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* :

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

- 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, s^2)
- pdf: $$f(y_i \mid m_i, s^2) = \frac{1}{\sqrt{2ps^2}} e^{-\frac{1}{2}\frac{(y_i - m_i)^2}{s^2}}$$
- Reparameterize $\mu_i = x_i ß$
- Likelihood for case *i* : $$L_i = \frac{1}{\sqrt{2ps^2}} e^{-\frac{1}{2}\frac{(y_i - x_i b)^2}{s^2}}$$
- Log-likelihood for case *i* (**this is what Stata wants**):

$$\ln L_i = -\frac{1}{2}\ln(2p) - \frac{1}{2}\ln(s^2) - \frac{1}{2}\left[\frac{(y_i - x_i b)^2}{s^2}\right]$$

Programming Likelihood Functions: ML Normal Regression

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

$$L = \prod_{i=1}^{N} \frac{1}{\sqrt{2ps^{2}}} e^{-\frac{1}{2}\frac{(y_i - x_i b)^2}{s^{2}}}$$

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

$$\ln L = -\frac{N}{2}\ln(2p) - \frac{N}{2}\ln(s^{2}) - \frac{1}{2}\sum_{i=1}^{N}\left[\frac{(y_i - x_i b)^2}{s^{2}}\right]$$

- 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 progname
   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: $\text{theta1}_i = x_{1i}\,b$
- `theta2' is associated with the second parameter, containing evaluation of the 2nd equation: $\text{theta2}_i = x_{2i}\,b$

Programming Likelihood Functions: Syntax

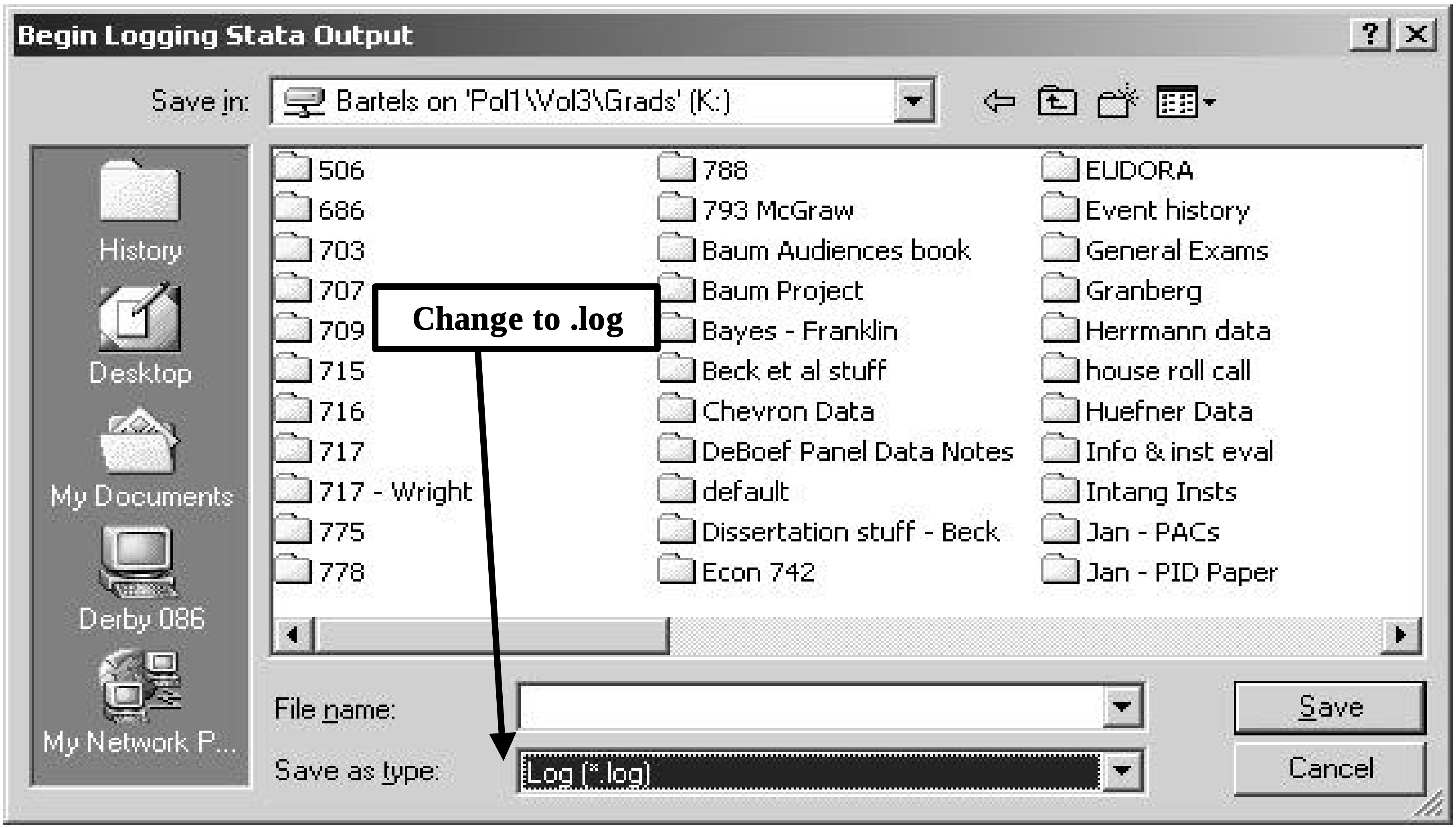
- Global macros:

  $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

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

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

normreg.do - Stata Do-file Editor

File Edit Search Tools

Run the program

```
program define normreg

	args lnf theta1 theta2
	/* This is a two parameter model: mean and variance. We reparameterize
	mu_i = x_i*b and Stata will understand this when we call up the model  */

	quietly replace `lnf'=-.5*ln(2*_pi)-.5*ln(`theta2')-.5*(($ML_y1-`theta1')^2)/`theta2'
	/* This is simply the log-likelihood for case i */

end
```

Recall:

$$\ln L_i = -\frac{1}{2}\ln(2p) - \frac{1}{2}\ln(s^2) - \frac{1}{2}\left[\frac{(y_i - x_i b)^2}{s^2}\right]$$

Line number: 5

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*

Programming Likelihood Functions: Maximizing the Likelihood Function

- Other useful commands to run after *ml model*:

  ml check verifies that the program you wrote works

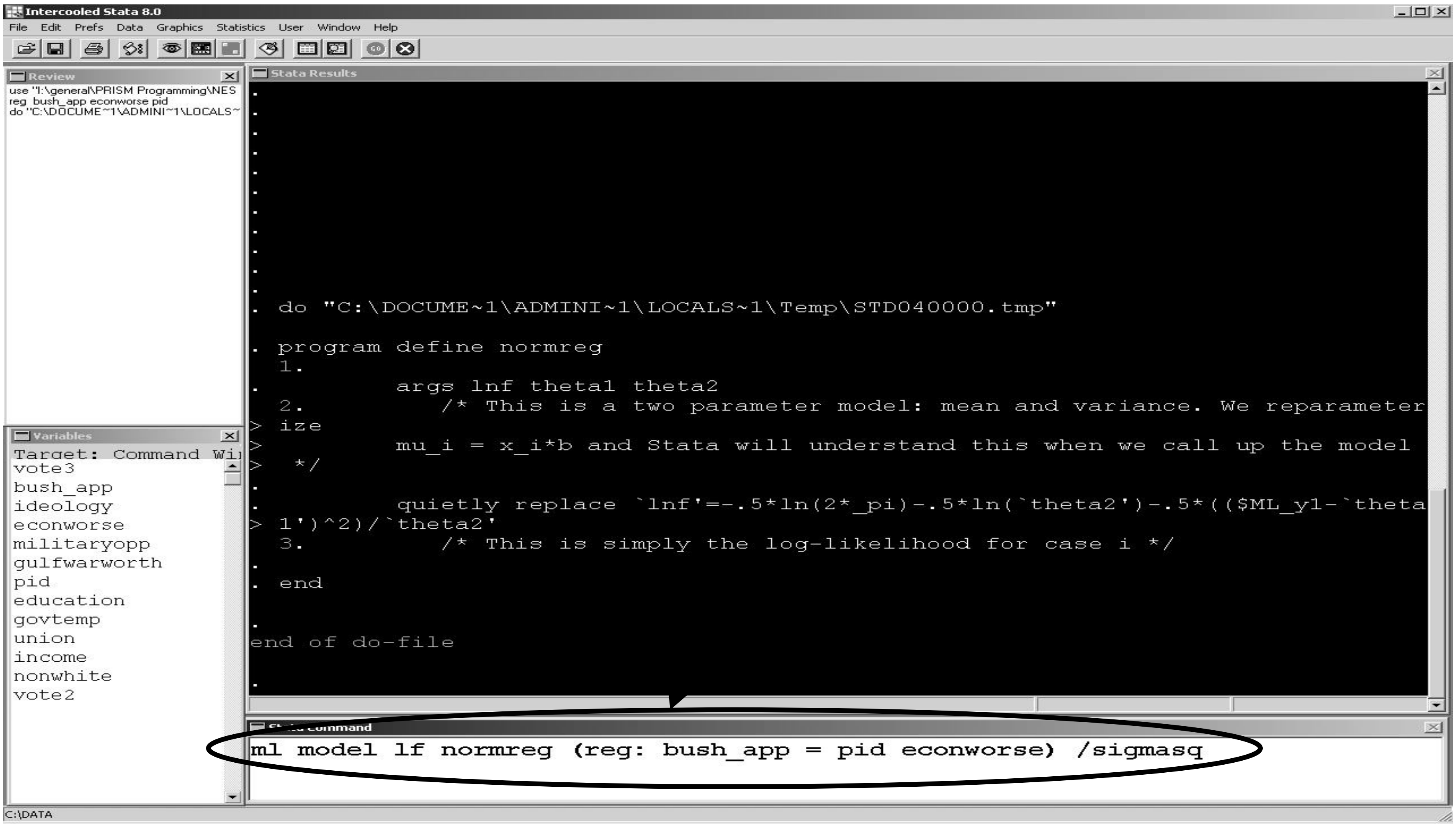
  ml search searches for better starting values

  lf0(#k LL0) 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.

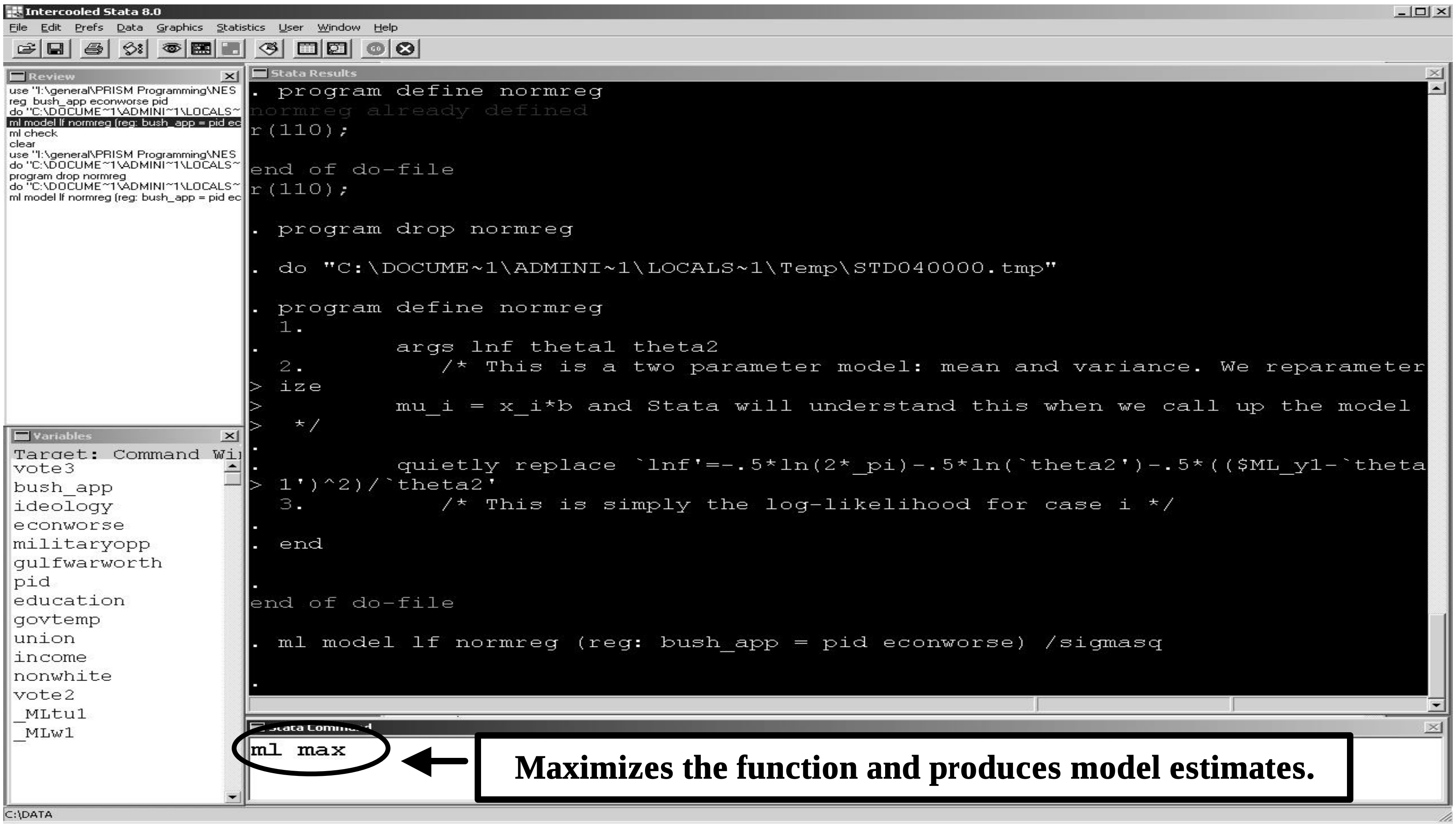
Programming Likelihood Functions: Estimating the ML Normal Model

- Let’s estimate a simple model; we’ll regress George H.W. Bush’s approval on PID and economic perceptions.

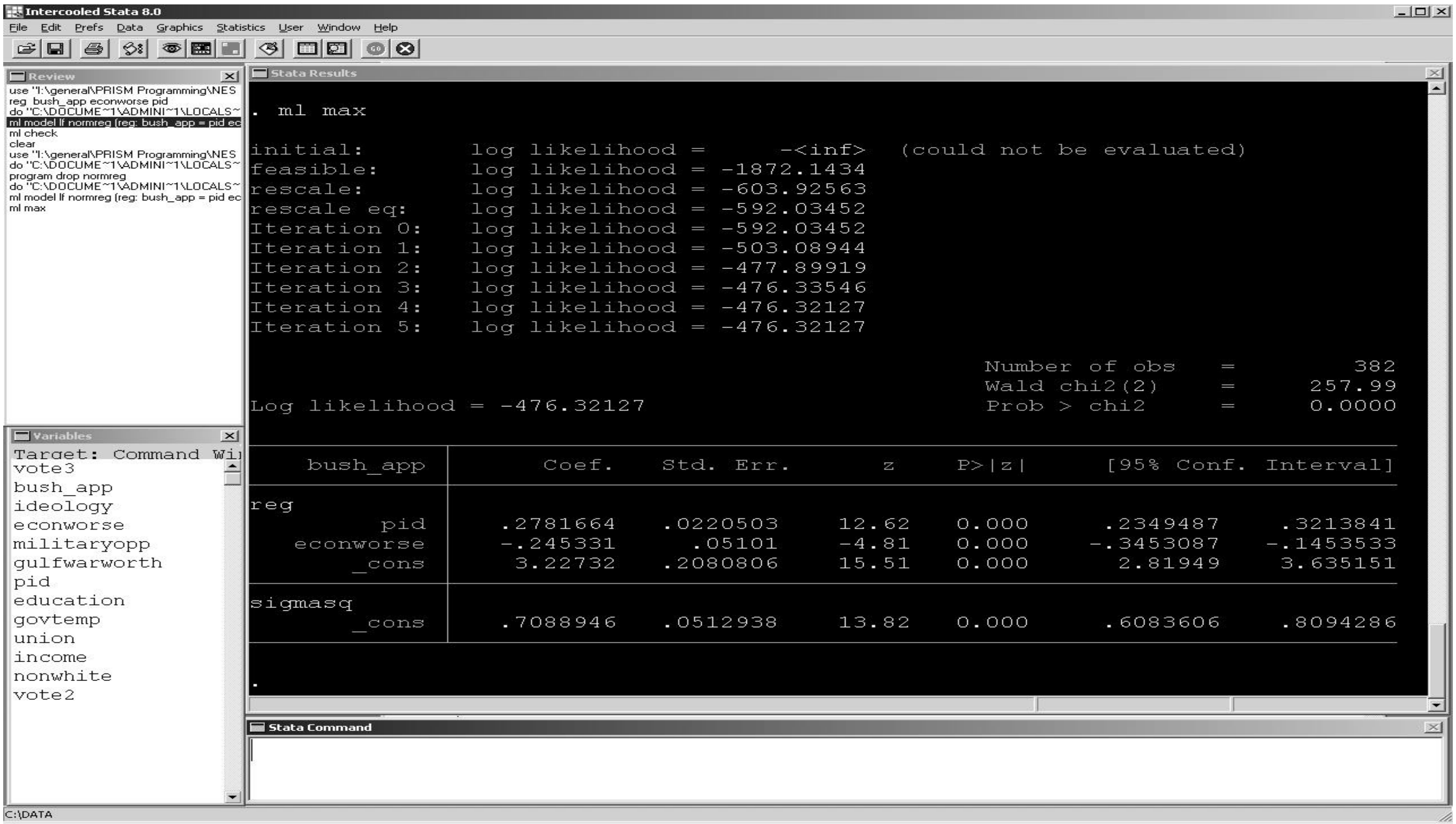
Programming Likelihood Functions: Estimating the ML Normal Model



Programming Likelihood Functions: Estimating the ML Normal Model



Programming Likelihood Functions: Estimating the ML Normal Model



Programming Likelihood Functions: Comparing ML Reg to OLS

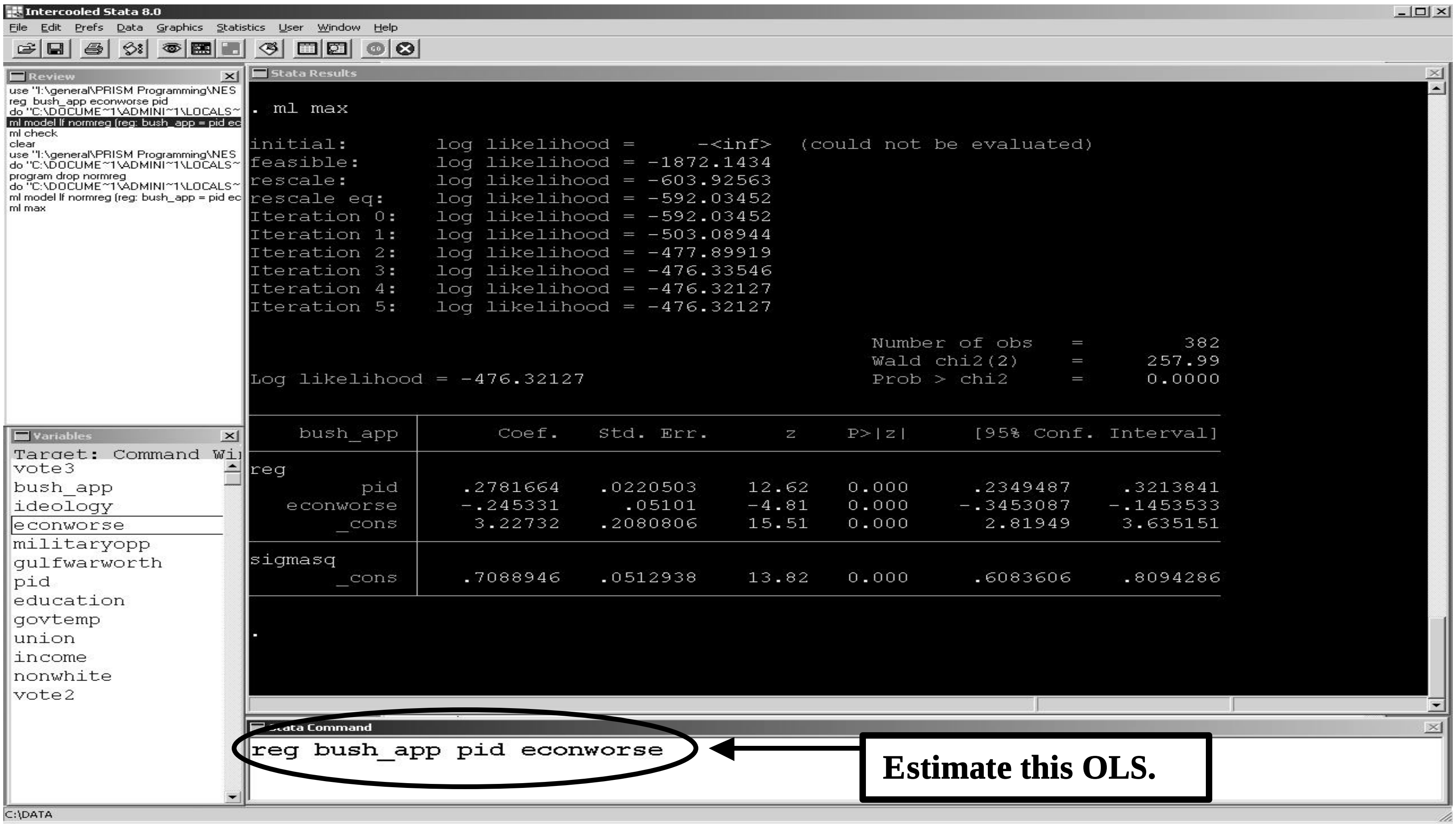
- OLS and ML Normal Regression produce identical parameter estimates. It can be shown that the analytical solution for ML Normal Regression is:

  $$ß = (X'X)^{-1}X'Y$$

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

- Standard errors will be different, though, because:
  - In ML: $s^2 = \frac{\sum_{i=1}^{N} e_i^2}{N}$
  - In OLS: $s^2 = \frac{\sum_{i=1}^{N} e_i^2}{N-k}$

Programming Likelihood Functions: Comparing ML Reg to OLS



Programming Likelihood Functions: Comparing ML Reg to OLS

Intercooled Stata 8.0

File Edit Prefs Data Graphics Statistics User Window Help

Review

```
use "I:\general\PRISM Programming\NES
reg  bush_app econworse pid
do "C:\DOCUME~1\ADMINI~1\LOCALS~
ml model lf normreg (reg: bush_app = pid ec
ml check
clear
use "I:\general\PRISM Programming\NES
do "C:\DOCUME~1\ADMINI~1\LOCALS~
program drop normreg
do "C:\DOCUME~1\ADMINI~1\LOCALS~
ml model lf normreg (reg: bush_app = pid ec
ml max
reg bush_app pid econworse
```

Variables

```
Target: Command Wi
vote3
bush_app
ideology
econworse
militaryopp
gulfwarworth
pid
education
govtemp
union
income
nonwhite
vote2
```

Stata Results

```
    bush_app |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
reg          |
         pid |   .2781664   .0220503    12.62   0.000     .2349487    .3213841
   econworse |   -.245331     .05101    -4.81   0.000    -.3453087   -.1453533
       _cons |    3.22732   .2080806    15.51   0.000      2.81949    3.635151
-------------+----------------------------------------------------------------
sigmasq      |
       _cons |   .7088946   .0512938    13.82   0.000     .6083606    .8094286
------------------------------------------------------------------------------

.

. reg bush_app pid econworse

      Source |       SS       df       MS              Number of obs =     382
-------------+------------------------------           F(  2,   379) =  127.98
       Model |  182.888133     2  91.4440663           Prob > F      =  0.0000
    Residual |  270.797731   379  .714505887           R-squared     =  0.4031
-------------+------------------------------           Adj R-squared =  0.4000
       Total |  453.685864   381  1.19077655           Root MSE      =  .84528

------------------------------------------------------------------------------
    bush_app |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         pid |   .2781664   .0221374    12.57   0.000      .234639    .3216938
   econworse |   -.245331   .0512114    -4.79   0.000    -.3460251   -.1446368
       _cons |    3.22732   .2089025    15.45   0.000     2.816567    3.638073
------------------------------------------------------------------------------

.
```

Stata Command

Coefficients are identical, SEs are different.

C:\DATA

Programming Likelihood Functions: Logit and Probit

- In binary response models, we want to model the probability of "success" for case *i*, i.e., Pr(y_i=1) = ? $_i$
- We parameterize ? $_i$ as a cumulative distribution function (cdf) of a particular distribution, i.e., $F\ (x_i ß\ )$
  - For logit, we use the logistic cdf:

$$\Pr(y_i = 1) = F(x_i \boldsymbol{b}) = \frac{\exp(x_i \boldsymbol{b})}{1 + \exp(x_i \boldsymbol{b})}$$

  - For probit, we use the normal cdf:

$$\Pr(y_i = 1) = F(x_i \boldsymbol{b}) = \Phi(x_i \boldsymbol{b})$$

Programming Likelihood Functions: Logit and Probit

- The likelihood for case *i* is:

$$L_i = [F(x_i b)]^{y_i}[1 - F(x_i b)]^{1-y_i}$$

- The log-likelihood for case *i* is:

$$\ln L_i = y_i \ln[F(x_i b)] + (1 - y_i)\ln[1 - F(x_i b)]$$

Again, this is what we're going to give Stata

- For logit, we'll replace $F(x_i ß)$ 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

mylogit.do - Stata Do-file Editor

File Edit Search Tools

```
program define mylogit

        args lnf theta1
        /*  only one parameter here, pi, which we reparameterize as x_i*b  */

        tempvar p
        /*  creating this temp var makes organization a little easier. Here, "p" is going to
        stand for F(xb), i.e., the logistic cdf  */

        quietly gen double `p'=exp(`theta1')/(1+exp(`theta1'))
        /*  we define the F(xb) as the logistic cdf  */

        quietly replace `lnf'=$ML_y1*ln(`p')+(1-$ML_y1)*ln(1-`p')
        /*  now, we just put `p' in where the cdf is defined in the lnL  */

end
```

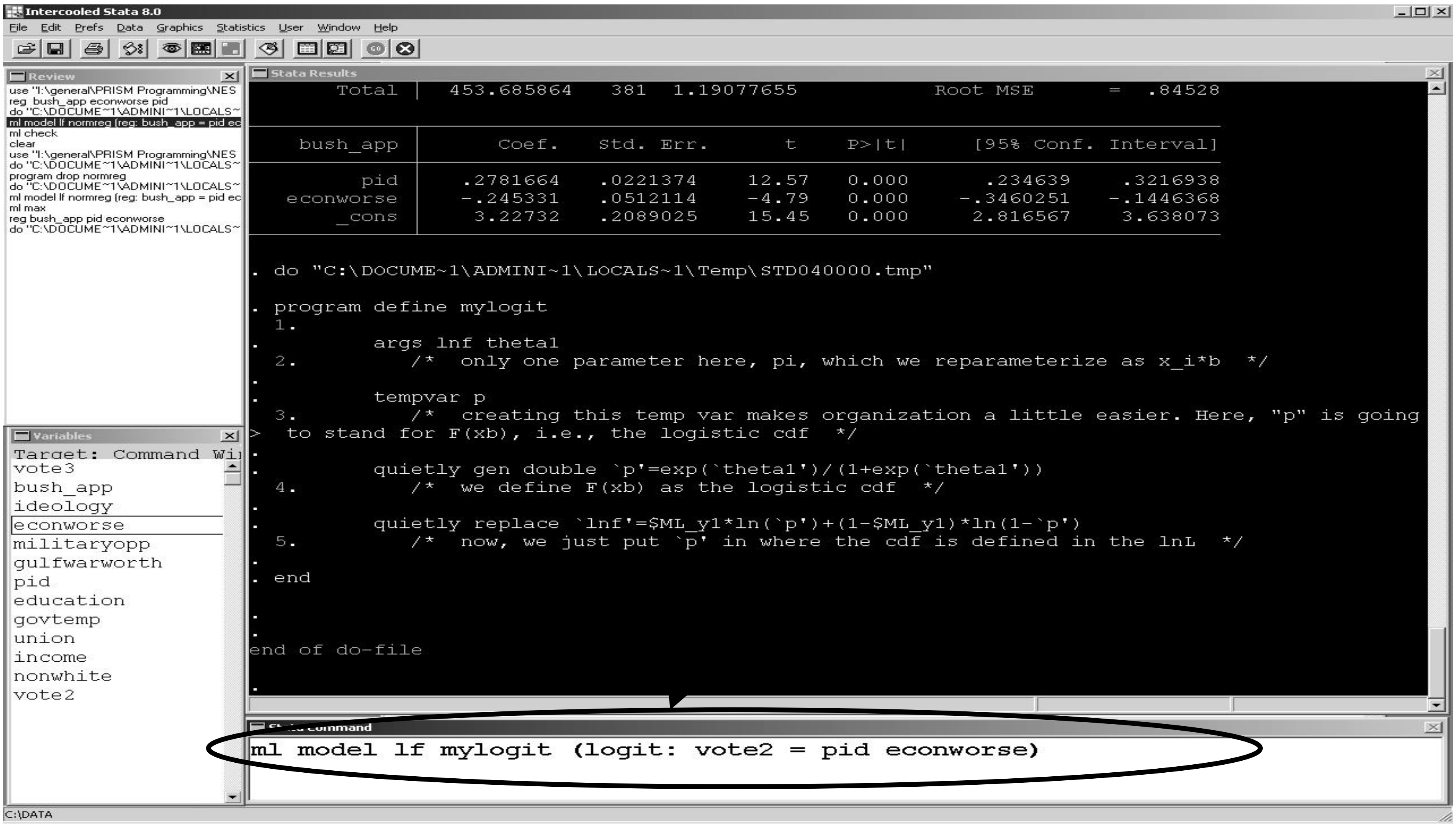
Run the program

Line number: 8

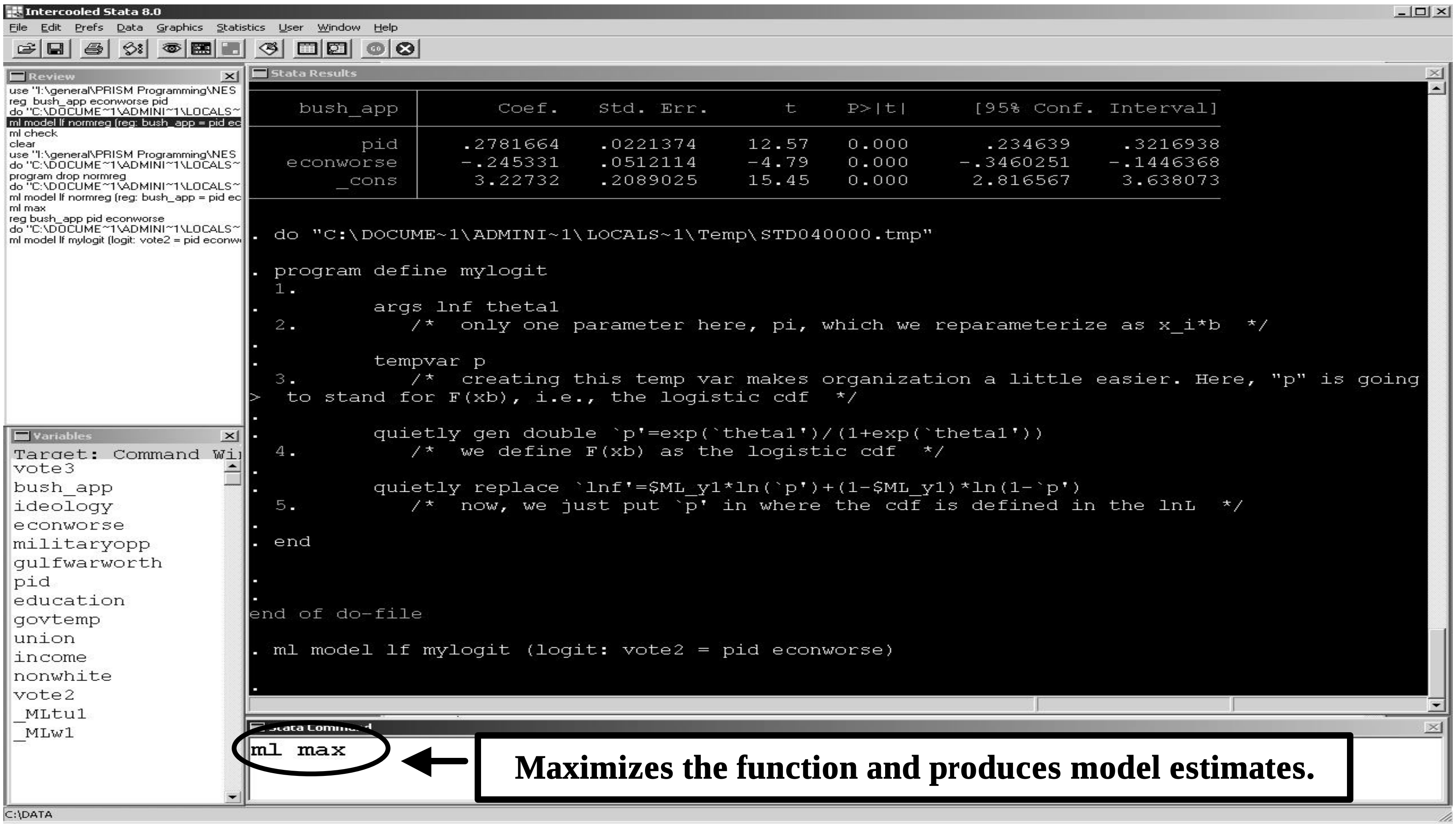
Recall:

$$\ln L_i = y_i \ln[F(x_i b)] + (1 - y_i)\ln[1 - F(x_i b)]$$

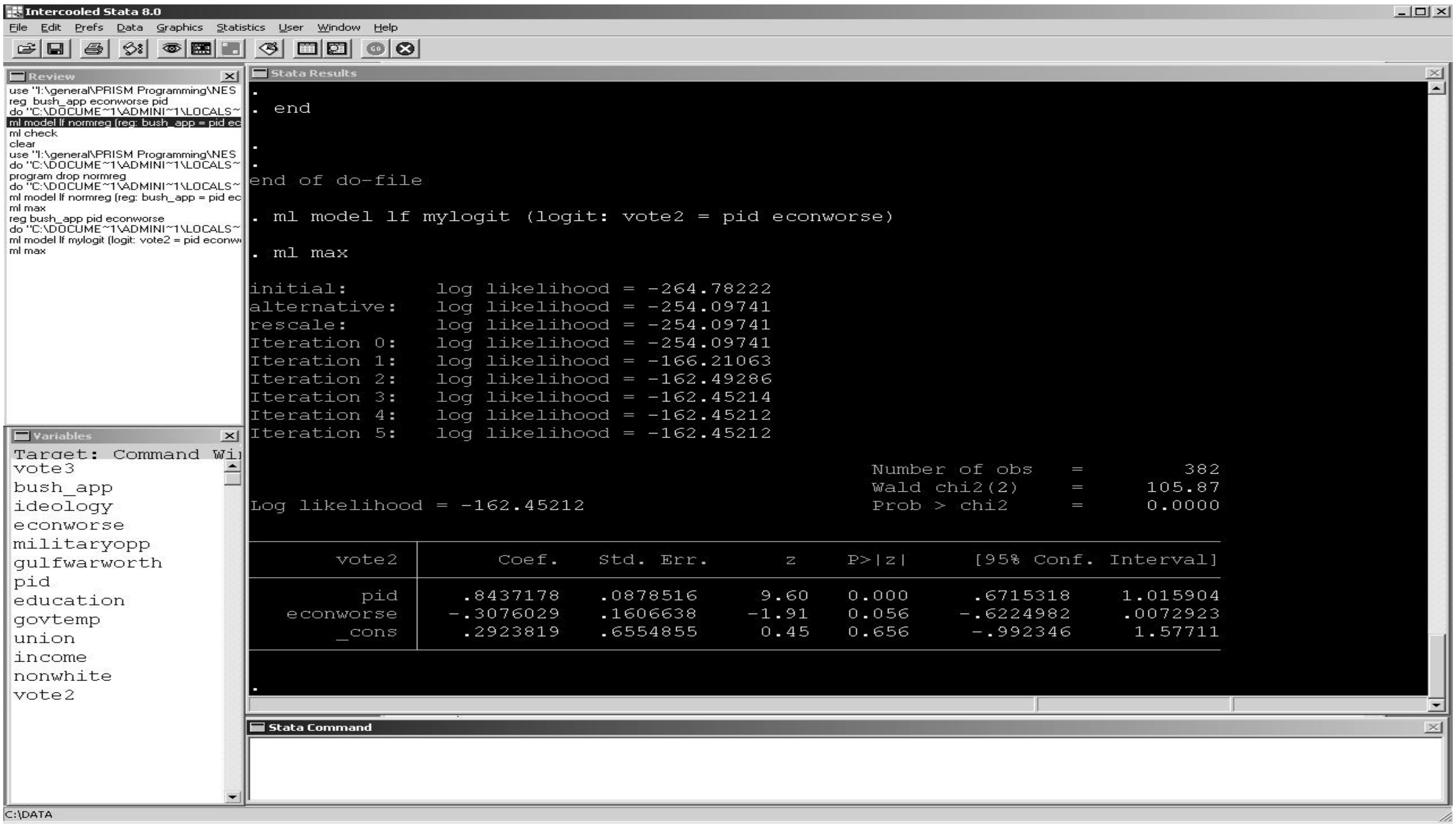
Programming Likelihood Functions: Logit and Probit



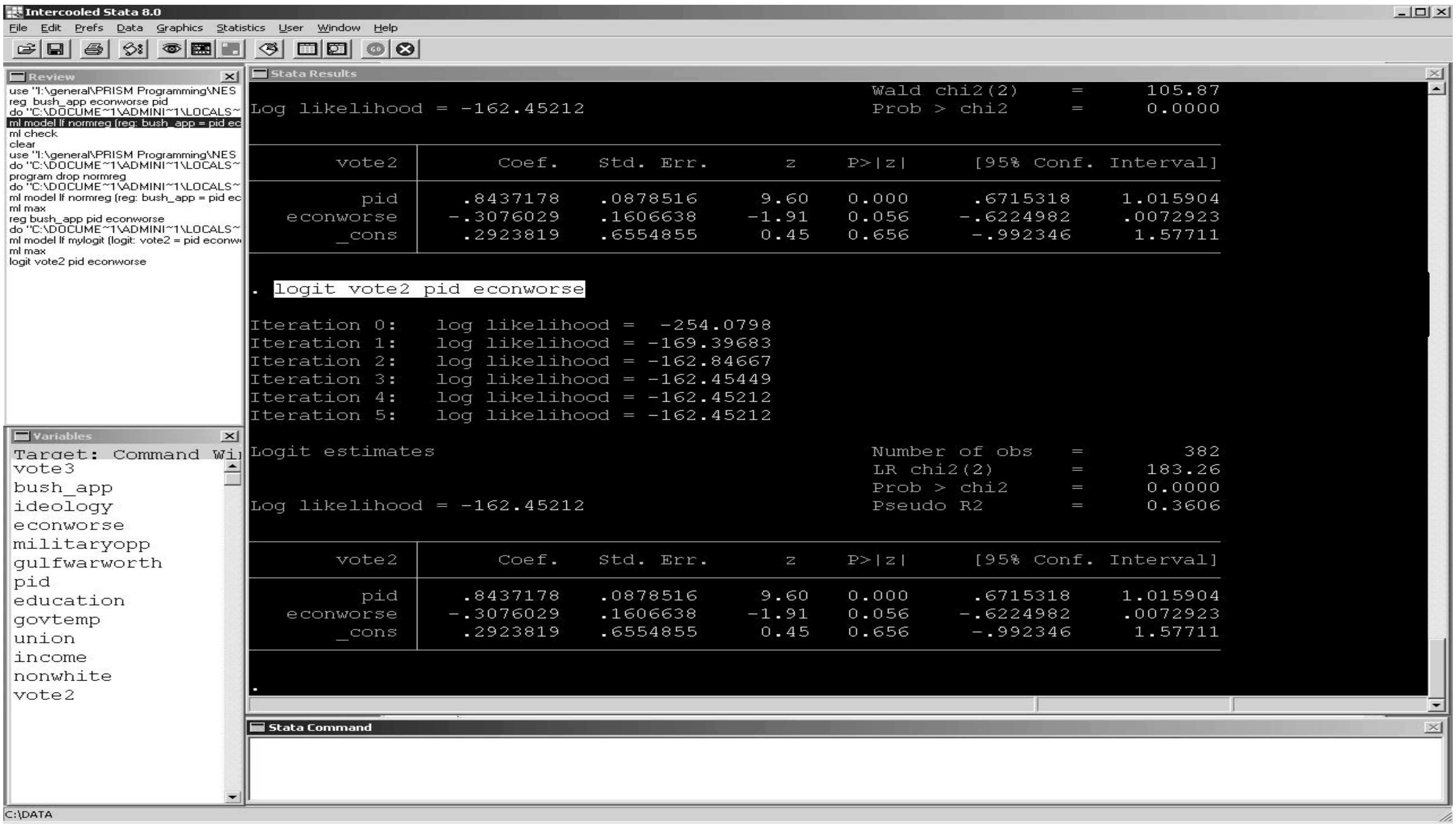
Programming Likelihood Functions: Logit and Probit



Programming Likelihood Functions: Logit and Probit



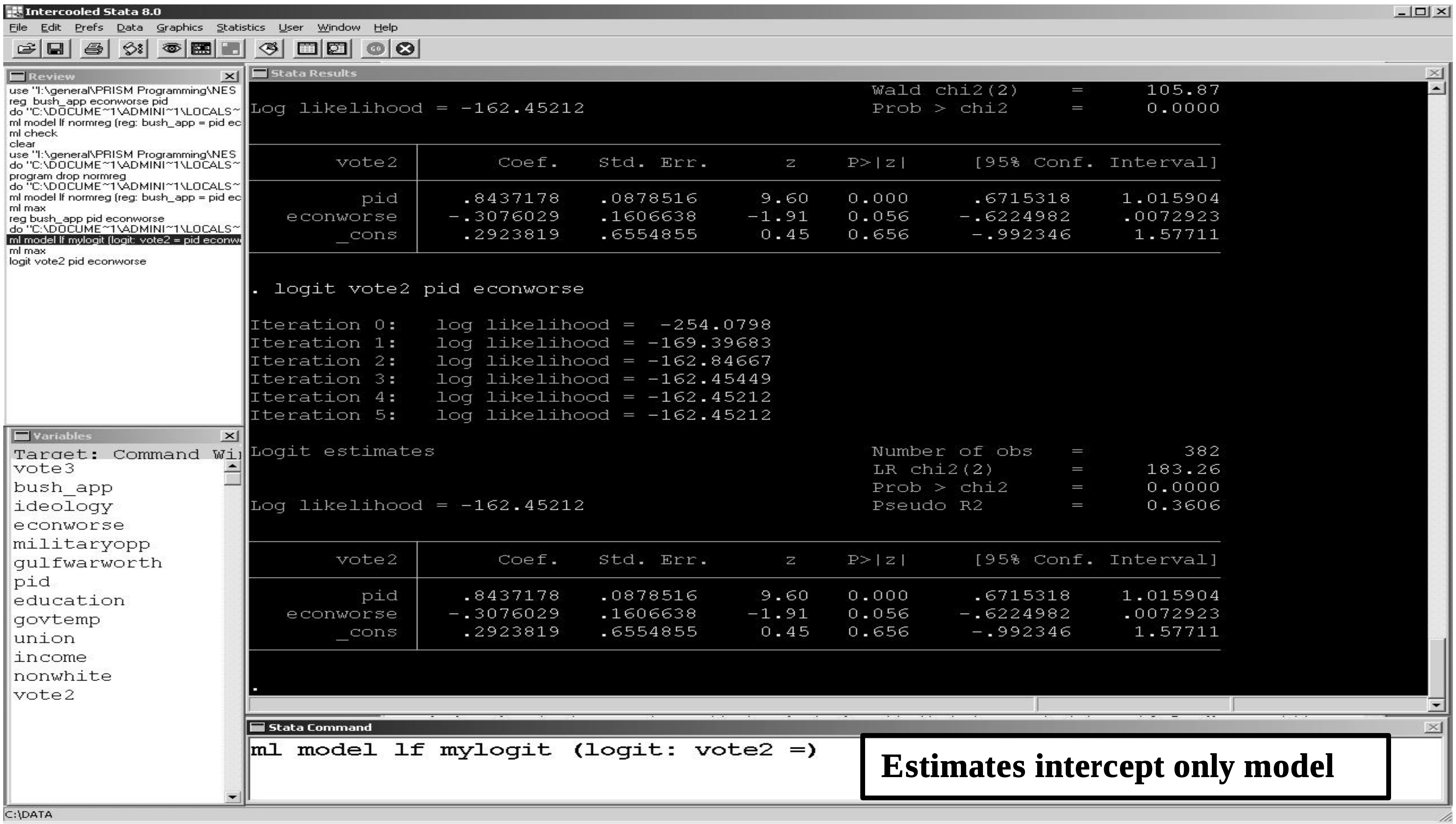
Programming Likelihood Functions: Logit and Probit



Programming Likelihood Functions: The Likelihood Ratio Test

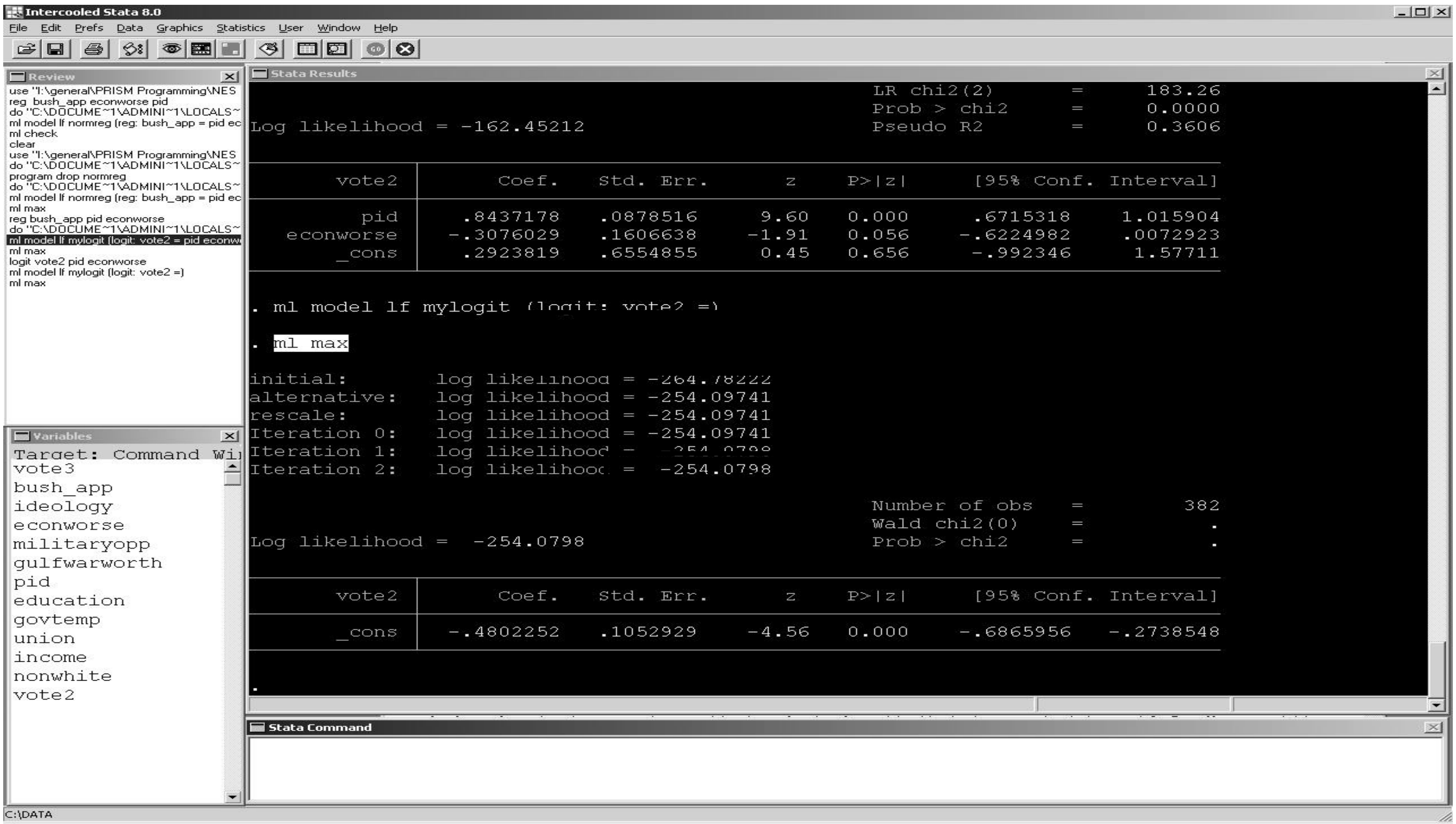
- By default, “ml model” produces a Wald test for overall goodness of fit test (which tests that the coefficients are jointly equal to zero).
- To get an LR test, we need to:
  - Estimate an intercept only model to get LL0, the initial LL.
  - We need to specify *k* for the intercept-only model, which in this case is 1.
  - After the “ml model” command, we enter *lf0(k LL0)*.

Programming Likelihood Functions: The Likelihood Ratio Test

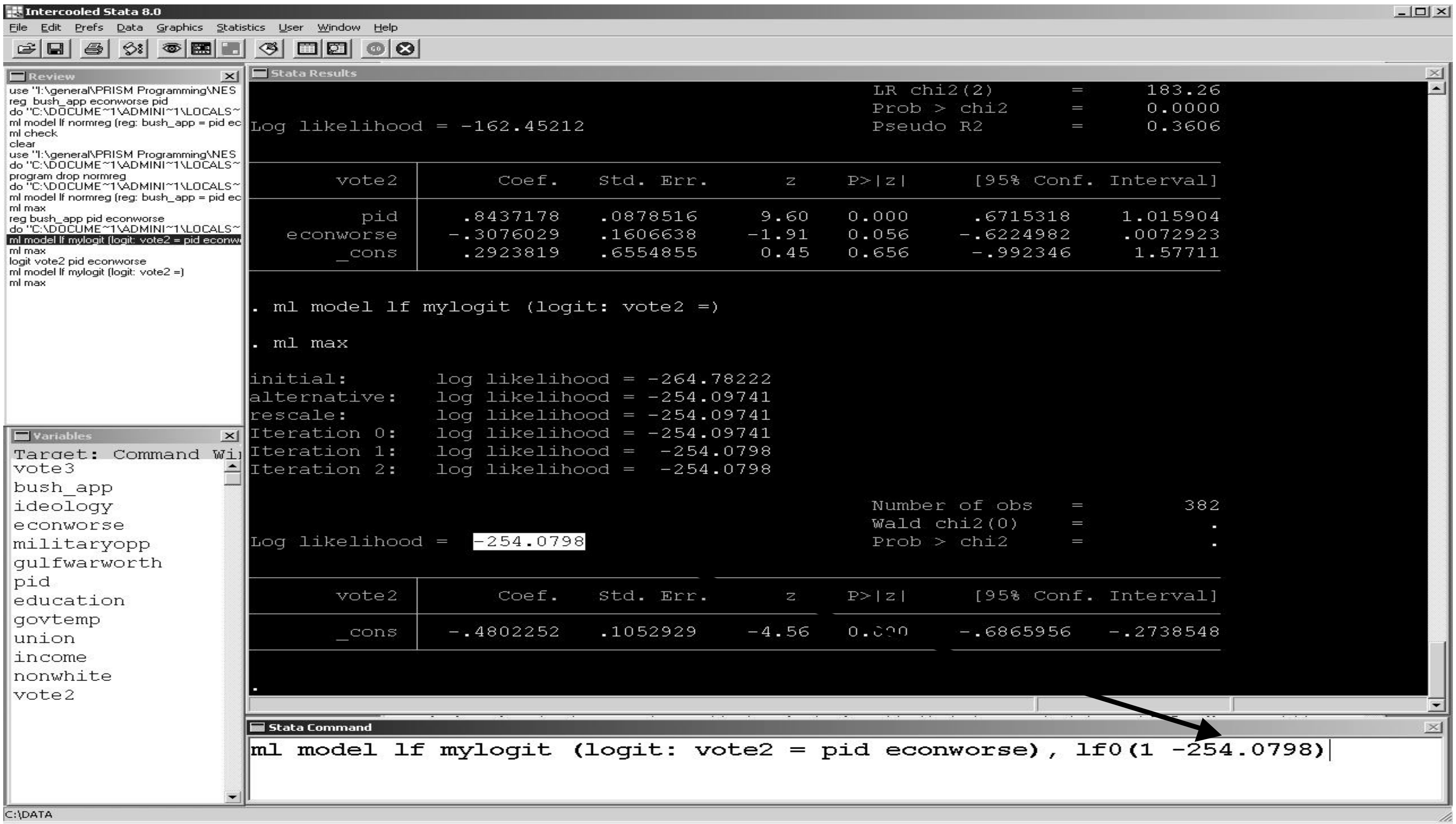


Estimates intercept only model

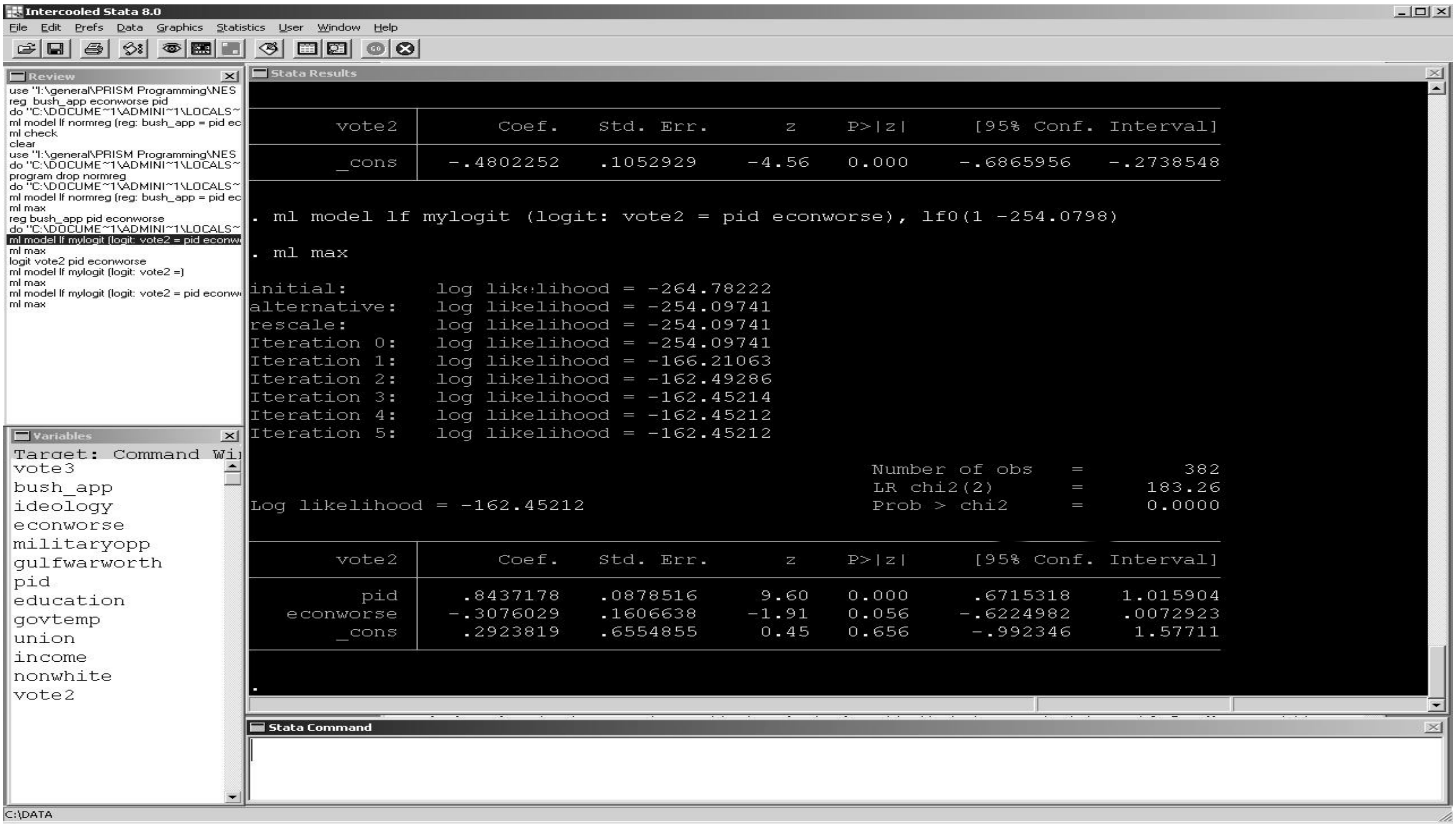
Programming Likelihood Functions: The Likelihood Ratio Test



Programming Likelihood Functions: The Likelihood Ratio Test



Programming Likelihood Functions: The Likelihood Ratio Test



Programming Likelihood Functions: Probit Program

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 s_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 s_i^2 as:

$$s_i^2 = e^{z_i g}$$

- Where the z_i's exogenous variables that influence the variability around the expected value, and gamma is a vector of parameters.
- Log-likelihood:

$$\ln L_i = -\frac{1}{2}\ln(2p) - \frac{1}{2} z_i g - \frac{1}{2}\left[\frac{(y_i - x_i b)^2}{e^{z_i g}}\right]$$

Programming Likelihood Functions: Heteroskedastic Regression

hetreg.do - Stata Do-file Editor

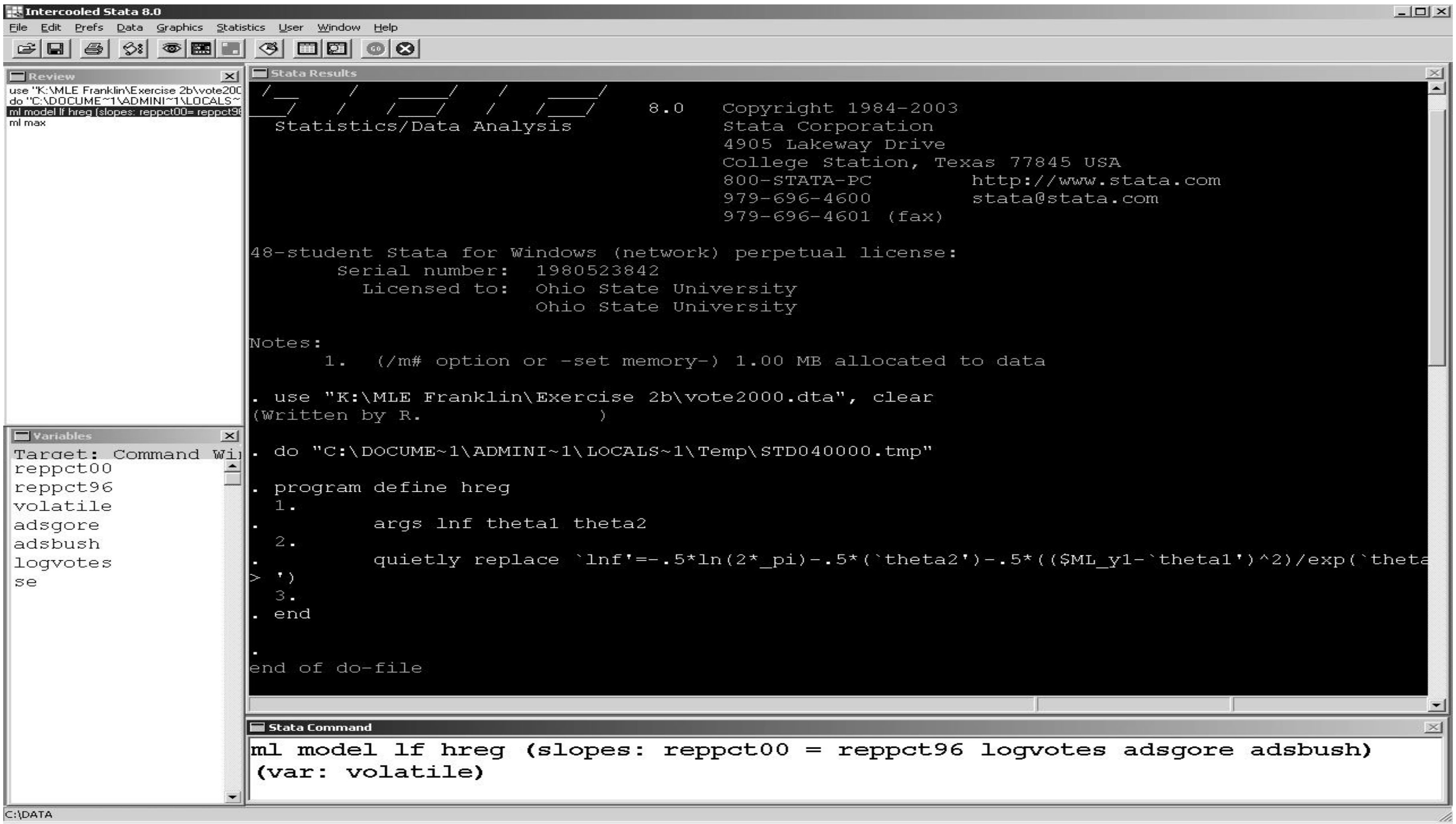
File Edit Search Tools

```
program define hreg
        args lnf theta1 theta2
        quietly replace `lnf'=-.5*ln(2*_pi)-.5*(`theta2')-.5*((\$ML_y1-`theta1')^2)/exp(`theta2')
end
```

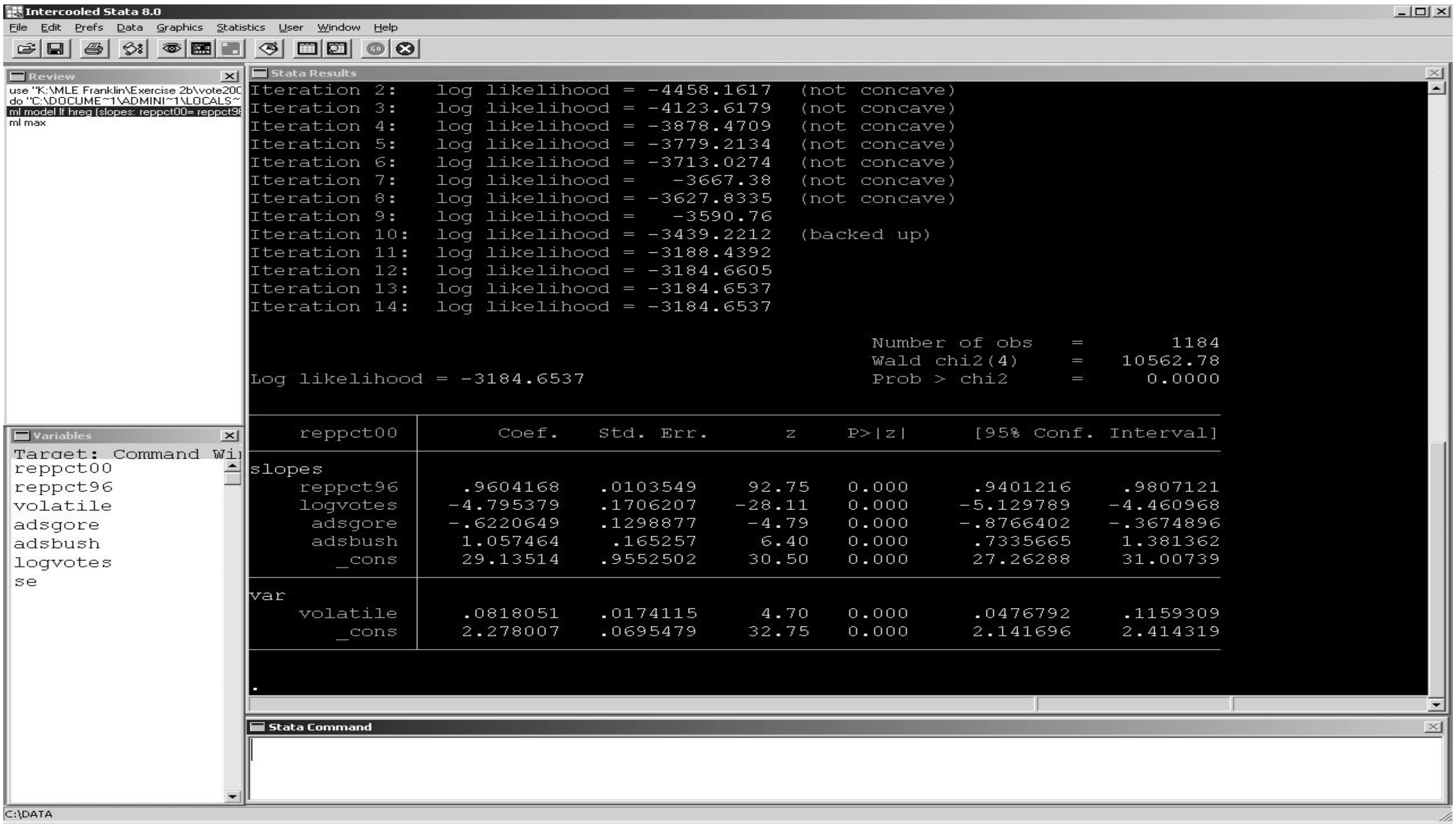
$$\ln L_i = -\frac{1}{2}\ln(2\pi) - \frac{1}{2} z_i\gamma - \frac{1}{2}\left[\frac{(y_i - x_i\beta)^2}{e^{z_i\gamma}}\right]$$

Line number: 8

Programming Likelihood Functions: Heteroskedastic Regression



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

splitpop.do - Stata Do-file Editor

File Edit Search Tools

```
program define splitpop

	args lnf theta1 theta2 theta3
	/*  theta1: XB for duration equation; theta2: XB for logit equation;
	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+(`l'*$ML_y1)^(1/`theta3')))
	/*  survival function  */

	quietly gen double `p'=ln(`d')-ln(`theta3')+((1/`theta3')-1)*ln($ML_y1)+(1/`theta3')* /*
	*/ ln(`l')-2*ln(1+(`l'*$ML_y1)^(1/`theta3'))
	/*  pdf  */

	quietly replace `lnf'=$ML_y2*(`p')+(1-$ML_y2)*ln(`s')

end
```

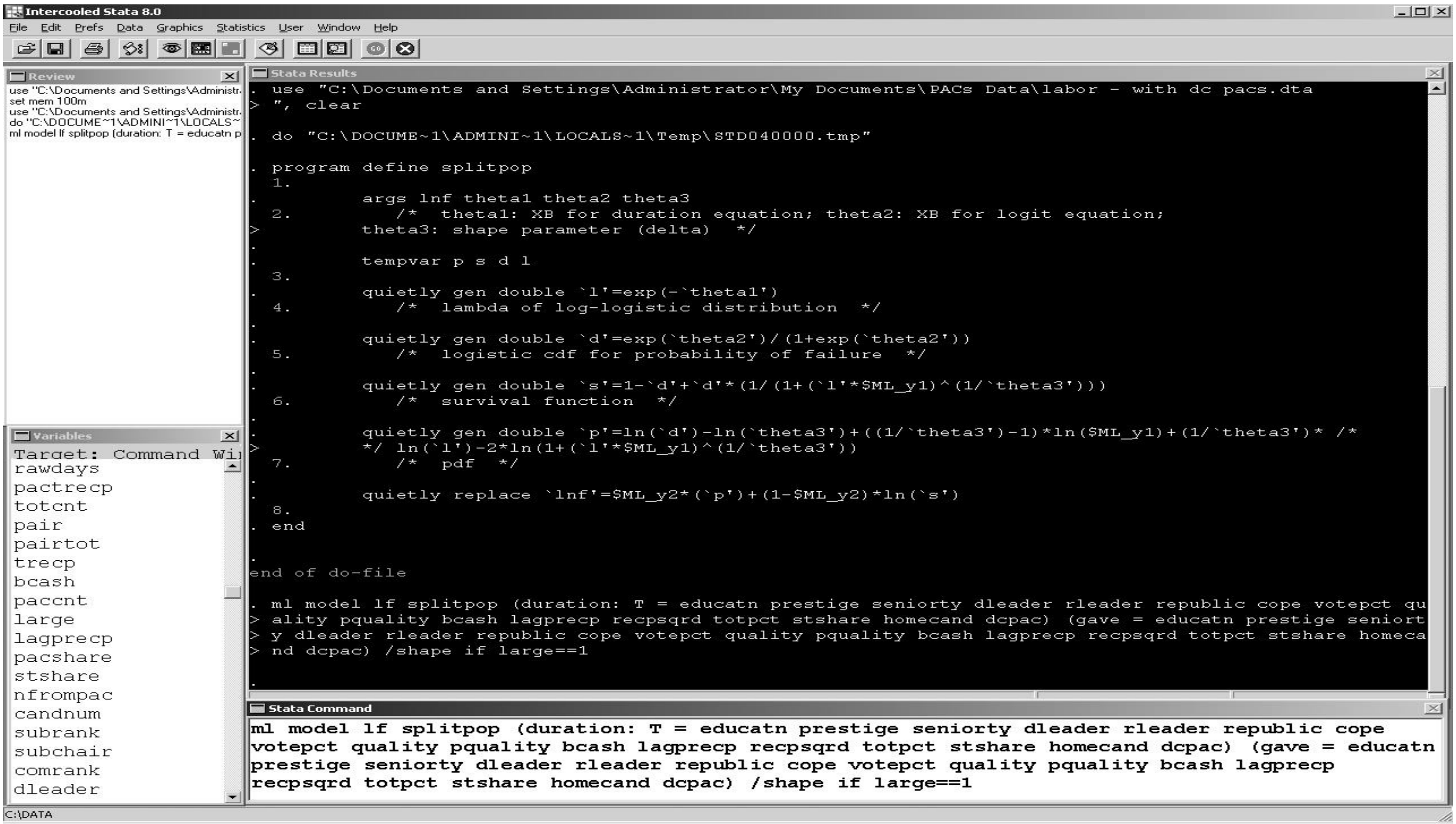
$$\ln L_i = \underbrace{R_i}_{\textbf{censoring indicator}}\underbrace{[\ln d_i + \ln g(t_i,q)]}_{\textbf{pdf}} + (1-R_i)\underbrace{\ln[1-d_i + d_i G(t_i,q)]}_{\textbf{survivor function}}$$

Line number: 25

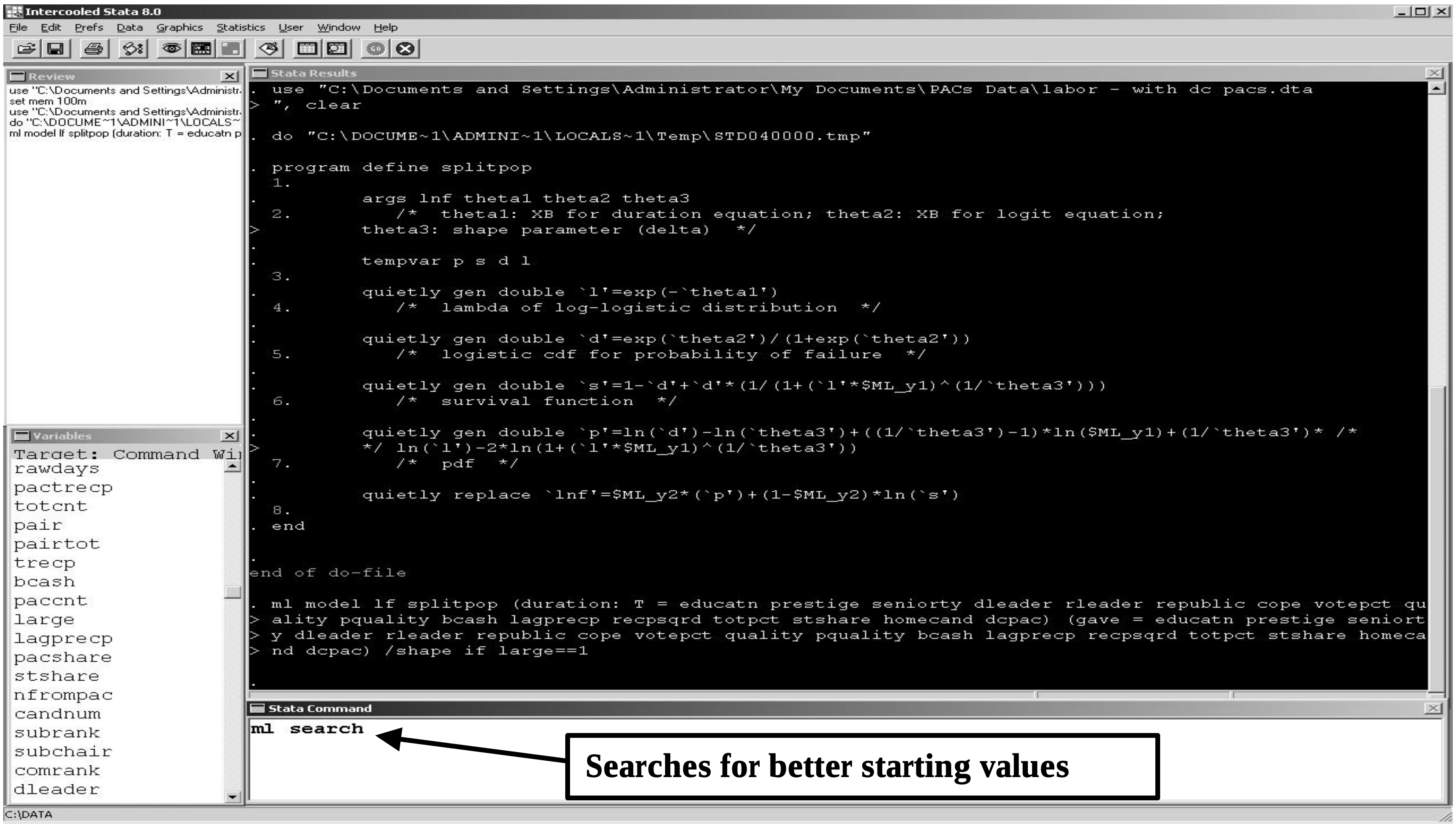
Programming Likelihood Functions: Split-Population Duration Model

- 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.

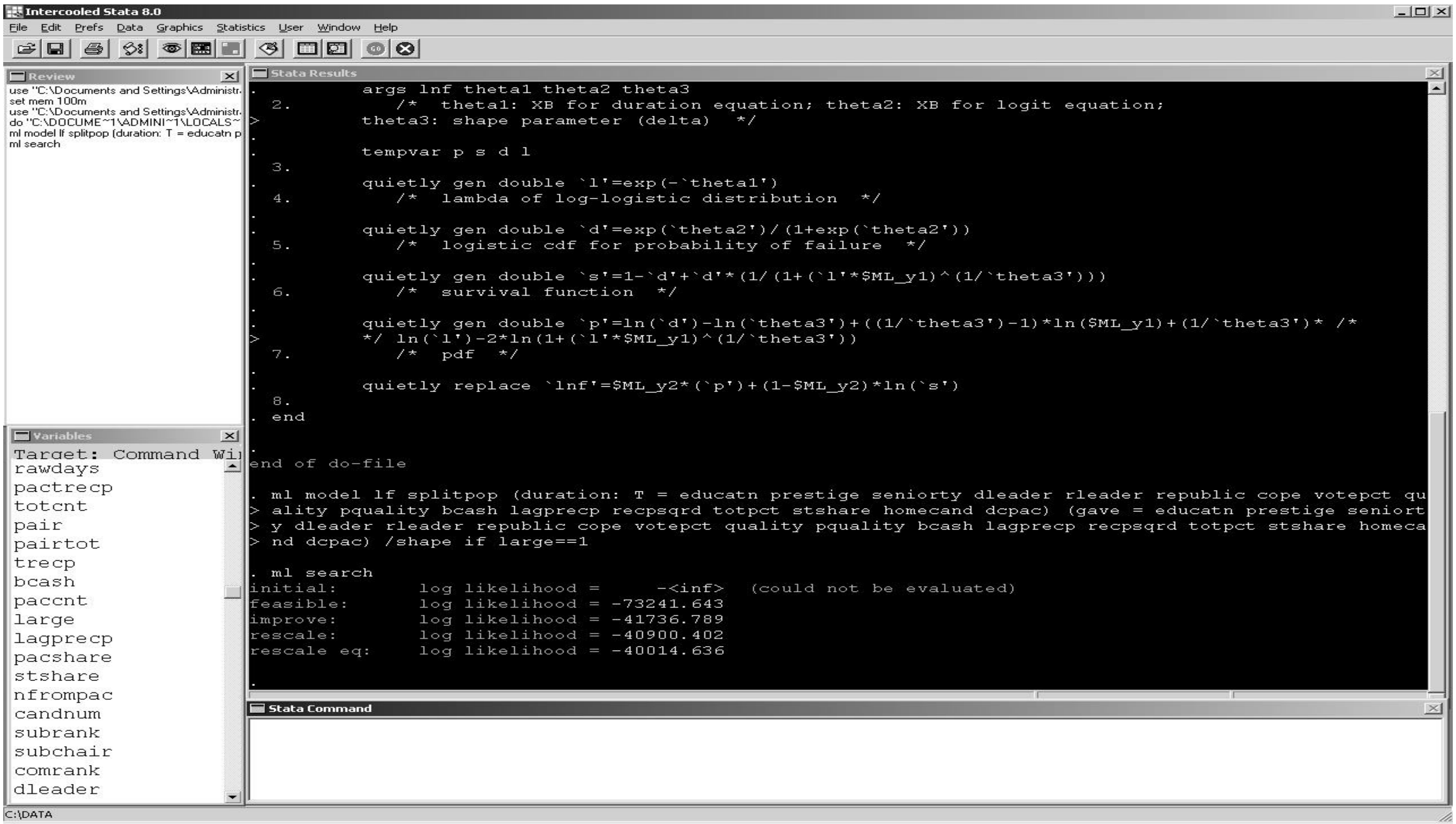
Programming Likelihood Functions: Split-Population Duration Model



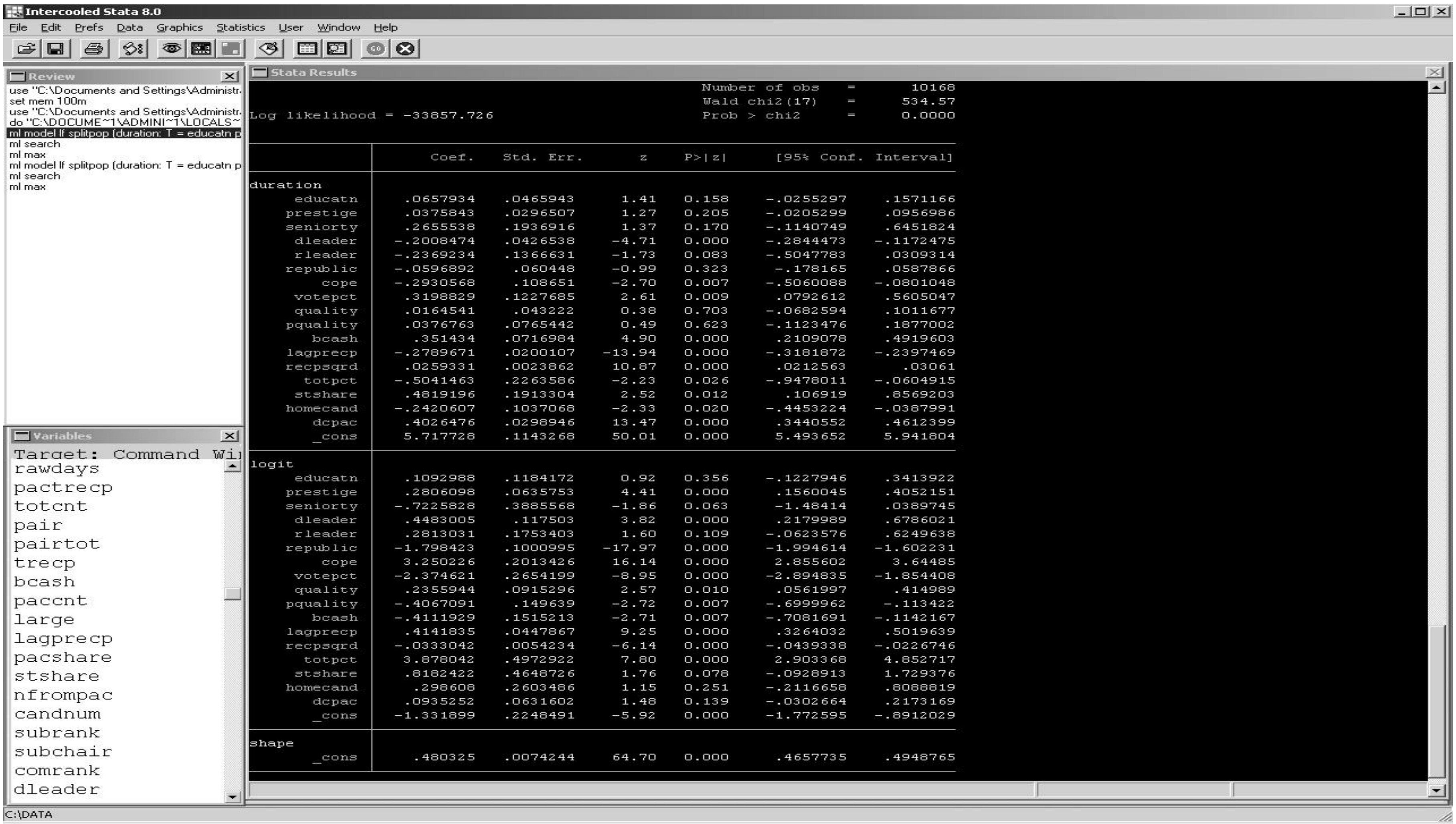
Programming Likelihood Functions: Split-Population Duration Model



Programming Likelihood Functions: Split-Population Duration Model



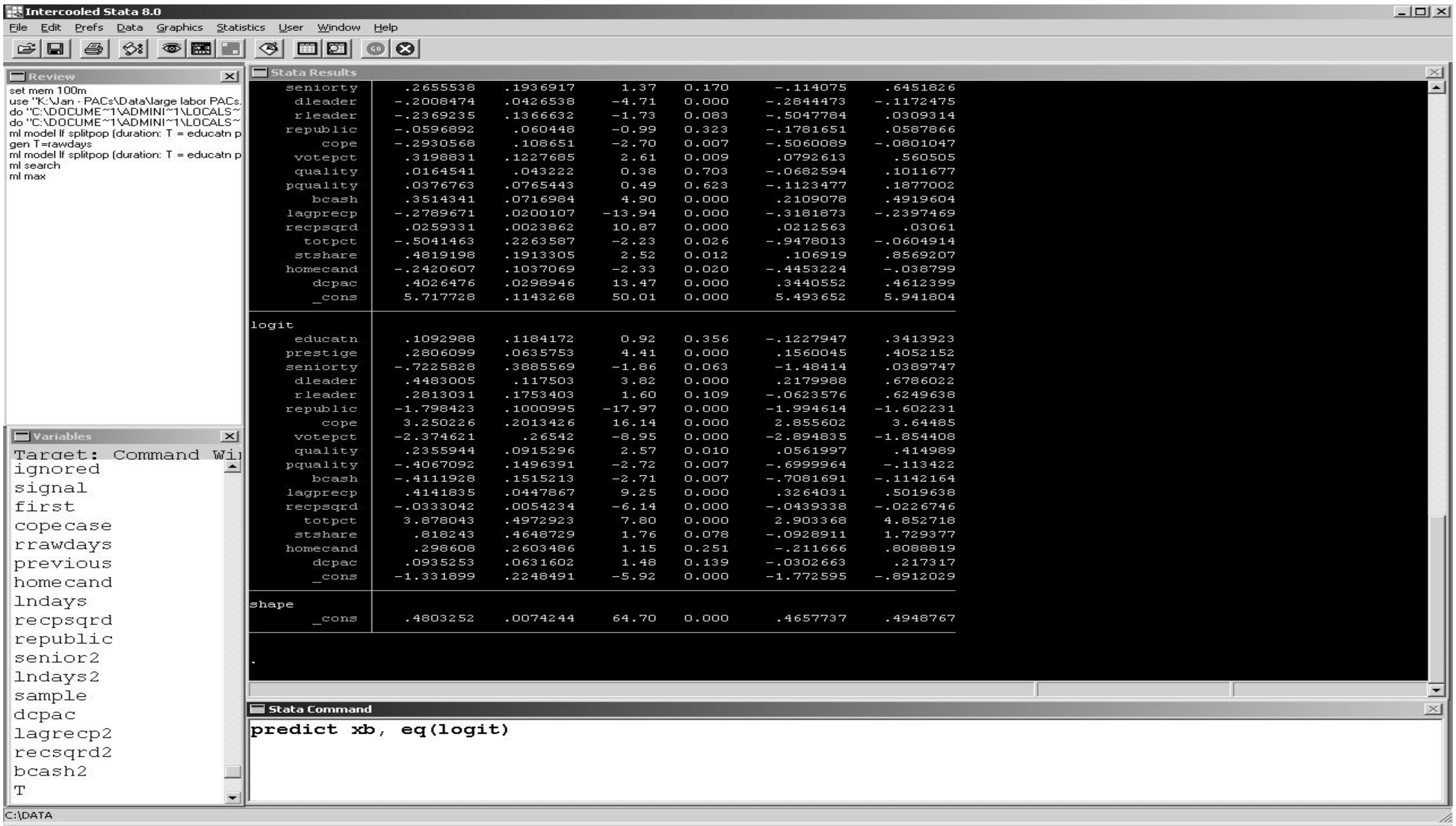
Programming Likelihood Functions: Split-Population Duration Model



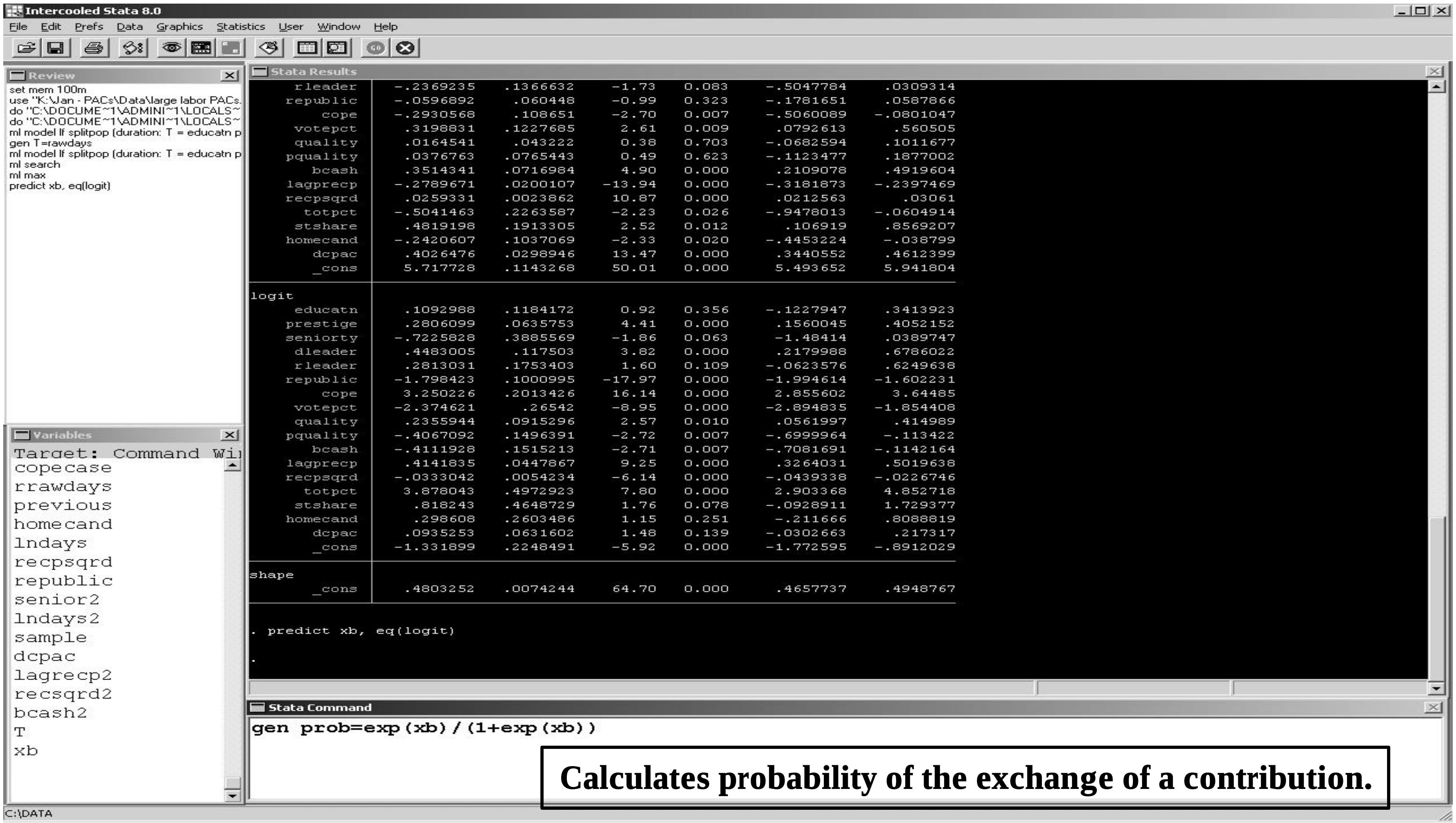
Number of obs = 10168
Wald chi2(17) = 534.57
Log likelihood = -33857.726
Prob > chi2 = 0.0000

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
duration						
educatn	.0657934	.0465943	1.41	0.158	-.0255297	.1571166
prestige	.0375843	.0296507	1.27	0.205	-.0205299	.0956986
seniorty	.2655538	.1936916	1.37	0.170	-.1140749	.6451824
dleader	-.2008474	.0426538	-4.71	0.000	-.2844473	-.1172475
rleader	-.2369234	.1366631	-1.73	0.083	-.5047783	.0309314
republic	-.0596892	.060448	-0.99	0.323	-.178165	.0587866
cope	-.2930568	.108651	-2.70	0.007	-.5060088	-.0801048
votepct	.3198829	.1227685	2.61	0.009	.0792612	.5605047
quality	.0164541	.043222	0.38	0.703	-.0682594	.1011677
pquality	.0376763	.0765442	0.49	0.623	-.1123476	.1877002
bcash	.351434	.0716984	4.90	0.000	.2109078	.4919603
lagprecp	-.2789671	.0200107	-13.94	0.000	-.3181872	-.2397469
recpsqrd	.0259331	.0023862	10.87	0.000	.0212563	.03061
totpct	-.5041463	.2263586	-2.23	0.026	-.9478011	-.0604915
stshare	.4819196	.1913304	2.52	0.012	.106919	.8569203
homecand	-.2420607	.1037068	-2.33	0.020	-.4453224	-.0387991
dcpac	.4026476	.0298946	13.47	0.000	.3440552	.4612399
_cons	5.717728	.1143268	50.01	0.000	5.493652	5.941804
logit						
educatn	.1092988	.1184172	0.92	0.356	-.1227946	.3413922
prestige	.2806098	.0635753	4.41	0.000	.1560045	.4052151
seniorty	-.7225828	.3885568	-1.86	0.063	-1.48414	.0389745
dleader	.4483005	.117503	3.82	0.000	.2179989	.6786021
rleader	.2813031	.1753403	1.60	0.109	-.0623576	.6249638
republic	-1.798423	.1000995	-17.97	0.000	-1.994614	-1.602231
cope	3.250226	.2013426	16.14	0.000	2.855602	3.64485
votepct	-2.374621	.2654199	-8.95	0.000	-2.894835	-1.854408
quality	.2355944	.0915296	2.57	0.010	.0561997	.414989
pquality	-.4067091	.149639	-2.72	0.007	-.6999962	-.113422
bcash	-.4111929	.1515213	-2.71	0.007	-.7081691	-.1142167
lagprecp	.4141835	.0447867	9.25	0.000	.3264032	.5019639
recpsqrd	-.0333042	.0054234	-6.14	0.000	-.0439338	-.0226746
totpct	3.878042	.4972922	7.80	0.000	2.903368	4.852717
stshare	.8182422	.4648726	1.76	0.078	-.0928913	1.729376
homecand	.298608	.2603486	1.15	0.251	-.2116658	.8088819
dcpac	.0935252	.0631602	1.48	0.139	-.0302664	.2173169
_cons	-1.331899	.2248491	-5.92	0.000	-1.772595	-.8912029
shape						
_cons	.480325	.0074244	64.70	0.000	.4657735	.4948765

Programming Likelihood Functions: The Predict Command and Est. Split



Programming Likelihood Functions: The Predict Command and Est. Split



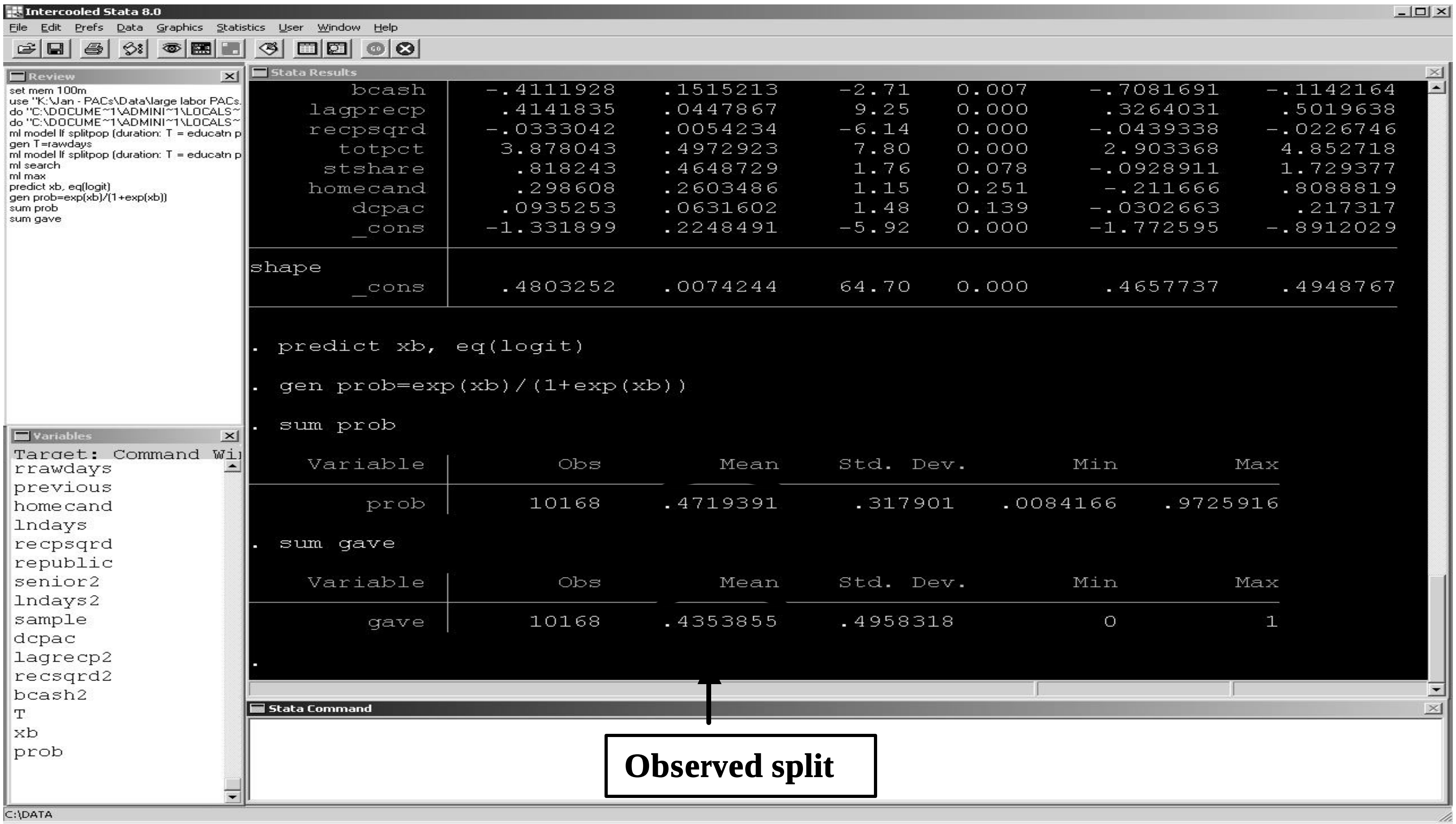
rleader	-.2369235	.1366632	-1.73	0.083	-.5047784	.0309314
republic	-.0596892	.060448	-0.99	0.323	-.1781651	.0587866
cope	-.2930568	.108651	-2.70	0.007	-.5060089	-.0801047
votepct	.3198831	.1227685	2.61	0.009	.0792613	.560505
quality	.0164541	.043222	0.38	0.703	-.0682594	.1011677
pquality	.0376763	.0765443	0.49	0.623	-.1123477	.1877002
bcash	.3514341	.0716984	4.90	0.000	.2109078	.4919604
lagprecp	-.2789671	.0200107	-13.94	0.000	-.3181873	-.2397469
recpsqrd	.0259331	.0023862	10.87	0.000	.0212563	.03061
totpct	-.5041463	.2263587	-2.23	0.026	-.9478013	-.0604914
stshare	.4819198	.1913305	2.52	0.012	.106919	.8569207
homecand	-.2420607	.1037069	-2.33	0.020	-.4453224	-.038799
dcpac	.4026476	.0298946	13.47	0.000	.3440552	.4612399
_cons	5.717728	.1143268	50.01	0.000	5.493652	5.941804
logit						
educatn	.1092988	.1184172	0.92	0.356	-.1227947	.3413923
prestige	.2806099	.0635753	4.41	0.000	.1560045	.4052152
seniorty	-.7225828	.3885569	-1.86	0.063	-1.48414	.0389747
dleader	.4483005	.117503	3.82	0.000	.2179988	.6786022
rleader	.2813031	.1753403	1.60	0.109	-.0623576	.6249638
republic	-1.798423	.1000995	-17.97	0.000	-1.994614	-1.602231
cope	3.250226	.2013426	16.14	0.000	2.855602	3.64485
votepct	-2.374621	.26542	-8.95	0.000	-2.894835	-1.854408
quality	.2355944	.0915296	2.57	0.010	.0561997	.414989
pquality	-.4067092	.1496391	-2.72	0.007	-.6999964	-.113422
bcash	-.4111928	.1515213	-2.71	0.007	-.7081691	-.1142164
lagprecp	.4141835	.0447867	9.25	0.000	.3264031	.5019638
recpsqrd	-.0333042	.0054234	-6.14	0.000	-.0439338	-.0226746
totpct	3.878043	.4972923	7.80	0.000	2.903368	4.852718
stshare	.818243	.4648729	1.76	0.078	-.0928911	1.729377
homecand	.298608	.2603486	1.15	0.251	-.211666	.8088819
dcpac	.0935253	.0631602	1.48	0.139	-.0302663	.217317
_cons	-1.331899	.2248491	-5.92	0.000	-1.772595	-.8912029
shape						
_cons	.4803252	.0074244	64.70	0.000	.4657737	.4948767

```
. predict xb, eq(logit)

.
```


Calculates probability of the exchange of a contribution.

Programming Likelihood Functions: The Predict Command and Est. 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!