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

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.

November 14, 2008

1. Introduction

- Consider nonlinear cross-section regression of y_i on $\mathbf{x}_i$.
- Example is $y_i \; |\mathbf{x}_i \sim$ Poisson with mean $\mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$.
- This talk demonstrates various ways to code up the estimator,
  - using Stata command `ml`
  - and Mata command `optimize`

Outline

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

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------
docvis          int    %8.0g                  number of doctor visits
private         byte   %8.0g                  = 1 if private insurance
chronic         byte   %8.0g                  = 1 if a chronic condition
female          byte   %8.0g                  = 1 if female
income          float  %9.0g                  Income in $ / 1000


. summarize docvis private chronic female income

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      docvis |      4412    3.957389    7.947601          0        134
     private |      4412    .7853581    .4106202          0          1
     chronic |      4412    .3263826    .4689423          0          1
      female |      4412    .4718948    .4992661          0          1
      income |      4412    34.34018    29.03987    -49.999    280.777
```

Built-in command poisson

```
. 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                                Number of obs   =       4412
                                                  Wald chi2(4)    =     594.72
                                                  Prob > chi2     =     0.0000
Log pseudolikelihood = -18503.549                 Pseudo R2       =     0.1930
```

docvis	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
private	.7986652	.1090014	7.33	0.000	.5850263	1.012304
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: $\partial E[y|\mathbf{x}]/\partial x_j = \beta_j \times \exp(\mathbf{x}'\boldsymbol{\beta})$.

```
. mfx

Marginal effects after poisson
      y  = predicted number of events (predict)
         =  3.0296804
```

variable	dy/dx	Std. Err.	z	P>|z|	[95% C.I.	]	X
private*	1.978178	.20441	9.68	0.000	1.57755	2.37881	.785358
chronic*	4.200068	.27941	15.03	0.000	3.65243	4.7477	.326383
female*	1.528406	.17758	8.61	0.000	1.18036	1.87645	.471895
income	.0107766	.00331	3.25	0.001	.00428	.017274	34.3402

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

. 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	.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

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

- 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 `ML_y1`.
   It is referred to as `$ML_y1`
2. $\mathbf{x}$ is combined with $\boldsymbol{\beta}$ as the index $\mathbf{x}_i'\boldsymbol{\beta}$
   It is referred to as the program argument `theta1`
3. $\ln f(y|\mathbf{x},\boldsymbol{\beta})$ is referred to as the program argument `lf`

```
. program define lfpois
    version 10.0
    args lnf theta1                   // theta1=x'b, lnf=lnf(y)
    tempvar lnyfact mu
    local y "$ML_y1"                  // Define y so program more readable
    generate double `lnyfact' = lnfactorial(`y')
    generate double `mu' = exp(`theta1')
    quietly replace `lnf' = -`mu' + `y'*`theta1' - `lnyfact'
  end
```

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

```
. * 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
Log pseudolikelihood = -18503.549                 Prob > chi2     =     0.0000

------------------------------------------------------------------------------
             |               Robust
      docvis |      Coef.   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    .6072775
      income |    .003557   .0010825     3.29   0.001     .0014354    .0056787
       _cons |  -.2297263   .1108733    -2.07   0.038    -.4470339   -.0124188
------------------------------------------------------------------------------
```

- Command `ml` is not restricted to likelihood functions.
  e.g. For OLS maximize $-\sum_{i=1}^{N}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2$.
  `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 $\mathbf{x}_i'\boldsymbol{\beta}$ and α.
  `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

$$\begin{aligned} y_i &\sim \text{Poisson}[\exp(\alpha + \beta x_i)] \\ x_i &\sim \text{N}[0, 0.5^2] \\ \alpha &= 2;\ \beta = 1. \end{aligned}$$

- To check consistency
  - Set $N = 100,000$
  - Does $\widehat{\alpha} = 1$? Does $\widehat{\beta} = 1$?

- To check computation of the standard errors $s_{\widehat{\alpha}}$ and $s_{\widehat{\beta}}$.
    - Set $N = 500$.
    - Draw 2,000 samples of size N and obtain 2,000 estimates using command `simulate` or command `postfile`
    - Does $\sqrt{\frac{1}{1999}\sum_{s=1}^{2000}(\widehat{\beta}^{(s)} - \overline{\widehat{\beta}})^2} = \frac{1}{2000}\sum_{s=1}^{2000} s_{\widehat{\beta}}^{(s)}$?
    - i.e. Over the simulations
      does the st. deviation of $\widehat{\beta}$ = the average st. error of $\widehat{\beta}$?

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 β directly, rather than via index $\mathbf{x}'\beta$.
- 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 β to $\mathbf{x}'\beta$
- `mlsum` converts $\mathbf{x}_i'\beta$ to $\sum_{i=1}^{N} \mathbf{x}_i'\beta$.

```
. * Method d0: Program d0opois 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

initial:       log likelihood = -33899.609
alternative:   log likelihood = -28031.767
rescale:       log likelihood = -24020.669
Iteration 0:   log likelihood = -24020.669
Iteration 1:   log likelihood = -18845.464
Iteration 2:   log likelihood = -18510.257
Iteration 3:   log likelihood = -18503.552
Iteration 4:   log likelihood = -18503.549
Iteration 5:   log likelihood = -18503.549

                                                  Number of obs   =       4412
                                                  Wald chi2(4)    =    8052.34
Log likelihood = -18503.549                       Prob > chi2     =     0.0000
```

docvis	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

- 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
    version 10.0
    args todo b lnf g negH            // Add g and negH to the arguments list
    tempvar theta1                    // theta1 = x'b where x given in eq(1)
    mleval `theta1' = `b', eq(1)
    local y $ML_y1                    // Define y so program more readable
    mlsum `lnf' = -exp(`theta1') + `y'*`theta1' - lnfactorial(`y')
    if (`todo'==0 | `lnf'>=.) exit   // d1 extra code from here
    tempname d1
    mlvecsum `lnf' `d1' = `y' - exp(`theta1')
    matrix `g' = (`d1')
    if (`todo'==0 | `lnf'>=.) exit   // d2 extra code from here
    tempname d11
    mlmatsum `lnf' `d11' = exp(`theta1')
    matrix `negH' = `d11'
  end
```

6. Newton-Raphson algorithm using Mata

- Iterative algorithms are rules to compute $\widehat{\boldsymbol{\theta}}_{s+1}$ given $\widehat{\boldsymbol{\theta}}_s$.
- Gradient methods use a rule of the form

$$\widehat{\boldsymbol{\theta}}_{s+1} = \widehat{\boldsymbol{\theta}}_s + \mathbf{A}_s \mathbf{g}_s$$

  where $\mathbf{g}_s$ is the gradient of the objective function evaluated at $\widehat{\boldsymbol{\theta}}_s$.
- Newton-Raphson (NR) method approximates the objective function at $\widehat{\boldsymbol{\theta}}_s$ by a quadratic function.
  It chooses $\widehat{\boldsymbol{\theta}}_{s+1}$ to maximize this approximation.
  Then

$$\widehat{\boldsymbol{\theta}}_{s+1} = -\mathbf{H}_s^{-1} \mathbf{g}_s$$

  where $\mathbf{H}_s$ is the Hessian evaluated at $\widehat{\boldsymbol{\theta}}_s$.

- Poisson objective function, gradient and Hessian are:

$$\begin{aligned} Q(\boldsymbol{\beta}) &= \textstyle\sum_{i=1}^{N} \{-\exp(\mathbf{x}_i'\boldsymbol{\beta}) + y_i\mathbf{x}_i'\boldsymbol{\beta} - \ln y_i!\} \\ \mathbf{g}(\boldsymbol{\beta}) &= \textstyle\sum_{i=1}^{N} (y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta}))\mathbf{x}_i \\ \mathbf{H}(\boldsymbol{\beta}) &= \textstyle\sum_{i=1}^{N} -\exp(\mathbf{x}_i'\boldsymbol{\beta})\mathbf{x}_i\mathbf{x}_i'. \end{aligned}$$

- So NR is

$$\begin{aligned} \widehat{\boldsymbol{\beta}}_{s+1} &= \widehat{\boldsymbol{\beta}}_s - \mathbf{H}(\widehat{\boldsymbol{\beta}}_s)^{-1} \times \mathbf{g}(\widehat{\boldsymbol{\beta}}_s) \\ &= \widehat{\boldsymbol{\beta}}_s + \left[\sum_{i=1}^{N} \exp(\mathbf{x}_i'\widehat{\boldsymbol{\beta}}_s)\mathbf{x}_i\mathbf{x}_i'\right]^{-1} \times \sum_{i=1}^{N} (y_i - \exp(\mathbf{x}_i'\widehat{\boldsymbol{\beta}}_s))\mathbf{x}_i. \end{aligned}$$

- Core Mata code is

```
> 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 $\mathbf{x}$

```
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 $\widehat{\beta}$.

```
. * Complete Mata code for Poisson MLE NR iterations
. mata
------------------------------------------------- mata (type end to exit) ----------
:   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 values

:   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 $\widehat{\beta}$.

```
:   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                              // num iterations
  13

:   cha                               // stopping criterion
  1.11465e-24

:   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.

```
. * 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
  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 $\mathbf{v}$, where $\mathbf{e}'\mathbf{v} = f(\mathbf{p})$.
  Suited to m-estimators (MLE, LS, just-identified NLIV).
- Type `d` evaluator provides formula for scalar v where $v = f(\mathbf{p})$.
  Suited to over-identified generalized method of moments (GMM).

Declare the function poissonmle and st_view data

```
. mata
------------------------------------------ mata (type end to exit) ------
:   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.

```
:   S = optimize_init()

:   optimize_init_evaluator(S, &poissonmle())

:   optimize_init_evaluatortype(S, "v2")

:   optimize_init_argument(S, 1, y)

:   optimize_init_argument(S, 2, X)

:   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 = optimize_result_V_robust(S)

:   serob = (sqrt(diagonal(Vbrob)))'

:   b \ serob
                 1              2              3              4              5
    +--------------------------------------------------------------------------+
  1 |   .7986653788    1.091865108    .4925480693    .0035570127   -.2297263376 |
  2 |   .1090014507    .0559951312    .0585364746    .0010824894    .1108732568 |
    +--------------------------------------------------------------------------+

: end
-----------------------------------------------------------------------------------
```

Note: Can `st_matrix` back to Stata and `ereturn display` results.

8. NL2SLS example

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

- GMM estimator minimizes

$$Q(\boldsymbol{\beta}) = \mathbf{h}(\boldsymbol{\beta})'\mathbf{W}\mathbf{h}(\boldsymbol{\beta}).$$

- The $K \times 1$ gradient vector is

$$\mathbf{g}(\boldsymbol{\beta}) = \partial Q(\boldsymbol{\beta})/\partial\boldsymbol{\beta} = \mathbf{G}(\boldsymbol{\beta})'\mathbf{W}\mathbf{h}(\boldsymbol{\beta}).$$

- The $K \times K$ expected Hessian is

$$\mathbf{H}(\boldsymbol{\beta}) = \partial^2 Q(\boldsymbol{\beta})/\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}' = \mathbf{G}(\boldsymbol{\beta})'\mathbf{W}\mathbf{G}(\boldsymbol{\beta})'.$$

- Where

$$\begin{aligned} \mathbf{G}(\boldsymbol{\beta}) &= -\textstyle\sum_i \exp(\mathbf{x}_i'\boldsymbol{\beta})\mathbf{z}_i\mathbf{x}_i' \\ \mathbf{h}(\boldsymbol{\beta}) &= \textstyle\sum_i \mathbf{z}_i(y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})) \\ \mathbf{W} &= (\mathbf{Z}'\mathbf{Z})^{-1} = \left(\textstyle\sum_i \mathbf{z}_i\mathbf{z}_i'\right)^{-1} \end{aligned}$$

Declare the function pgmm and st_view data

```
. mata
------------------------------------------------- mata (type end to exit) ------
:   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'"))
```

Initialize command optimize and optimize using d2 evaluator.

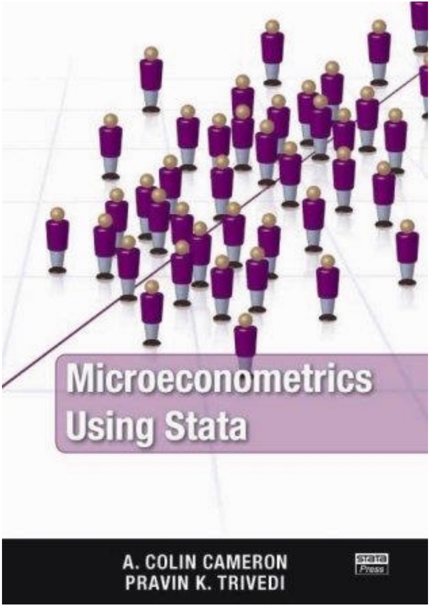
```
:   S = optimize_init()

:   optimize_init_which(S,"min")

:   optimize_init_evaluator(S, &pgmm())

:   optimize_init_evaluatortype(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_technique(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
```

	1	2	3	4	5
1	1.340291853	1.072907529	.477817773	.0027832801	-.6832461817
2	1.559899278	.0763116698	.0690784466	.0021932119	1.350370916

```
: end
```


Microeconometrics
Using Stata
A. COLIN CAMERON
PRAVIN K. TRIVEDI
Stata Press

Book Outline

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