Estimating user-defined nonlinear regression models in Stata and in Mata

A. Colin Cameron
Univ. of Calif. - Davis

Based on A. Colin Cameron and Pravin K. Trivedi, Microeconometrics using Stata, Stata Press.

November 14, 2008
Consider nonlinear cross-section regression of $y_i$ on $x_i$.

Example is $y_i \mid x_i \sim \text{Poisson with mean } \mu_i = \exp(x_i'\beta)$.

This talk demonstrates various ways to code up the estimator,

- using Stata command \texttt{ml}
- and Mata command \texttt{optimize}
Outline

1. Introduction
2. Built-in command poisson
3. Command ml method lf
4. Checking program by simulation
5. Command ml methods d0, d1, d2
6. Newton-Raphson algorithm in Mata
7. Mata command optimize
8. NL2SLS example
2. Built-in command poisson

- Data from 2002 U.S. Medical Expenditure Panel Survey (MEPS). Data due to Deb, Munkin and Trivedi (2006)
- Aged 25-64 years working in private sector but not self-employed and not receiving public insurance (Medicare and Medicaid)
- Model docvis - annual number of doctor visits.
. use mus10data.dta, clear
. quietly keep if year02==1
. describe docvis private chronic female income

<table>
<thead>
<tr>
<th>variable name</th>
<th>type</th>
<th>format</th>
<th>label</th>
<th>value label</th>
</tr>
</thead>
<tbody>
<tr>
<td>docvis</td>
<td>int</td>
<td>%8.0g</td>
<td></td>
<td>number of doctor visits</td>
</tr>
<tr>
<td>private</td>
<td>byte</td>
<td>%8.0g</td>
<td></td>
<td>= 1 if private insurance</td>
</tr>
<tr>
<td>chronic</td>
<td>byte</td>
<td>%8.0g</td>
<td></td>
<td>= 1 if a chronic condition</td>
</tr>
<tr>
<td>female</td>
<td>byte</td>
<td>%8.0g</td>
<td></td>
<td>= 1 if female</td>
</tr>
<tr>
<td>income</td>
<td>float</td>
<td>%9.0g</td>
<td></td>
<td>Income in $ / 1000</td>
</tr>
</tbody>
</table>

. summarize docvis private chronic female income

<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>docvis</td>
<td>4412</td>
<td>3.957389</td>
<td>7.947601</td>
<td>0</td>
<td>134</td>
</tr>
<tr>
<td>private</td>
<td>4412</td>
<td>.7853581</td>
<td>.4106202</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>chronic</td>
<td>4412</td>
<td>.3263826</td>
<td>.4689423</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>female</td>
<td>4412</td>
<td>.4718948</td>
<td>.4992661</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>income</td>
<td>4412</td>
<td>34.34018</td>
<td>29.03987</td>
<td>-49.999</td>
<td>280.777</td>
</tr>
</tbody>
</table>

A. Colin Cameron  Univ. of Calif. - Davis  User-defined models in Stata
November 14, 2008 5 / 36
**Built-in command poisson**

```stata
. poisson docvis private chronic female income, vce(robust)
```

Iteration 0: log pseudolikelihood = -18504.413
Iteration 1: log pseudolikelihood = -18503.549
Iteration 2: log pseudolikelihood = -18503.549

Poisson regression

| docvis     | Coef. | Robust Std. Err. | z    | P>|z| | [95% Conf. Interval] |
|------------|-------|------------------|------|-----|---------------------|
| private    | .7986652 | .1090014         | 7.33 | 0.000 | .5850263 -.1012304 |
| chronic    | 1.091865 | .0559951         | 19.50 | 0.000 | .9821167 1.201614 |
| female     | .4925481 | .0585365         | 8.41 | 0.000 | .3778187 .6072774 |
| income     | .003557  | .0010825         | 3.29 | 0.001 | .0014354 .0056787 |
| _cons      | -.2297262 | .1108732        | -2.07 | 0.038 | -.4470338 -.0124186 |

Note: Nonrobust standard errors are (erroneously) much smaller.
Marginal effects for nonlinear model: \( \frac{\partial E[y|x]}{\partial x_j} = \beta_j \times \exp(x'\beta) \).

\[ \text{mfx} \]

Marginal effects after poisson
\( y = \text{predicted number of events (predict)} \)
\( X = 3.0296804 \)

| variable  | dy/dx   | Std. Err. | z    | P>|z|  | [ 95% C.I. ] | X     |
|-----------|---------|-----------|------|-------|--------------|-------|
| private*  | 1.978178| 0.20441   | 9.68 | 0.000 | 1.57755      | 2.37881| .785358|
| chronic*  | 4.200068| 0.27941   | 15.03| 0.000 | 3.65243      | 4.7477 | .326383|
| female*   | 1.528406| 0.17758   | 8.61 | 0.000 | 1.18036      | 1.87645| .471895|
| income    | 0.0107766| .00331   | 3.25 | 0.001 | .00428       | .017274| 34.3402|

(*) dy/dx is for discrete change of dummy variable from 0 to 1

\[ \text{. margeff} \]

Average marginal effects on \( E(docvis) \) after poisson

| docvis | Coef.   | Std. Err. | z    | P>|z|  | [95% Conf. Interval] |
|--------|---------|-----------|------|-------|----------------------|
| private | 2.404721| .2438573  | 9.86 | 0.000 | 1.926769           | 2.882672|
| chronic | 4.599174| .2886176  | 15.94| 0.000 | 4.033494           | 5.164854|
| female  | 1.900212| .2156694  | 8.81 | 0.000 | 1.477508           | 2.322917|
| income  | 0.0140765| .004346  | 3.24 | 0.001 | .0055585           | .0225945|
3. Command `ml` method `lf`

- First write a program we call `lfpois`. This constructs the log-likelihood

\[
\sum_{i=1}^{N} \ln f(y_i | x_i, \beta) = \sum_{i=1}^{N} \{- \exp(x'_i \beta) + y_i x'_i \beta - \ln y_i!\}.
\]

- Then give commands
  - `ml model lf lfpois (docvis = private chronic female income), vce(robust)`
  - `ml check`
  - `ml search`
  - `ml maximize`

  The `ml check` and `ml search` are optional.
1. $y$ is stored in global macro $\text{ML}_y1$. It is referred to as $\$\text{ML}_y1$

2. $x$ is combined with $\beta$ as the index $x'\beta$
   It is referred to as the program argument $\theta_1$

3. $\ln f(y|x, \beta)$ is referred to as the program argument $\text{lf}$

```
. program define lfpois
   1.   version 10.0
   2.   args lnf theta1           // $\theta_1=x'b$, $\lnf=\ln f(y)$
   3.   tempvar lnyfact mu
   4.   local y "$\text{ML}_y1$"       // Define $y$ so program more readable
   5.   generate double `lnyfact' = `lnfactorial(`y')
   6.   generate double `mu' = exp(`theta1')
   7.   quietly replace `lnf' = -`mu' + `y'*`theta1' - `lnyfact'
   8.   end
```

Arguments, temporary variables and local variables are local macros, referenced in single quotes.


.m * Compute the estimator
.ml maximize

initial: log pseudolikelihood = -23017.072
rescale: log pseudolikelihood = -23017.072
Iteration 0: log pseudolikelihood = -23017.072
Iteration 1: log pseudolikelihood = -19777.405
Iteration 2: log pseudolikelihood = -18513.54
Iteration 3: log pseudolikelihood = -18503.556
Iteration 4: log pseudolikelihood = -18503.549
Iteration 5: log pseudolikelihood = -18503.549

Number of obs = 4412
Wald chi2(4) = 594.72
Prob > chi2 = 0.0000

Log pseudolikelihood = -18503.549

| docvis    | Coef.     | Robust Std. Err. | z    | P>|z| | 95% Conf. Interval |
|-----------|-----------|------------------|------|------|-------------------|
| private   | .7986654  | .1090015         | 7.33 | 0.000 | .5850265 - 1.012304 |
| chronic   | 1.091865  | .0559951         | 19.50| 0.000 | .9821167 - 1.201614 |
| female    | .4925481  | .0585365         | 8.41 | 0.000 | .3778187 - 1.6072775 |
| income    | .003557   | .0010825         | 3.29 | 0.001 | .0014354 - .0056787 |
| _cons     | -.229726  | .1108733         | -2.07| 0.038 | -.4470339 - .0124188 |
Command `ml` is not restricted to likelihood functions. e.g. For OLS maximize $-\sum_{i=1}^{N} (y_i - x_i' \beta)^2$.
quietly replace ‘lnf’ = -(‘y’-exp(‘theta1’))^2
But must then use robust standard errors.

Command `ml` can handle models with more than one index. e.g. For negative binomial have two indexes $x_i' \beta$ and $\alpha$.
args lnf theta1 a
and
ml model lf lfnb (docvis = private chronic female income) ()

Number of numerical derivatives = number of indexes. Fast if few indexes.
4. Check program by simulation

- Generate sample of size $N$ from

\[ y_i \sim \text{Poisson} \left[ \exp(\alpha + \beta x_i) \right] \]
\[ x_i \sim \text{N}[0, 0.5^2] \]
\[ \alpha = 2; \beta = 1. \]

- To check consistency
  - Set $N = 100,000$
  - Does $\hat{\alpha} = 1$? Does $\hat{\beta} = 1$?
To check computation of the standard errors $s_{\hat{\alpha}}$ and $s_{\hat{\beta}}$.

- Set $N = 500$.
- Draw 2,000 samples of size $N$ and obtain 2,000 estimates using command `simulate` or command `postfile`.
- Does
  \[
  \sqrt{\frac{1}{1999} \sum_{s=1}^{2000} (\hat{\beta}^{(s)} - \bar{\hat{\beta}})^2} = \frac{1}{2000} \sum_{s=1}^{2000} s_{\hat{\beta}}^{(s)}?
  \]
- i.e. Over the simulations does the st. deviation of $\hat{\beta} = \text{the average st. error of } \hat{\beta}$?
5. Command ml methods d0, d1, d2

- More general.
- Computes the log-density for each observation. This then needs to be summed using mlsum.
- Enters parameters $\beta$ directly, rather than via index $x'\beta$.
- Method d0 needs to compute $q$ numerical derivatives if $q$ parameters.
- Can provide first derivatives (method d1) and second derivatives (method d2).
  This speeds up computation.
- For method d0 extra arguments is todo
- mleval converts \( \beta \) to \( x'\beta \)
- mlsum converts \( x_i\beta \) to \( \sum_{i=1}^{N} x_i\beta \).

```
. * Method d0: Program d0pois to be called by command ml method d0
. program define d0pois
  1.   version 10.0
  2.   args todo b lnf               // todo is not used, b=b, lnf=lnL
  3.   tempvar theta1
       // theta1=x'b given in eq(1)
  4.   mleval `theta1' = `b', eq(1)
  5.   local y $ML_y1        // Define y so program more readable
  6.   mlsum `lnf' = -exp(`theta1') + `y'*`theta1' - lnfactorial(`y')
  7.   end
```
. ml model d0 d0pois (docvis = private chronic female income)
. ml maximize

| Coef.   | Std. Err. | z     | P>|z| | [95% Conf. Interval] |
|---------|-----------|-------|------|----------------------|
| private | .7986653  | .027719 | 28.81 | 0.000 | .7443371 - .8529936 |
| chronic | 1.091865  | .0157985 | 69.11 | 0.000 | 1.060901 - 1.12283  |
| female  | .4925481  | .0160073 | 30.77 | 0.000 | .4611744 - .5239218 |
| income  | .003557   | .0002412 | 14.75 | 0.000 | .0030844 - .0040297 |
| _cons   | -.2297263 | .0287022 | -8.00 | 0.000 | -.2859815 -.173471  |

Log likelihood = -18503.549

Number of obs = 4412
Wald chi2(4)  = 8052.34
Prob > chi2   = 0.0000
• Preceding gives nonrobust standard errors.
• To get robust standard errors need to use method d1 or d2.

. * Method d2: Program d2pois to be called by command ml method d2
. program define d2pois
  1.  version 10.0
  2.  args todo b lnf g negH         // Add g and negH to the arguments list
  3.  tempvar theta1              // theta1 = x'b where x given in eq(1)
  4.  mleval `theta1' = `b', eq(1)
  5.  local y $ML_y1              // Define y so program more readable
  6.  mlsum `lnf' = -exp(`theta1') + `y'*`theta1' - lnfactorial(`y')
  7.  if (`todo'=0 | `lnf'>=.) exit  // d1 extra code from here
  8.  tempname d1
  9.  mlvecsum `lnf' `d1' = `y' - exp(`theta1')
10.  matrix `g' = (`d1')
11.  if (`todo'=0 | `lnf'>=.) exit  // d2 extra code from here
12.  tempname d11
13.  mlmatsum `lnf' `d11' = exp(`theta1')
14.  matrix `negH' = `d11'
15.  end
Iterative algorithms are rules to compute $\hat{\theta}_{s+1}$ given $\hat{\theta}_s$.

Gradient methods use a rule of the form

$$\hat{\theta}_{s+1} = \hat{\theta}_s + A_s g_s$$

where $g_s$ is the gradient of the objective function evaluated at $\hat{\theta}_s$.

Newton-Raphson (NR) method approximates the objective function at $\hat{\theta}_s$ by a quadratic function.
It chooses $\hat{\theta}_{s+1}$ to maximize this approximation.

Then

$$\hat{\theta}_{s+1} = -H_s^{-1} g_s$$

where $H_s$ is the Hessian evaluated at $\hat{\theta}_s$. 
Poisson objective function, gradient and Hessian are:

\[ Q(\beta) = \sum_{i=1}^{N} \{ -\exp(x_i'\beta) + y_i x_i' \beta - \ln y_i! \} \]

\[ g(\beta) = \sum_{i=1}^{N} (y_i - \exp(x_i'\beta)) x_i \]

\[ H(\beta) = \sum_{i=1}^{N} \exp(x_i'\beta) x_i x_i' \]

So NR is

\[
\hat{\beta}_{s+1} = \hat{\beta}_s - H(\hat{\beta}_s)^{-1} \times g(\hat{\beta}_s)
\]

\[
= \hat{\beta}_s + \left[ \sum_{i=1}^{N} \exp(x_i'\hat{\beta}_s) x_i x_i' \right]^{-1} \times \sum_{i=1}^{N} (y_i - \exp(x_i'\hat{\beta}_s)) x_i.
\]
Core Mata code is

```mata
> mata
> cha = 1                  // initialize stopping criterion
> do {
>     mu = exp(x*b)
>     grad = X'(y-mu)       // kx1 gradient vector
>     hes = makesymmetric((X:*mu)'X) // negative of the kxk hessian matrix
>     bold = b
>     b = bold + cholinv(hes)*(grad)
>     cha = (bold-b)'(bold-b)/(b'b)
>     iter = iter + 1
> } while (cha > 1e-16)     // end of iteration loops
> end
```
Define $y$ and $x$

generate cons = 1
local y docvis
local xlist private chronic female income cons

Read these in to Mata using `st_view`
:  `st_view(y=., ., "'y'")`
:  `st_view(X=., ., tokens("'xlist'"))`

Do the analysis and compute $b$ and $V$

Pass these back to Stata using `st_matrix`
`st_matrix("b",b')`
`st_matrix("V",vb)`

Post results using command `ereturn`
Do the NR iterations to compute \( \hat{\beta} \).

```plaintext
* Complete Mata code for Poisson MLE NR iterations
.mata

: st_view(y=., ., "\`y'") // read in stata data to y and X
: st_view(X=., ., tokens("\`xlist'"))

: b = J(cols(X),1,0) // compute starting value
: n = rows(X)

: iter = 1 // initialize number of iterations
: cha = 1 // initialize stopping criterion

: do {
>   mu = exp(X*b)
>   grad = X'(y-mu) // kx1 gradient vector
>   hes = makesymmetric((X:*mu)'X) // negative of the kxk hessian matrix
>   bold = b
>   b = bold + cholinv(hes)*(grad)
>   cha = (bold-b)'(bold-b)/(b'b)
>   iter = iter + 1
> } while (cha > 1e-16) // end of iteration loops
```
Compute the variance-covariance matrix of $\hat{\beta}$.

```stata
: mu = exp(x*b)
: hes = (x:*mu)'x
: vgrad = ((x:*y-mu))'(x:*y-mu))
: vb = cholinv(hes)*vgrad*cholinv(hes)*n/(n-cols(X))
: iter 13 // num iterations
: cha 1.11465e-24 // stopping criterion
: st_matrix("b",b') // pass results from Mata to Stata
: st_matrix("V",vb) // pass results from Mata to Stata
: end
```
Present results nicely formatted.

```stata
. * Present results, nicely formatted using Stata command ereturn
.matrix colnames b = `xlist'
.matrix colnames V = `xlist'
.matrix rownames V = `xlist'
. ereturn post b V
. ereturn display
```

| Coef.  | Std. Err. | z    | P>|z|  | [95% Conf. Interval] |
|--------|-----------|------|------|----------------------|
| private| .7986654  | .1090509 | 7.32 | 0.000 | .5849295 | 1.012401 |
| chronic| 1.091865  | .0560205 | 19.49 | 0.000 | .9820669 | 1.201663 |
| female | .4925481  | .058563  | 8.41  | 0.000 | .3777666 | .6073295 |
| income | .003557   | .001083  | 3.28  | 0.001 | .0014344 | .0056796 |
| cons   | -.2297263 | .1109236 | -2.07 | 0.038 | -.4471325 | -.0123202 |
```
7. Mata command optimize

- Mata command optimize uses same optimizer as command ml, but different syntax.

- Minimal syntax is
  
  \[
  \text{void evaluator(todo, p, v, g, H)}
  \]

  where
  
  - \(p\) is parameter vector
  - \(v\) defines objective function, and
  - if \(todo = 0\) then gradient \(g\) and Hessian \(H\) are optional.

- Type \(v\) evaluator provides formula for \(1 \times N\) vector \(v\), where

  \[
  e'v = f(p)
  \]

  Suited to m-estimators (MLE, LS, just-identified NLIV).

- Type \(d\) evaluator provides formula for scalar \(v\) where

  \[
  v = f(p)
  \]

  Suited to over-identified generalized method of moments (GMM).
Declare the function poissonmle and st_view data

```
.mata
void poissonmle(todo, b, y, X, lndensity, g, H)
{
    xb = X*b'
    mu = exp(xb)
    lndensity = -mu + y:*xb - lnfactorial(y)
    if (todo == 0) return
    g = (y-mu)*X
    if (todo == 1) return
    H = - cross(X, mu, X)
}

st_view(y=., ., "\'y'")
st_view(X=., ., tokens("\'xlist'"))
```
Initialize command optimize and optimize using v2 evaluator.

```plaintext
: S = optimize_init()
: optimize_init_evaluator(S, &poissonmle())
: optimize_init_evaluatorevaltype(S, "v2")
: optimize_init_argument(S, 2, X)
: optimize_init_argument(S, 1, y)
: optimize_init_params(S, J(1,cols(X),0))
: b = optimize(S)
Iteration 0:  f(p) = -33899.609
Iteration 1:  f(p) = -19668.697
Iteration 2:  f(p) = -18585.609
Iteration 3:  f(p) = -18503.779
Iteration 4:  f(p) = -18503.549
Iteration 5:  f(p) = -18503.549
```
Compute variance covariance matrix and list results.

: \vbrob = \text{optimize}\_result\_V\_robust(S)
: \serob = (\sqrt{\text{diagonal}(\vbrob)})'
: \text{b} \ \text{\serob}

\begin{tabular}{c c c c c}
1 & 2 & 3 & 4 & 5 \\
1 & .7986653788 & 1.091865108 & .4925480693 & .0035570127 & -.2297263376 \\
2 & .1090014507 & .0559951312 & .0585364746 & .0010824894 & .1108732568 \\
\end{tabular}

: \text{end}

\textbf{Note: Can st_matrix back to Stata and ereturn display results.}
8. NL2SLS example

- Poisson MLE inconsistent if \( E[y - \exp(x'\beta)|x] \neq 0 \), due to endogenous regressors.
- Assume there are instruments \( z \) such that
  \[
  E[z_i(y_i - \exp(x'_i\beta))] = 0.
  \]
- Define the \( r \times 1 \) vector
  \[
  h(\beta) = \left[ \sum_i z_i(y_i - \exp(x'_i\beta)) \right].
  \]
- In just-identified case: \( \# \) instruments = \( \# \) regressors \( (r = K) \)
  use the nonlinear instrumental variabels (NLIV) estimator that solves
  \[
  h(\hat{\beta}) = 0.
  \]
- In over-identified case \( (r > K) \) the GMM estimator minimizes
  \[
  Q(\beta) = h(\beta)'Wh(\beta).
  \]
GMM estimator minimizes

\[ Q(\beta) = h(\beta)'Wh(\beta). \]

The \( K \times 1 \) gradient vector is

\[ g(\beta) = \frac{\partial Q(\beta)}{\partial \beta} = G(\beta)'Wh(\beta). \]

The \( K \times K \) expected Hessian is

\[ H(\beta) = \frac{\partial^2 Q(\beta)}{\partial \beta \partial \beta'} = G(\beta)'WG(\beta)'. \]

Where

\[
\begin{align*}
G(\beta) & = - \sum_i \exp(x_i'\beta)z_ix_i' \\
h(\beta) & = \sum_i z_i(y_i - \exp(x_i'\beta)) \\
W & = (Z'Z)^{-1} = \left(\sum_i z_izi_i'\right)^{-1}
\end{align*}
\]
Declare the function pgmm and st_view data

```plaintext
. mata
: void pgmm(todo, b, y, X, Z, Qb, g, H)
  
  { 
    Xb = X*b'
    mu = exp(Xb)
    h = Z'(y-mu)
    W = cholinv(cross(Z,Z))
    Qb = h'W*h
    if (todo == 0) return
    G = -(mu:*Z)'X
    g = (G'W*h)'
    if (todo == 1) return
    H = G'W*G
  _makesymmetric(H)
  }

  st_view(y=., ., "y'")
  st_view(X=., ., tokens("xlist'"))
  st_view(Z=., ., tokens("zlist'"))
  mata (type end to exit)
```

A. Colin Cameron  Univ. of Calif. - Davis (Prepared for 2008 West Coast Stata Users' Group Meeting, San Francisco, November 13-14, 2008. Based on A. Colin Cameron and Pravin K. Trivedi, Microeconometrics using Stata, Stata Press.)
Initialize command `optimize` and optimize using `d2` evaluator.

```stata
: S = optimize_init()
: optimize_init_which(S,"min")
: optimize_init_evaluator(S, &pgmm())
: optimize_init_evaluatorevaluortype(S, "d2")
: optimize_init_argument(S, 1, y)
: optimize_init_argument(S, 2, X)
: optimize_init_argument(S, 3, Z)
: optimize_init_params(S, J(1,cols(X),0))
: optimize_init_which(S,"nr")
: b = optimize(S)
```

```
Iteration 0:  f(p) = 71995.212
Iteration 1:  f(p) = 9259.0408
Iteration 2:  f(p) = 1186.8103
Iteration 3:  f(p) = 3.4395408
Iteration 4:  f(p) = .00006905
Iteration 5:  f(p) = 5.672e-14
Iteration 6:  f(p) = 1.953e-27
```
Compute variance covariance matrix (manually) and list results.

: // Compute robust estimate of VCE and se's
: Xb = X*b'
: mu = exp(Xb)
: h = Z'(y-mu)
: W = cholinv(cross(Z,Z))
: G = -(mu:*Z)'X
: Shat = ((y-mu):*Z)'((y-mu):*Z)*rows(X)/(rows(X)-cols(X))
: Vb = luinv(G'W*G)*G'W*Shat*W*G*luinv(G'W*G)
: seb = (sqrt(diagonal(Vb)))'
: b \ seb

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.340291853</td>
<td>1.072907529</td>
<td>0.477817773</td>
<td>0.0027832801</td>
<td>-0.6832461817</td>
</tr>
<tr>
<td>2</td>
<td>1.559899278</td>
<td>0.0763116698</td>
<td>0.0690784466</td>
<td>0.0021932119</td>
<td>1.350370916</td>
</tr>
</tbody>
</table>

: end
1. Stata basics
2. Data management and graphics
3. Linear regression basics
4. Simulation
5. GLS regression
6. Linear instrumental variable regression
7. Quantile regression
8. Linear panel models
9. Nonlinear panel models
10. Nonlinear regression methods
11. Nonlinear optimization methods
12. Testing methods
13. Bootstrap methods
14. Binary outcome models
15. Multinomial models
16. Tobit and selection models
17. Count models
18. Nonlinear panel models
   A. Programming in Stata
   B. Mata