lf` is the easiest approach in Stata but not always the most accurate. However, in my programming experience I have never encountered an instance in which the results from `lf` were misleading.

[3]"Replace" indicates that the user is substituting a new expression. "Quietly" implies that Stata does not echo this substitution—i.e., it is not displayed on the screen or in the output.

Example 1: To show the use of these commands, consider the simple example of the Poisson distribution:

$$f(y|\mu) = \frac{\mu^y e^{-\mu}}{y!}$$

Here μ is the parameter that we want to estimate. For a sample of n independent observations, this distribution produces the following log-likelihood function:

$$l(\mu|y_1, y_2 \cdots y_n) = \sum_i y_i \ln(\mu) - n\mu - \sum_i \ln(y_i!)$$

To program this function, we could use the following syntax:

```
program define poisson
   version 1.0
   args lnf mu
   quietly replace ‘lnf’ = $ML_y1*ln(‘mu’)- ‘mu’ - lnfact($ML_y1)
end
```

Let us analyze what this program does. In the first line we define the program, calling it `poisson`. In the second line, we show the version of the program (`version 1.0`). The third line provides a name for the log-likelihood function (`lnf`) and its one parameter (`mu`). The fourth line specifies the log-likelihood function and the fifth line ends the program.

The action, of course, is in the fourth line. This line is based on the arguments specified in `args`. Because we are referring to arguments, they should be placed in apostrophes. (In fact, the leading apostrophe is backward leaning and is typically located on the same key as the tilde; the second apostrophe is straight and is typically located on the same key as the double apostrophe.) The fourth line also contains the variable `$ML_y1`, which is the internal label for the (first) dependent variable. Stata will replace this with an appropriate variable from the data set after the `ml model` command has been specified.[4] Finally, the fourth line specifies a function. (The last term in this expression, `lnfact($ML_y1)`, stands for $\ln(y!)$.)

A careful inspection of the fourth line of code shows that it looks a lot like the log-likelihood function, except that it does not include summations. In fact, this line gives the log-likelihood function for a single observation:

$$l(\mu|y_i) = y_i \ln(\mu) - \mu - \ln(y_i!)$$

As long as the observations are independent (i.e., the linear form restriction on the log-likelihood function is met), this is all you have to specify. Stata knows that it should evaluate this function for each observation in the data and then sum the results. This greatly simplifies programming log-likelihood functions.[5]

[4]By not referring to a specific variable name, the program can be used for any data set. This is quite useful, as you do not have to go back into the program to change variable names.

[5]Keep in mind, however, that this will only work if the observations are independent and the linear form restriction is met.

Example 2: As a second example, consider the normal probability density function:

$$\begin{aligned} f(y|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right\} \\ &= \frac{1}{\sigma}\phi(z) \end{aligned}$$

where $z = \frac{(y-\mu)}{\sigma}$ and $\phi(.)$ denotes the standard normal distribution.[6] Imagine that we draw a sample of n independent observations from the normal distribution, then the log-likelihood function is given by

$$l(\mu, \sigma^2|y_1, y_2 \cdots y_n) = -n\ln(\sigma) + \sum_i \ln[\phi(z_i)]$$

We can program this function using the following syntax:

```
program define normal
   version 1.0
   args lnf mu sigma
   quietly replace ‘lnf’=ln(normd(($ML_y1-‘mu’)/‘sigma’))-
   ln(‘sigma’)
end
```

Here `normd` is $\phi(.)$ and `($ML_y1-‘mu’)/‘sigma’` is z_i. Again, we only have to specify the log-likelihood function for a single observation. Stata will evaluate this function for all observations and accumulate the results to obtain the overall log-likelihood.

[6]To derive the second line of this equation, we proceed as follows. First, we substitute z in the formula for the normal distribution:

$$f(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \; -.5z^2$$

Next, we compare this result to the standard normal distribution:

$$\phi(z) = \frac{1}{\sqrt{2\pi}} \exp \; -.5z^2$$

We see that the generic normal distribution is almost identical to the standard normal distribution; the only difference is a term in σ^2. Factoring this term out, we get

$$\begin{aligned} f(y|\mu, \sigma^2) &= \frac{1}{\sqrt{\sigma^2}} \frac{1}{\sqrt{2\pi}} \exp \; -.5z^2 \\ &= \frac{1}{\sqrt{\sigma^2}}\phi(z) \\ &= \frac{1}{\sigma}\phi(z) \end{aligned}$$

3.2 Ml Model

To apply a program to a data set, you need to issue the `ml model` command. This command also controls the method of maximization that is used, but I will assume that this method is `lf`—i.e., the linear form restrictions hold and the derivatives are obtained numerically.[7]

The syntax for `ml model` is:

```
ml model lf name equations [if] [, options]
```

Here `ml model` may be abbreviated to `ml mod`, *name* is the name of the MLE program (e.g., `poisson`), and *equations* specifies the model that should be estimated through the program. A subset of the data may be selected through the `if` statement. It is also possible to specify various options, as will be discussed below.

3.2.1 Equations

To perform MLE, Stata needs to know the model that you want to estimate. That is, it needs to know the dependent and, if relevant, the predictor variables. These variables are declared by specifying one ore more equations. The user can specify these equations before running `ml model` by using an alias. It is also possible to specify the equations in the `ml model` command, placing each equation in parentheses.

The general rule in Stata is that a separate equation is specified for each mean model and for each (co)variance model. For example, if we wanted to estimate the mean and variance of a normal distribution, we would need an equation for the mean and an equation for the variance. In a linear regression model, we would need an equation for the conditional mean of y (i.e., $E[y_i|\mathbf{x}_i\mathbf{b}]$) and for the variance (the latter model would include only a constant, unless we specify a heteroskedastic regression model). In a simultaneous equations model, there would be as many equations as endogenous variables, plus additional equations to specify the covariance structure.

Aliasing. We can specify equations before the `ml model` command, giving them an alias that can be used in the command. For example, for the Poisson distribution we could specify the following equation:

```
mean:  y=
```

[7]The name `lf` is an abbreviation for linear form. If the linear form restrictions do not hold, then the user may choose from three other maximization methods: `d0`, `d1`, and `d2`. The difference between these methods lies in the way in which the first and second (partial) derivatives are obtained. Both the first and second derivative are obtained numerically with `d0`. With `d1`, the user has to derive the first derivative analytically (i.e., through calculus), while the second derivative is obtained numerically. With `d2`, both derivatives are obtained analytically; this is generally the most accurate approach. You should note that the use of `d0`, `d1`, and `d2` necessitates additional lines of code in the program to specify the log-likelihood function more completely and, if necessary, to specify the first and second derivatives (see Gould and Sribney 1999; Gould, Pitblado, and Sribney 2003).

We have now generated an equation by the name or alias of "mean" that specifies the model for μ in the Poisson distribution. The equation specifies the dependent variable on the left-hand side—this is the name of a variable in the data set. The right-hand side is empty because there are no predictors of μ (other than the constant).[8]

To estimate this model we type:

```
ml model lf poisson mean
```

Here `lf` is the maximization method, `poisson` is the name of the maximum likelihood program, and `mean` is the alias for the equation specifying the mean model. The alias will appear in the output and can make it easier to read.

Direct Specification. We can also specify the equation directly in the `ml model` command. For the Poisson distribution, we would do this as follows:

```
ml model lf poisson (y=)
```

Since we have not named the equation, it will be labeled in the output as `eq`, followed by a number.[9]

3.2.2 Additional Examples

Example 1: Earlier we wrote a program, titled `normal`, to estimate the parameters of a normal density function. Imagine we want to apply this to a variable named `Y`. Then we need the following specification of the `ml model` command:

```
ml model lf normal (Y=) (Y=)
```

Notice that we now have specified two equations. The first equation calls for the estimation of μ. The second equation calls for the estimation of σ.

Example 2: Now imagine that the normal density describes the conditional distribution of `Y` given two predictors, `X` and `Z`. We assume that the conditional variance of `Y` is constant and given by σ^2. We also assume that the conditional mean of `Y` is given by $\beta_0 + \beta_1 X + \beta_2 Z$. In other words, we are considering the classical linear regression model under the assumption of normality. To estimate this model, one should issue the following command:

```
ml model lf normal (Y=X Z) (Y=)
```

The first equation calls for the estimation of the conditional mean of `Y`, which is a function of the predictors `X` and `Z`. The second equation pertains to the estimation of σ, which is constant so that no predictors are specified.

One sees that the estimation of a normal regression model requires no additional programming compared to the estimation of the mean and variance of

[8]If there are predictor variables, these should be specified after the equal sign.

[9]The direct and aliasing methods may be combined. For details see Gould and Sribney (1999) and Gould, Pitblado, and Sribney (2003).

a normal distribution. This minimizes the burden of programming and gives MLE routines a great deal of “portability.” Both of these features are important benefits of Stata.

3.2.3 Options

There are two options that can be specified with `ml model`, both of which produce robust variance estimates (or Huber-White or sandwhich estimates).

> (1) `robust` generates heteroskedasticity-corrected standard errors. (This may be abbreviated as `rob`.)
> (2) `cluster(`*varname*`)` generates cluster-corrected standard errors, where *varname* is the name of the clustering variable. (This may be abbreviated as `cl`.)

Both of these commands may be specified with the `lf` maximization method.[10] A discussion of robust variance estimation can be found in Gould and Sribney (1999), Gould, Pitblado, and Sribney (2003), and in advanced econometrics textbooks (Davidson and MacKinnon 1993; Greene 2000).

3.3 Ml Check

It is useful to check an MLE program for errors. In Stata, you can do this by issuing the command `ml check`. This command evaluates if the program can compute the log-likelihood function and its first and second derivatives. If there is a problem with the log-likelihood function, or with its derivatives, `ml check` will let the user know. Stata will not be able to estimate the model before these problems are fixed.

3.4 Ml Search

The Newton-Raphson algorithm needs an initial guess of the parameter estimates to begin the iterations. These initial guesses are the so-called *starting values*. In Stata, the user has two options: (1) use a default procedure for starting values or (2) do a more extensive search.

The default procedure in Stata is to set the initial values to 0. If the log-likelihood function cannot be evaluated for this choice of starting values, then Stata uses a pseudo-random number generator to obtain the starting values. (It will regenerate numbers until the log-likelihood function can be evaluated.) This procedure is a quick-and-dirty way to start the Newton-Raphson algorithm.

Through `ml search` (which may be abbreviated as `ml sea`) the selection of starting values can be improved. The `ml search` command searches for starting values based on equations. A nice feature here is that the user can specify boundaries on the starting values. For example, before estimating the Poisson distribution, we could specify

[10] However, other maximization methods may not allow these options or may require adjustments in the program.

```
ml search 1 3
```

This causes the search command to pick starting values for μ that lie between 1 and 3. If the ML estimate lies within these bounds, beginning the iterations there can speed up estimation considerably. Thus, I recommend using the `ml search` command (even if you do not specify bounds), although it can be bypassed.

3.5 Ml Maximize

None of the commands discussed so far actually causes Stata to generate a table of parameter estimates. To do this, the MLE program has to be executed and this is done through the `ml maximize` command. You simply type

```
ml maximize [, options]
```

(which may be abbreviated to `ml max`) and the estimation process commences—at least, when the program is correct and there are no "funky" things in the data.

It is possible to add several options to the `ml maximize` command that control the output and convergence. In general, I recommend against tinkering with these features, but for the sake of completeness I will list the most important options.

(1) `nolog` suppresses the iteration log. This reduces the length of the output, in particular if convergence problems are encountered (see below). Since the iteration log contains important information about convergence, one should not suppress it too quickly.
(2) `iterate(##)` sets the maximum number of iterations (##). The default value is 16000. Since this default is very large—it could take hours to reach this limit—there is usually no reason to change it.
(3) `ltolerance(##)` sets the tolerance, which controls the point at which Stata cuts off the iterations. Specifically, convergence is declared when

$$\left|\frac{l_j - l_{j-1}}{l_{j-1}}\right| \leq ltolerance$$

where l_j is the log-likelihood function for the set of estimates generated in the jth iteration and l_{j-1} is the log-likelihood function for the set of estimates in the previous iteration. The default of `ltolerance` is 1e-7. By increasing this value, convergence can be speeded. However, this is a risky approach because the resulting estimates may not be true maximum likelihood estimates. Hence, the decision to change `ltolerance` should not be made lightly.
(4) `difficult` forces Stata to put in extra effort to estimate difficult log-likelihood functions. Such functions tend to have many ridges, flat areas, and saddle points so that they are not concave.

This makes it impossible to compute the direction vector, which is used to update estimates. The `difficult` option prompts Stata to determine if the direction vector exists. If not, then the program supplements the Newton-Raphson algorithm with a variation on the steepest ascent method to obtain new estimates (see Gould and Sribney 1999; Gould, Pitblado, and Sribney 2003). Specifying `difficult` increases estimation time, so I do not suggest using it by default. However, if Stata generates many warnings about non-concavity, especially on later iterations, it may be worthwhile to repeat estimation using this option.

3.6 Monitoring Convergence and Ml Graph

Even when the program produces output, it is useful to check the convergence of the Newton-Raphson algorithm. One should pay attention to three pieces of information. First, the algorithm should converge relatively quickly. A dozen or so iterations would generally not worry us, especially not for complex estimation problems. But if the algorithm requires a large number of iterations (e.g., in the hundreds), then this could indicate a serious problem.

Second, one should pay attention to warning messages about the concavity of the log-likelihood function. The message `not concave`, which follows the value of the log-likelihood function in the iteration log, indicates that the log-likelihood function is essentially flat at a particular iteration. This means that Stata cannot establish a direction vector to guide new parameter estimates. if this warning message occurs early in the iteration process, it can be safely ignored. However, if this message appears on the last iteration, then this indicates a serious problem. One cannot trust the parameter estimates in this case and should consider re-estimating the model with the `difficult` option (see above).

Third, one should monitor the convergence path. If we place the iterations on the horizontal axis and the corresponding values of l on the vertical axis, then we should ideally see a concave (marginally declining) function. That is, initially the changes in the log-likelihood function should be large between iterations. However, nearing the end of the iterations, these changes should be relatively tiny. Small perturbations of this pattern need not worry us. However, if the iteration function is convex (or a straight line), then we should be worried. This could indicate a bug in the program or an ill-behaved log-likelihood function.

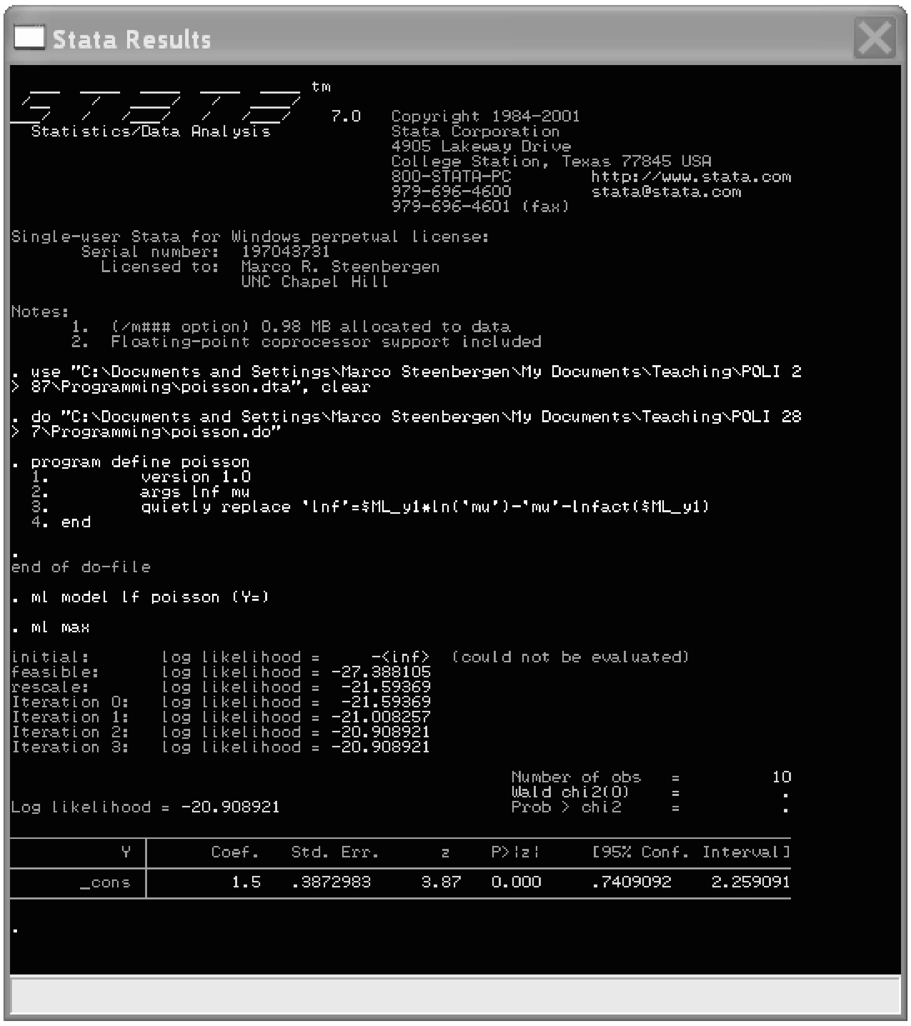
Plotting the log-likelihood against the iterations in Stata is easy. All one has to do is to issue the command

```
ml graph
```

(which may be abbreviated as `ml gr`) after running `ml maximize`. I recommend that you always create this plot, as it reveals a lot about convergence.

4 Output

After running a MLE program, Stata will produce the following output.



```
                                      tm
                                         7.0   Copyright 1984-2001
  Statistics/Data Analysis                     Stata Corporation
                                               4905 Lakeway Drive
                                               College Station, Texas 77845 USA
                                               800-STATA-PC        http://www.stata.com
                                               979-696-4600        stata@stata.com
                                               979-696-4601 (fax)

Single-user Stata for Windows perpetual license:
       Serial number:  197043731
         Licensed to:  Marco R. Steenbergen
                       UNC Chapel Hill

Notes:
      1.  (/m### option) 0.98 MB allocated to data
      2.  Floating-point coprocessor support included

. use "C:\Documents and Settings\Marco Steenbergen\My Documents\Teaching\POLI 2
> 87\Programming\poisson.dta", clear

. do "C:\Documents and Settings\Marco Steenbergen\My Documents\Teaching\POLI 28
> 7\Programming\poisson.do"

. program define poisson
  1.          version 1.0
  2.          args lnf mu
  3.          quietly replace 'lnf'=$ML_y1*ln('mu')-'mu'-lnfact($ML_y1)
  4. end

.
end of do-file

. ml model lf poisson (Y=)

. ml max

initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -27.388105
rescale:       log likelihood =  -21.59369
Iteration 0:   log likelihood =  -21.59369
Iteration 1:   log likelihood = -21.008257
Iteration 2:   log likelihood = -20.908921
Iteration 3:   log likelihood = -20.908921

                                                  Number of obs   =         10
                                                  Wald chi2(0)    =          .
Log likelihood = -20.908921                       Prob > chi2     =          .

------------------------------------------------------------------------------
           Y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |        1.5   .3872983     3.87   0.000     .7409092    2.259091
------------------------------------------------------------------------------

.
```

Figure 1: Stata MLE Commands and Output

(1) An iteration log, showing the iterations and the value of the log-likelihood at each iteration. (This log will not be shown if you specified `nolog` as an option for the `ml maximize` command.)
(2) The final value of the log-likelihood function. (This is always shown, even if you have specified the `nolog` option.)
(3) The number of observations on which the estimation is based.
(4) The Wald chi-square test and its (asymptotic) p-value.
(5) For each equation, the parameter estimates, their estimated standard errors, the test statistics and their p-value, and the upper and lower bounds of the 95% confidence intervals. If the option `robust` or `cluster` was specified on the `ml model` command, then robust standard errors are reported. The test statistic is the ratio of the estimate to its standard error.

Figure 1 shows the output generated by the "poisson" program that we created earlier (in Example 1 of section 3.1) as applied to some artificial data. Following the `ml max` command we first see the iteration history (or iteration log). It took the program 3 iterations to find the ML estimate for μ. Notice that the initial iteration produced an error message because Stata started by setting $\hat{\mu} = 0$ and for this value $\ln(\mu)$, which is part of the log-likelihood function, is not defined. On the final iteration, the log-likelihood function was -20.908921; this value is repeated underneath the iteration log. To the right of the (final) log-likelihood we find the number of observations and the Wald chi-squared and its associated p-value (see below). The bottom portion of the output shows the estimate (1.5), the estimated standard error (.3872988), the z-test statistic (3.87) and its associated p-value (0.000), and the lower and upper bounds of the 95% confidence interval (.7409092 and 2.259091, respectively).

The optional `ml graph` command produces the output in Figure 2 (Stata will show this output in a separate window). The horizontal axis of this graph shows the iteration number and the vertical axis, labeled `ln L0` gives the value of the log-likelihood function at that iteration. The iteration path shown in Figure 2 is precisely what we would like to see: as the iterations progress, changes in the log-likelihood function become ever smaller. (In fact, this example shows no change from the 2nd to 3rd iteration because there is a closed form solution for the ML estimator.)

5 Performing Wald Tests

When using the `ml maximize` command, Stata by default reports a Wald test statistic and its p-value. This statistic compares the fit of a model including predictors to the fit of a model excluding *all* of those predictors. The first model is the unconstrained model, while the second model is the constrained model.[11]

[11]When the model contains no predictors, Stata reports a period for the Wald test statistic and its p-value.

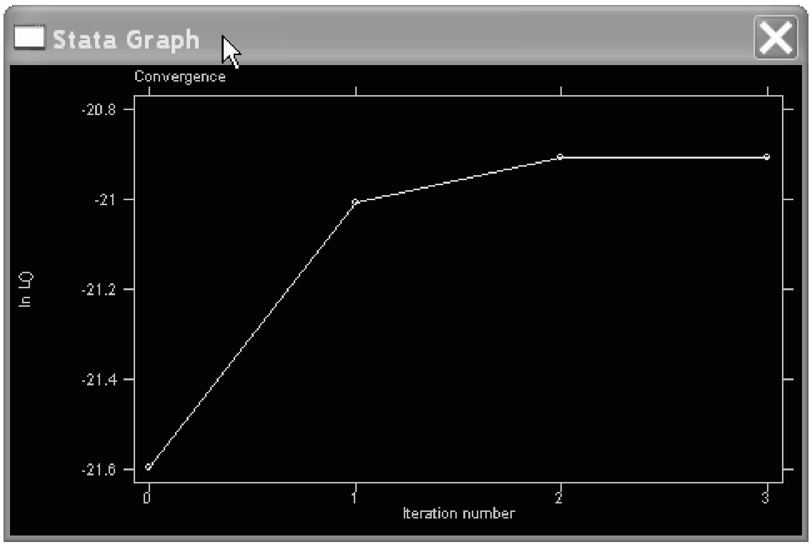


Figure 2: Graph Produced by `ml graph`

Sometimes we may want to exclude only a subset of the predictors. In this case, we can use the `test` command to obtain the relevant Wald test statistic. The test command is very easy to use. After running `ml max`, you simply type in `test` followed by the variables you want to exclude from the model. Stata will then show the relevant Wald test statistic and its p-value.

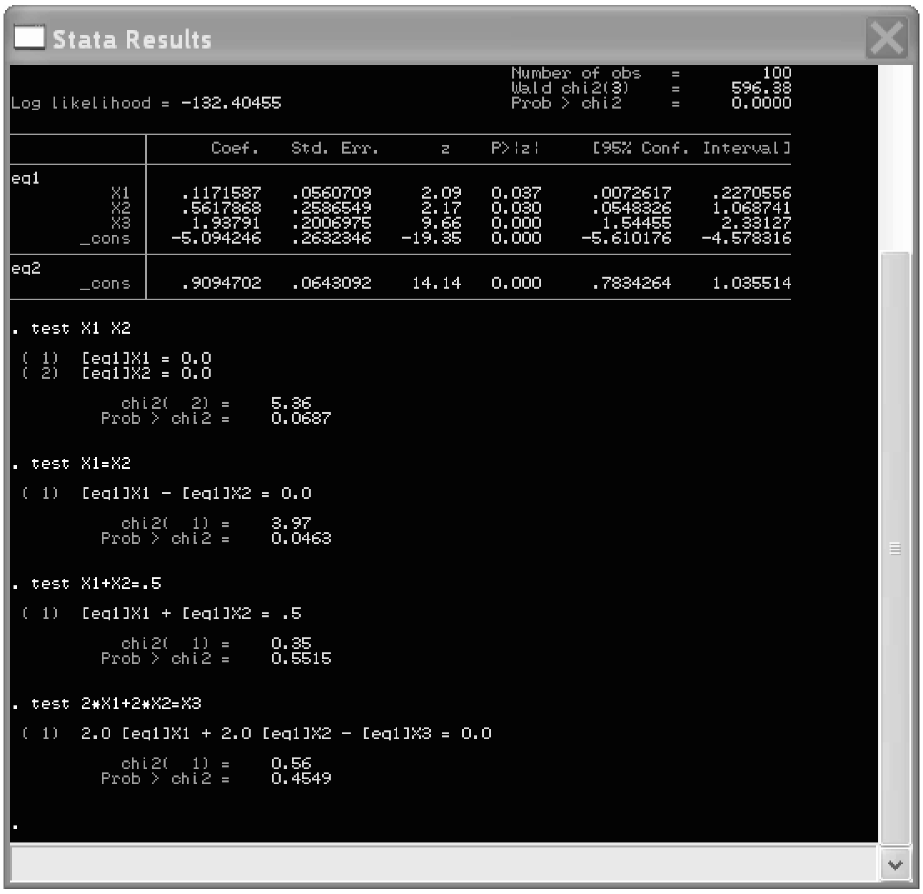
The `test` command can be used to test any hypothesis that involves a linear combination of the parameters. Figure 3 shows a number of examples pertaining to the classical linear regression model with normally distributed errors (Example 2 in Section 3.1).

6 Performing Likelihood Ratio Tests

The Wald test does not actually estimate the constrained model, but evaluates its fit based on the difference between the parameter estimate and its constrained value, as well as the curvature of the log-likelihood function (as measured by the second derivative). A more precise approach to testing is to explicitly estimate the constrained model and to perform a likelihood ratio test. The Stata command for doing this is `lrtest`. This test compares the values of the log-likelihood functions for the constrained and unconstrained models and computes the p-value of the resulting likelihood ratio test statistic.

To use `lrtest`, a specific sequence of estimation commands is typically followed.

(1) Estimate the unconstrained model.
(2) Save the statistics associated with the unconstrained model:



Stata Results

```
                                                  Number of obs   =        100
                                                  Wald chi2(3)    =     596.38
Log likelihood = -132.40455                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
          X1 |   .1171587   .0560709     2.09   0.037     .0072617    .2270556
          X2 |   .5617868   .2586549     2.17   0.030     .0548326    1.068741
          X3 |    1.93791   .2006975     9.66   0.000      1.54455     2.33127
       _cons |  -5.094246   .2632346   -19.35   0.000    -5.610176   -4.578316
-------------+----------------------------------------------------------------
eq2          |
       _cons |   .9094702   .0643092    14.14   0.000     .7834264    1.035514
------------------------------------------------------------------------------

. test X1 X2

 ( 1)  [eq1]X1 = 0.0
 ( 2)  [eq1]X2 = 0.0

           chi2(  2) =    5.36
         Prob > chi2 =    0.0687

. test X1=X2

 ( 1)  [eq1]X1 - [eq1]X2 = 0.0

           chi2(  1) =    3.97
         Prob > chi2 =    0.0463

. test X1+X2=.5

 ( 1)  [eq1]X1 + [eq1]X2 = .5

           chi2(  1) =    0.35
         Prob > chi2 =    0.5515

. test 2*X1+2*X2=X3

 ( 1)  2.0 [eq1]X1 + 2.0 [eq1]X2 - [eq1]X3 = 0.0

           chi2(  1) =    0.56
         Prob > chi2 =    0.4549

.
```

Figure 3: Wald Test Examples

```
lrtest, saving name
```

where *name* is an arbitrary name of no more than 4 characters.
(3) Estimate the constrained model.
(4) Type `lrtest` to perform the likelihood ratio test.

For example, consider the regression model shown in Figure 3. The following sequence of commands allows one to perform the likelihood ratio test for the null hypothesis that $\beta_1 = \beta_2 = 0$.

1. `ml model lf normal (Y=X1 X2 X3) (Y=)`
2. `ml max`
3. `lrtest, saving(0)`
4. `ml model lf normal (Y=X3) (Y=)`
5. `ml max`
6. `lrtest`

The results are shown in Figure 4.

7 General Programming Issues

MLE is a part of Stata's programming language. The following general programming commands will prove quite useful (see Stata 2001).

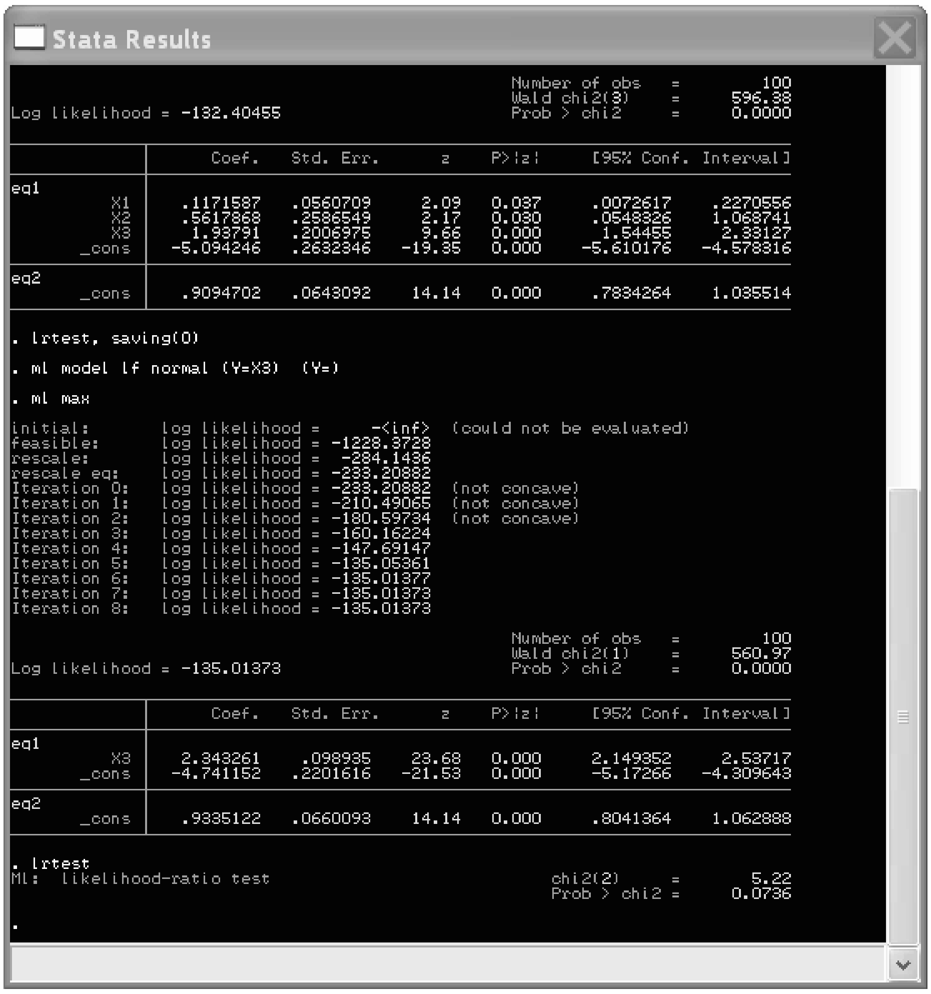
(1) `program dir`: This command shows a list of programs (including `ado` files) that are currently in memory. If a program is already in memory, you cannot define it again.
(2) `program drop`: This command drops a program from memory. Once this command is issued, Stata can no longer execute the affected program(s).

8 Additional MLE Features in Stata 8

The syntax described up to this point will work in Stata Version 7 and Version 8. However, if you have access to Stata 8, you should use it because it has a nice set of additional features. Here, I describe the most important of these features; for a complete discussion see Gould, Pitblado, and Sribney (2003).

Quasi-Newton Algorithms. In Stata 7, numerical optimization always occurs through the Newton-Raphson algorithm (at least, under normal conditions). Stata 8 allows the user to rely instead on one of three different quasi-Newton algorithms, namely BHHH, DFP, and BFGS.[12] Specification of the algorithm occurs in the `ml model` command, as an option:

[12]Fisher scoring is not available in Stata 8, but one would generally not use this when quasi-Newton methods can be used instead.



```
                                                  Number of obs   =        100
                                                  Wald chi2(3)    =     596.38
Log likelihood = -132.40455                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
          X1 |   .1171587   .0560709     2.09   0.037     .0072617    .2270556
          X2 |   .5617868   .2586549     2.17   0.030     .0548326    1.068741
          X3 |    1.93791   .2006975     9.66   0.000      1.54455     2.33127
       _cons |  -5.094246   .2632346   -19.35   0.000    -5.610176   -4.578316
-------------+----------------------------------------------------------------
eq2          |
       _cons |   .9094702   .0643092    14.14   0.000     .7834264    1.035514
------------------------------------------------------------------------------

. lrtest, saving(0)

. ml model lf normal (Y=X3)  (Y=)

. ml max

initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -1228.3728
rescale:       log likelihood =  -284.1436
rescale eq:    log likelihood = -233.20882
Iteration 0:   log likelihood = -233.20882  (not concave)
Iteration 1:   log likelihood = -210.49065  (not concave)
Iteration 2:   log likelihood = -180.59734  (not concave)
Iteration 3:   log likelihood = -160.16224
Iteration 4:   log likelihood = -147.69147
Iteration 5:   log likelihood = -135.05361
Iteration 6:   log likelihood = -135.01377
Iteration 7:   log likelihood = -135.01373
Iteration 8:   log likelihood = -135.01373

                                                  Number of obs   =        100
                                                  Wald chi2(1)    =     560.97
Log likelihood = -135.01373                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
          X3 |   2.343261    .098935    23.68   0.000     2.149352     2.53717
       _cons |  -4.741152   .2201616   -21.53   0.000     -5.17266   -4.309643
-------------+----------------------------------------------------------------
eq2          |
       _cons |   .9335122   .0660093    14.14   0.000     .8041364    1.062888
------------------------------------------------------------------------------

. lrtest
Ml:  likelihood-ratio test                            chi2(2)     =       5.22
                                                      Prob > chi2 =     0.0736

.
```

Figure 4: Likelihood Ratio Test Examples

```
ml model lf name equations [if] , technique(algorithm)
```

where *algorithm* is the abbreviation of the algorithm, which can be `nr` for Newton-Raphson (the default), `bhhh` for Bendt-Hall-Hall-Hausman, `dfp` for Davidson-Fletcher-Powell, or `bfgs` for Broyden-Fletcher-Goldfarb-Shanno.[13]

Linear Constraints. Stata 8 allows maximum likelihood estimation of models with linear constraints. Constraints are specified as an option with the `ml model` command. More specifically, the `constraint` option invokes a constraint that was defined previously. For example, consider the linear regression model $y_i = \beta_0 + \beta_1 x_i + \beta_2 z_i + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma^2)$. Now imagine that we have a priori reason to believe that $\beta_2 = -\beta_1$ (or put differently, $\beta_1 + \beta_2 = 0$). We could build this constraint into the estimation by issuing the following commands:

```
constraint 1 Z=-X
ml model lf normal (Y=X Z) (Y=), constraint(1)
ml max
```

The program will now find estimates that satisfy the constraint, aliased as the number 1. (Whether these estimates are true ML estimates depends, of course, on the validity of the constraint.)[14]

9 References

Davidson, Russell, and James G. MacKinnon. 1993. *Estimation and Inference in Econometrics.* New York: Oxford University Press.

Gould, William, and William Sribney. 1999. *Maximum Likelihood Estimation with Stata.* College Station, TX: Stata Press.

Gould, William, Jeffrey Pitblado, and William Sribney. 2003. *Maximum Likelihood Estimation with Stata.* 2nd ed. College Station, TX: Stata Press.

Greene, William H. 2000. *Econometric Analysis.* 4th ed. Upper Saddle River: Prentice Hall.

Stata. 2001. *Stata Programming Manual, Release 7.* College Station, TX: Stata Press.

[13]It is also possible to switch between different algorithms. For details see Gould, Pitblado, and Sribney (2003).

[14]This example can serve to show a couple of other changes in Stata 8. First, while the `normal` program that we defined earlier will work, Stata 8 also allows a more simplified version of this program:

```
program normal2
Version 8.1
args lnf mu sigma
quietly replace 'lnf'=ln(normden($ML_y1,'mu','sigma'))
end
```

Notice that we can now write `program` instead of `program define` and that the log-likelihood function is simpler. Second, to apply this program we can issue the following command:

```
ml model lf normal2 (Y=X Z) /sigma
```

The instruction `/sigma` will cause Stata to output an estimate of σ.