Advanced Programming in Stata

Kevin Sweeney
PRISM Senior Methods Fellow

Brandon Bartels
PRISM Junior Methods Fellow
Advanced Programming in Stata

- Programming your own maximum likelihood estimator.
  - Basic syntax
  - Likelihood functions
- Examples:
  - Normal regression (easy)
  - Logit and probit (easy)
  - Heteroskedastic regression (harder)
  - Split population duration model (harder)
Programming Likelihood Functions: The Basics

• As you will see, programming your own ML estimator is incredibly easy to do in Stata.

• From last session, we learned how to write a program in Stata using .do files, macros, looping, etc.

• In writing our own likelihood function, we need the following information:
  - An understanding of some of Stata’s “ml” family of commands.
    • Note: The help menus provide very useful information on MLE programming; help ml and/ or help mlmethod
  - Log-likelihood function
  - Syntax for how to maximize the function
  - THAT’S IT! It’s so easy, it’s hard to believe!
Programming Likelihood Functions: Brief MLE Review

• In ML, we first need to specify the data generating process for the dependent variable under examination.
• In other words, we need to specify the probability distribution that generated the dependent variable; e.g., the normal for continuous variable, logit or probit for dichotomous, poisson for count data, etc.
Programming Likelihood Functions: Brief MLE Review

- Then, we specify the likelihood for case $i$:

  $$L_i = L(\theta \mid y_i)$$
  $$L(\theta \mid y_i) \propto p(y_i \mid \theta)$$

- The likelihood for the entire sample is simply the product of individual likelihoods:

  $$L = \prod_{i=1}^{N} L_i$$

- MLEs are the values of the parameters for which the likelihood of observing the sample is maximized.
Programming Likelihood Functions:  
ML Normal Regression

- **Y ~ N(μ, σ^2)**

- **pdf:**  
  \[ f(y_i | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \left(\frac{(y_i - \mu)^2}{\sigma^2}\right)} \]

- Reparameterize \( \mu_i = x_i \beta \)

- Likelihood for case \( i \) :  
  \[ L_i = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \left(\frac{(y_i - x_i \beta)^2}{\sigma^2}\right)} \]

- Log-likelihood for case \( i \) (**this is what Stata wants**):
  \[ \ln L_i = -\frac{1}{2} \ln(2\pi) - \frac{1}{2} \ln(\sigma^2) - \frac{1}{2} \left(\frac{(y_i - x_i \beta)^2}{\sigma^2}\right) \]
Programming Likelihood Functions: ML Normal Regression

• The likelihood for the entire sample is simply the product of the individual likelihoods:

\[ L = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2} (y_i - x_i\beta)^2} \]

• And the log-likelihood for the entire sample is simply:

\[ \ln L = -\frac{N}{2} \ln(2\pi) - \frac{N}{2} \ln(\sigma^2) - \frac{1}{2} \sum_{i=1}^{N} \left( \frac{(y_i - x_i\beta)^2}{\sigma^2} \right) \]

• Again, however, Stata only needs the log-likelihood for case i.
Programming Likelihood Functions: Syntax

- **Goal**: Write a program that Stata can use to maximize a log-likelihood function.
- **First**, Stata has 4 ML “evaluators”: `lf`, `d0`, `d1`, `d2`.
- “lf” is the most basic evaluator; the “d” evaluators are for more advanced programs. We’re only going to use “lf” in this session.
Programming Likelihood Functions: Syntax

```
program define proname
    args lnf theta1 theta2 ...
    tempvar tmp1 tmp2 ...
    quietly gen double `tmp1' = ...
    quietly replace `lnf' = ...
end
```

- `lnf` is a variable to be filled in with values of the log-likelihood for case \(i\) (i.e., \(\ln L_i\)).
- `theta1` is associated with the first parameter, containing evaluation of the 1st equation: \(\theta_1(i) = x_{1i} b\)
- `theta2` is associated with the second parameter, containing evaluation of the 2nd equation: \(\theta_2(i) = x_{2i} b\)
Programming Likelihood Functions: Syntax

• Global macros:
  $ML_y1 is a global macro for the name of the first dependent variable.
  $ML_y2 is a global macro for the name of the second dependent variable.
Onto the Machines: Start a .log File

Click here to start .log file.
Onto the Machines: Start a .log File
Programming Likelihood Functions:
ML Normal Regression Program

• Let’s open some data: 1992 NES
• File, Open
  Go to the I:\ drive
  Double-click on “general”
    Double-click on “PRISM Programming”
      Double-click on “NES 1992.dta”
Programming Likelihood Functions:
ML Normal Regression Program

Click here to open new .do file.
Programming Likelihood Functions:
ML Normal Regression Program

- Open “normreg” program from the .do file editor:
  Go to the I: drive
  Double-click on “general”
  Double-click on “PRISM Programming”
  Double-click on “normreg.do”
Programming Likelihood Functions: ML Normal Regression Program

Recall:

\[
\ln L_i = -\frac{1}{2} \ln(2\pi) - \frac{1}{2} \ln(\sigma^2) - \frac{1}{2} \left[ \frac{(y_i - x_i \beta)^2}{\sigma^2} \right]
\]

Run the program
Programming Likelihood Functions: Maximizing the Likelihood Function

• Once we’ve written the program, we need to tell Stata to estimate it. This takes two steps:

(1) `ml model lf progname (eq1: y=x1 x2 x3)`
    - or -
    `ml model lf progname (eq1: y=x1 x2 x3) (eq2: y=x1 x2 x3)`
    - or -
    `ml model lf progname (eq1: y=x1 x2 x3) / parameter`
    [If the second parameter is not reparameterized as a function of covariates, e.g., $s^2$ in ML normal regression.]

(2) `ml max`
• Other useful commands to run after `ml model`:
  
  `ml check` verifies that the program you wrote works
  `ml search` searches for better starting values
  `lf0(# k L L 0)` reports a likelihood ratio test (included after the “ml model” command), comparing fully specified model to an intercept only (i.e., null) model. The Wald test is produced by default. For the LR test, you need to specify the LL and the number of parameters for the intercept only model.
Programming Likelihood Functions: Estimating the ML Normal Model

- Let’s estimate a simple model; we’ll regress George H.W. Bush’s approval on PID and economic perceptions.
Programming Likelihood Functions: Estimating the ML Normal Model

ml model lf normreg (reg: bush_app = pid econworse) /sigmasq
Programming Likelihood Functions:
Estimating the ML Normal Model

Maximizes the function and produces model estimates.
Programming Likelihood Functions: Estimating the ML Normal Model

Log likelihood = -476.32127

| bush_app  | Coef. | Std. Err. | z   | P>|z| | [95% Conf. Interval] |
|-----------|-------|-----------|-----|-----|---------------------|
| reg       | .2781664 | .0220503 | 12.62 | 0.000 | .2349487 to .3213941 |
|           | - .245331 | .05101 | -4.81 | 0.000 | -.3453087 to -.1453533 |
|           | 3.22732 | .2080806 | 15.51 | 0.000 | 2.81949 to 3.635151 |
| sigma2    | .7088946 | .0512938 | 13.82 | 0.000 | .6093606 to .8094286 |
Programming Likelihood Functions: Comparing ML Reg to OLS

• OLS and ML Normal Regression produce identical parameter estimates. It can be shown that the analytical solution for ML Normal Regression is:

$$\hat{\beta} = (X'X)^{-1}X'Y$$

which is identical to the well-known formula for the OLS estimator.

• Standard errors will be different, though, because:
  - In ML:
    $$\sigma^2 = \frac{\sum_{i=1}^{N} e_i^2}{N}$$
  - In OLS:
    $$\sigma^2 = \frac{\sum_{i=1}^{N} e_i^2}{N - k}$$
Programming Likelihood Functions: Comparing ML Reg to OLS

Estimate this OLS.

```
reg bush_app pid econworse
```
Programming Likelihood Functions: Comparing ML Reg to OLS

Coefficients are identical, SEs are different.
In binary response models, we want to model the probability of “success” for case \( i \), i.e., \( \Pr(y_i = 1) = \) ?

We parameterize ? as a cumulative distribution function (cdf) of a particular distribution, i.e., \( F(x_i \beta) \)

- For logit, we use the logistic cdf:

\[
\Pr(y_i = 1) = F(x_i \beta) = \frac{\exp(x_i \beta)}{1 + \exp(x_i \beta)}
\]

- For probit, we use the normal cdf:

\[
\Pr(y_i = 1) = F(x_i \beta) = \Phi(x_i \beta)
\]
Programming Likelihood Functions: Logit and Probit

- The likelihood for case $i$ is:
  \[ L_i = [F(x_i \beta)]^{y_i} [1 - F(x_i \beta)]^{1-y_i} \]

- The log-likelihood for case $i$ is:
  \[ \ln L_i = y_i \ln[F(x_i \beta)] + (1 - y_i) \ln[1 - F(x_i \beta)] \]

  **Again, this is what we’re going to give Stata**

- For logit, we’ll replace $F( x_i \beta )$ with the logistic cdf, and for probit, the normal cdf.
Programming Likelihood Functions: Logit and Probit

• Open “mylogit.do” from the .do file editor.
  Go to the I: drive
  Double-click on “general”
  Double-click on “PRISM Programming”
  Double-click on “mylogit.do”
Programming Likelihood Functions: Logit and Probit

Recall:
\[
\ln L_i = y_i \ln[F(x_i \beta)] + (1 - y_i) \ln[1 - F(x_i \beta)]
\]

Run the program
Programming Likelihood Functions: Logit and Probit

```
. do "C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\STD040006.tmp"
.
. program define mylogit
1.    args lnf thetal
2.    /* only one parameter here, \pi, which we reparameterize as \chi_1 \beta */
3.    tempvar p
4.    /* creating this temp var makes organisation a little easier. Here, "p" is going
8.     to stand for \Phi(x), i.e., the logistic cdf */
5.    quietly gen double p=exp(`thetal')/(1+exp(`thetal'))
6.    /* we define F(x) as the logistic cdf */
7.    quietly replace `lnf'=$ML_y1*ln(p)+(1-$ML_y1)*ln(1-p')
8.    /* now, we just put \pi in where the cdf is defined in the lnl */
9. end
.
.
ml model lf mylogit (logit: vote2 = pid econworse)
```
Programming Likelihood Functions: Logit and Probit

Maximizes the function and produces model estimates.
Programming Likelihood Functions: Logit and Probit
Programming Likelihood Functions: Logit and Probit
Programming Likelihood Functions: The Likelihood Ratio Test

• By default, “ml model” produces a Wald test for overall goodness of fit test (which tests that the coefficients are jointly equal to zero).

• To get an LR test, we need to:
  – Estimate an intercept only model to get LL0, the initial LL.
  – We need to specify \(k\) for the intercept-only model, which in this case is 1.
  – After the “ml model” command, we enter \(lf0(k LL0)\).
Programming Likelihood Functions:  
The Likelihood Ratio Test

Estimates intercept only model
Programming Likelihood Functions:
The Likelihood Ratio Test
Programming Likelihood Functions:
The Likelihood Ratio Test

```
Command: \texttt{ml model lf mylogit (logit: vote2 = pid econworse), lf0(1 -254.0798)}
```

**Output:**

```
Log likelihood = -254.0798

| vote2 | Coef.       | Std. Err. | z     | P>|z|  | [95% Conf. Interval] |
|-------|-------------|-----------|-------|------|----------------------|
| pid   | .8437178    | .0878516  | 9.60  | 0.000| .6715318 - 1.0153904 |
| econworse | -.3076029  | .1606638  | -1.91 | 0.056| -.6242982 .0129923   |
| _cons | .2923819    | .6254885  | 0.45  | 0.656| -.992346 .157711     |

LR chi2(2) = 183.36
Prob > chi2 = 0.0000
Pseudo R2 = 0.3606
```

**Variables:**

- Target: \texttt{Command}
- vote3
- bush_app
- ideology
- econworse
- military__oppp
- gulfwar
- pid
- education
- govttemp
- union
- income
- nonwhite
- vote2
Programming Likelihood Functions: The Likelihood Ratio Test

```
. ml model lf mylogit (logit: vote2 = pid econworse), lf0(1 -254.0798)
. ml max

initial: log likelihood = -264.70222
alternative: log likelihood = -254.09741
rescale: log likelihood = -254.09741
Iteration 0: log likelihood = -254.09741
Iteration 1: log likelihood = -166.51063
Iteration 2: log likelihood = -162.49286
Iteration 3: log likelihood = -162.45214
Iteration 4: log likelihood = -162.45212
Iteration 5: log likelihood = -162.45212

Log likelihood = -162.45212

Number of obs = 382
LR chi2(2) = 183.46
Prob > chi2 = 0.0000

| vote2  | Coef. | Std. Err. | z    | P>|z|  | [95% Conf. Interval] |
|--------|-------|-----------|------|------|----------------------|
| _cons | -0.4002252 | 0.1052929 | -4.56 | 0.000 | -0.6865956 to -0.2739548 |
| pid   | 0.8437178  | 0.0878516  | 9.60  | 0.000 | 0.6715318 to 1.015904   |
| econworse   | -0.3076029 | 0.1606638  | -1.91  | 0.056 | -0.6224982 to 0.0172923 |
| _cons | 0.2923819  | 0.6548555  | 0.45  | 0.656 | -0.992346 to 1.57711    |
```
Programming Likelihood Functions: Probit Program

```
program define myprobit
    args lnf thetal
    tempvar p
    quietly gen double `p'=norm(`thetal')
    quietly replace `lnf'=ML_y1*ln(`p')+(1-ML_y1)*ln(1-`p')
end
```
Programming Likelihood Functions: Heteroskedastic Regression

• Heteroskedastic regression allows us to model the factors that influence both the expected value of $Y$ and the factors that affect the variability around that expected value (see Franklin 1991; Alvarez and Brehm 1995, 1997, 1998).

• In regression, we always assume homoskedasticity: $s^2$

• With het. reg., we’re explicitly interested in modeling the factors that influence $\sigma_i^2$.

• Good pedagogical example: it’s more complicated, it generates two sets of simultaneously generated coefficients. But, bottom line: all you have to do is know the likelihood function, and you can program it in Stata.
Programming Likelihood Functions: Heteroskedastic Regression

• We parameterize $\sigma_i^2$ as:

$$\sigma_i^2 = e^{z_i \gamma}$$

• Where the $z_i$’s exogenous variables that influence the variability around the expected value, and gamma is a vector of parameters.

• Log-likelihood:

$$\ln L_i = -\frac{1}{2} \ln(2\pi) - \frac{1}{2} z_i \gamma - \frac{1}{2} \left[ \frac{(y_i - x_i \beta)^2}{e^{z_i \gamma}} \right]$$
Programming Likelihood Functions: Heteroskedastic Regression

\[
\ln L_i = -\frac{1}{2} \ln(2\pi) - \frac{1}{2} z_i \gamma - \frac{1}{2} \left[ \frac{(y_i - x_i \beta)^2}{e^{z_i \gamma}} \right]
\]
Programming Likelihood Functions: Heteroskedastic Regression

```stata
. ml model lf hreg (slopes: reppct00 = reppct96 logvotes adsgore adsbush) (var: volatile)
```

```
     48-student Stata for Windows (network) perpetual license:
     Serial number: 198053842
     Licensed to: Ohio State University
     Ohio State University

     Notes:
     1. (/m# option or -set memory-) 1.00 MB allocated to data

     use "Ki\MLE Franklin\Exercise 2b\vote2000.dta", clear
     (written by A.)
     do "C:\DOCUMENTS-1\ADMINI-1\LOCALS-1\Temp\STD040000tmp"
     program define hreg
     1.     \args lnf theta1 theta2
     2.     \quietly replace \lnf\=-.5*ln(2*_pi)-.5*(\theta22)-.5*(\$ML_y1-\theta1)^2/exp(\theta22)
     3.     \end
     end
```

```
Statistics/Data Analysis
```

```
Copyright 1984-2003
Stata Corporation
4905 Lakeway Drive
College Station, Texas 77845 USA
800-STATACORP http://www.stata.com
979-696-4600 979-696-4601 (fax)
```
Programming Likelihood Functions: Heteroskedastic Regression
Programming Likelihood Functions: Split-Population Duration Model

• Standard duration models, which model the hazard of an event occurring, assume that all cases will eventually experience the event of interest.

• This assumption may not hold for the process under examination; if not, will produce incorrect inferences.

  - The timing of congressional overrides of Supreme Court decisions (Hettinger and Zorn 2004).
  - Corporate and labor PAC contributions to congressional candidates (Box-Steffensmeier et al. 2004).
Programming Likelihood Functions: Split-Population Duration Model

- The split population duration (SPD) model relaxes the assumption that all cases will eventually experience the event of interest (Schmidt and Witte 1988, 1989; Forster and A. Jones 2001; Box-Steffensmeier and B. Jones 2004).
- Simultaneously estimates two sets of coefficients:
  1. Explaining the likelihood of the event occurring (i.e., the censoring indicator is the DV).
  2. Explaining the timing of the event occurring, conditional on the event having occurred in the first place.
Programming Likelihood Functions: Split-Population Duration Model

- LIMDEP is the only package that has a canned routine for the SPD. Great example of an advanced model that hasn’t made its way into a lot of stat packages. But you can program it yourself!
- Acknowledgements to Forster and Jones...
Programming Likelihood Functions: Split-Population Duration Model

\[
\ln L_i = R_i [\ln \delta_i + \ln g(t_i, \theta)] + (1 - R_i) \ln [1 - \delta_i + \delta_i G(t_i, \theta)]
\]

censoring indicator   pdf   survivor function
Programming Likelihood Functions: Split-Population Duration Model

• Example: Explaining the incidence and timing of labor PAC contributions to incumbent House members, 1993-1994 (Box-Steffensmeier et al. 2004).

• We’re interested in the timing of contributions in an election cycle. Early money is “seed money” for a campaign effort, and it helps candidates raise more down the line (Jacobson 1992).

• We don’t expect labor PACs to contribute to every House incumbent, though. E.g., people trying to reform OSHA, or investigating the Teamsters.
Programming Likelihood Functions: Split-Population Duration Model

program define splitpop
 1. args lnL theta1 theta2 theta3
 2. /* theta1: Xβ for duration equation; theta2: Xβ for logit equation; theta3: shape parameter (delta) */
 3. tempvar p s d l
 4. quietly gen double `l' = exp(-`theta1')
 5. /* lambda of log-logistic distribution */
 6. quietly gen double `d' = exp(`theta2')/(1+exp(`theta2'))
 7. /* logistic cdf for probability of failure */
 8. quietly gen double `s' = 1-`d'+`d'*exp(`l'*(1`theta3'))
 9. /* survival function */
 10. quietly gen double `p' = ln(`d') - ln(`theta3') + (1`theta3')*ln(`s')
 11. /* pdf */
 12. quietly replace `lnL' = `s'*(`p') + (`s')*ln(`s')

end

ml model lf splitpop (duration: T = educatn prestige seniority dleader rleader republic cope votept quality pquality bcash lagprecp repsqrd totpt stshare homecand dcpac) (gave = educatn prestige seniority dleader rleader republic cope votept quality pquality bcash lagprecp repsqrd totpt stshare homecand dcpac) /shape if large==1
Programming Likelihood Functions: Split-Population Duration Model

Searches for better starting values
Programming Likelihood Functions:
Split-Population Duration Model

```
args lnf theta1 theta2 theta3
theta1: shape parameter (delta) */
theta2: shape parameter (delta) */
theta3: shape parameter (delta) */

tempvar p s d l

quietly gen double `l' = exp(`theta1') /* lambda of log-logistic distribution */
quietly gen double `d' = exp(`theta2')/(1+exp(`theta2')) /* logistic cdf for probability of failure */
quietly gen double `s' = 1^-`d' + `d'^*(1/(1+(1-2*`SML_y2')^(-1/~theta3')))) /* survival function */
quietly gen double `p' = ln(`d') - ln(`theta3') + ((1/~theta3') - 1)*ln(SML_y1) + (1/~theta3')*ln(`SML_y2') /* pdf */
quietly replace `lnf' = `SML_y2'*(`p') + (1-`SML_y2')*ln(`s')

end
```

```
miv model if splitpop (duration: T = educatn prestige seniority dleader rleader republic cope votepc quailty equality bcash lagprecp recpgd totpc stshare homeand depac) (gave = educatn prestige seniority dleader rleader republic cope votepc quailty equality bcash lagprecp recpgd totpc stshare homeand depac) /shape if large==1
```

```
miv search
initial: log likelihood = -lnf (could not be evaluated)
reliable: log likelihood = -73241.663
improve: log likelihood = -61256.289
rescale: log likelihood = -49900.402
rescale eq: log likelihood = -48914.638
```
Programming Likelihood Functions: Split-Population Duration Model
Programming Likelihood Functions: The Predict Command and Est. Split
Calculates probability of the exchange of a contribution.
Programming Likelihood Functions: The Predict Command and Est. Split

```
. predict xb, eq(logit)
. gen prob=exp(xb)/(1+exp(xb))
. sum prob

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>prob</td>
<td>10168</td>
<td>0.4719391</td>
<td>0.317901</td>
<td>0.0084166</td>
<td>0.9725916</td>
</tr>
</tbody>
</table>
```

```
. sum gave

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>gave</td>
<td>10168</td>
<td>0.4353855</td>
<td>0.4958318</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Observed split
Programming Likelihood Functions: Conclusion

• **Bottom line:**
  - If you need to estimate a model that is not canned in a popular software package, you can probably program it in Stata.
  - All you need to know is the likelihood function!