Programming and Post-Estimation

- Bootstrapping
- Monte Carlo
- Post-Estimation Simulation (Clarify)
- Extending Clarify to Other Models
  - Censored Probit Example

What is Bootstrapping?

- A computer-simulated nonparametric technique for making inferences about a population parameter based on sample statistics.
- If the sample is a good approximation of the population, the sampling distribution of interest can be estimated by generating a large number of new samples from the original.
- Useful when no analytic formula for the sampling distribution is available.

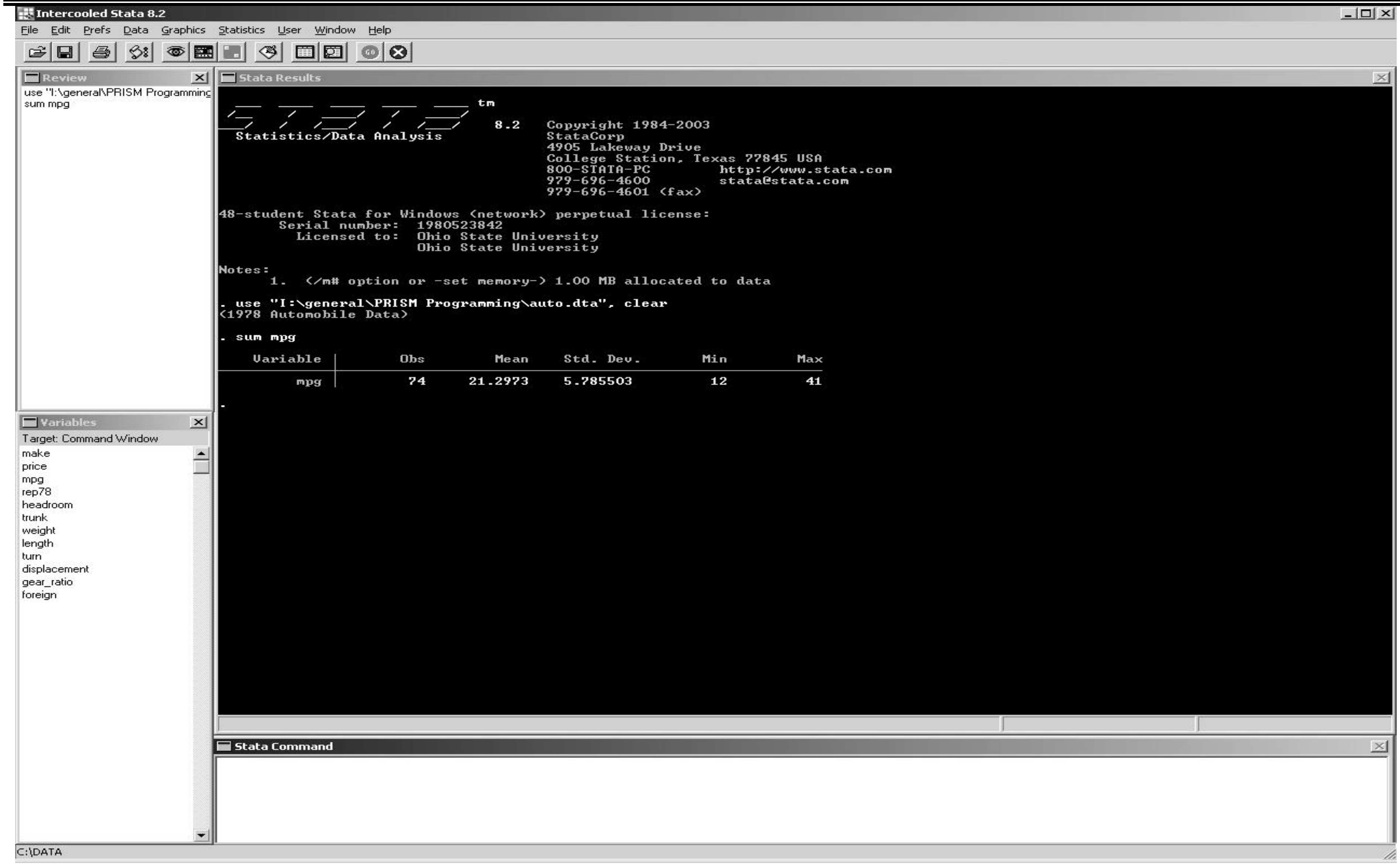
How do I do it?

$$\hat{SE}_B = \sqrt{\sum_{b=1}^{B}\left[s\left(x^B\right) - \sum_{b=1}^{B} s\left(x^b\right)/B\right]^2 / (B-1)}$$

1. Obtain a Sample from the population of interest. Call this **x** = (x_1, x_2, … , x_n).
2. Re-sample based on **x** by randomly sampling with replacement from it.
3. Generate many such samples, x^1, x^2, …, x^B – each of length n.
4. Estimate the desired parameter in each sample, s(x^1), s(x^2), …, s(x^B).
5. For instance the bootstrap estimate of the standard error is the standard deviation of the bootstrap replications.

Example: Standard Error of a Sample Mean

Canned in Stata



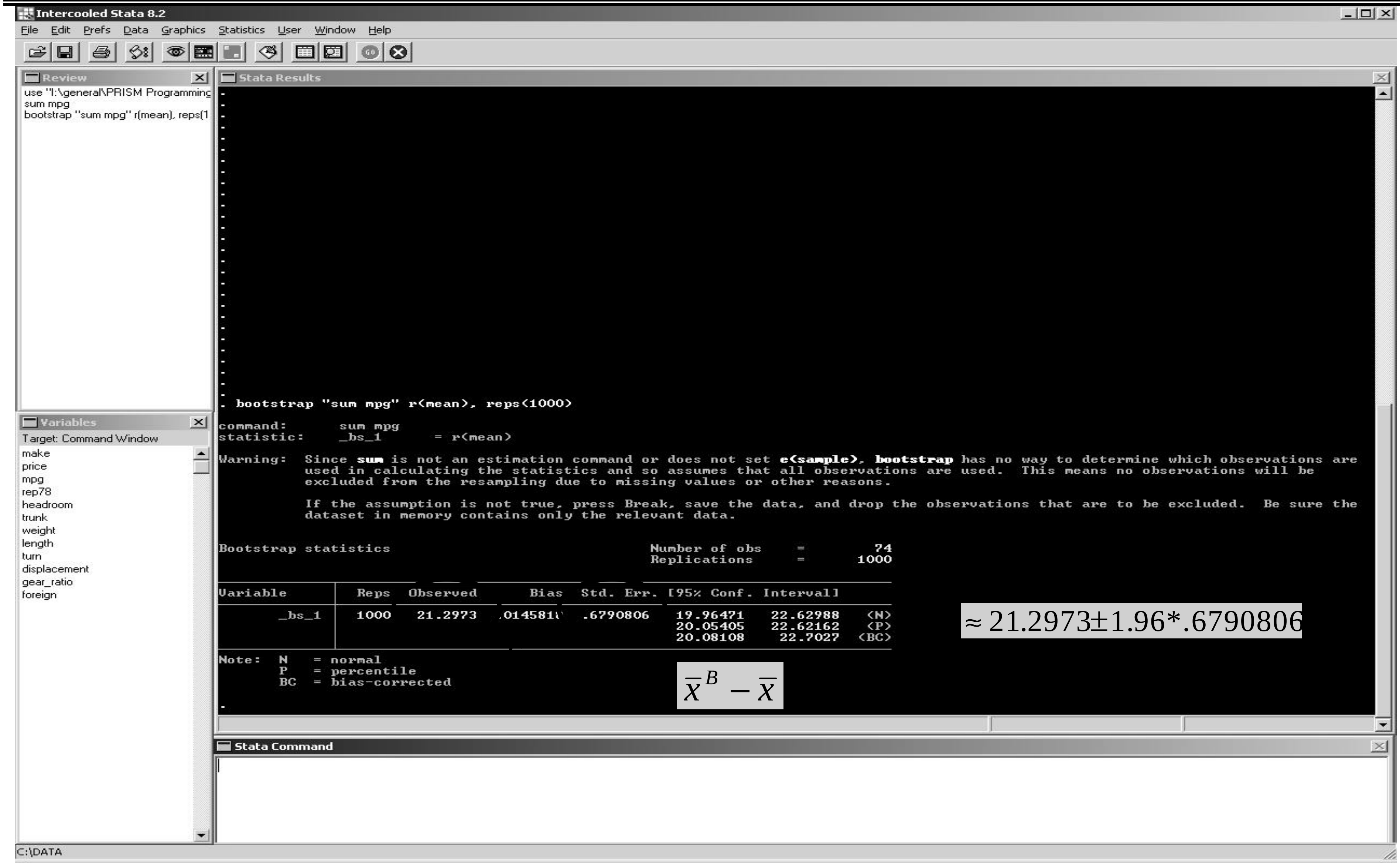
```
use "I:\general\PRISM Programming
sum mpg
```

```
                                   8.2   Copyright 1984-2003
  Statistics/Data Analysis               StataCorp
                                         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)

48-student Stata for Windows (network) perpetual license:
       Serial number:  1980523842
         Licensed to:  Ohio State University
                       Ohio State University

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

. use "I:\general\PRISM Programming\auto.dta", clear
(1978 Automobile Data)

. sum mpg

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         mpg |        74     21.2973    5.785503         12         41

.
```

```
make
price
mpg
rep78
headroom
trunk
weight
length
turn
displacement
gear_ratio
foreign
```

Example: Standard Error of a Sample Mean

Canned in Stata



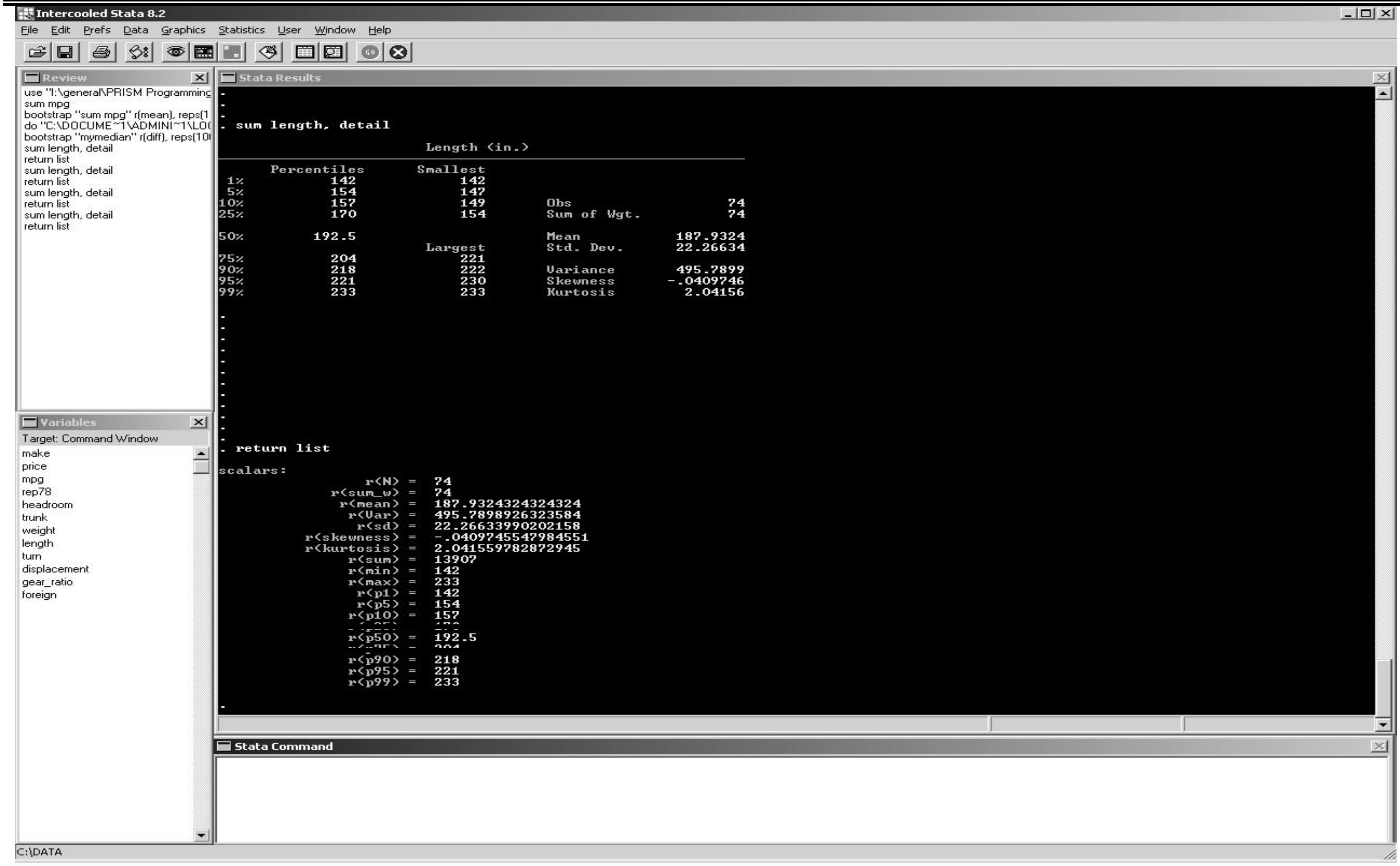
```
use "I:\general\PRISM Programming
sum mpg
bootstrap "sum mpg" r(mean), reps(1
```

```
. bootstrap "sum mpg" r(mean), reps(1000)

command:      sum mpg
statistic:    _bs_1      = r(mean)

Warning:  Since sum is not an estimation command or does not set e(sample), bootstrap has no way to determine which observations are
          used in calculating the statistics and so assumes that all observations are used.  This means no observations will be
          excluded from the resampling due to missing values or other reasons.

          If the assumption is not true, press Break, save the data, and drop the observations that are to be excluded.  Be sure the
          dataset in memory contains only the relevant data.

Bootstrap statistics                              Number of obs    =        74
                                                  Replications     =      1000

------------------------------------------------------------------------------
Variable     |  Reps  Observed      Bias  Std. Err. [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _bs_1 |  1000   21.2973   .014581   .6790806   19.96471   22.62988   (N)
             |                                       20.05405   22.62162   (P)
             |                                       20.08108    22.7027  (BC)
------------------------------------------------------------------------------
Note:  N   = normal
       P   = percentile
       BC  = bias-corrected

.
```

$\approx 21.2973 \pm 1.96 * .6790806$

$\overline{x}^B - \overline{x}$

Example: Difference of Medians Test



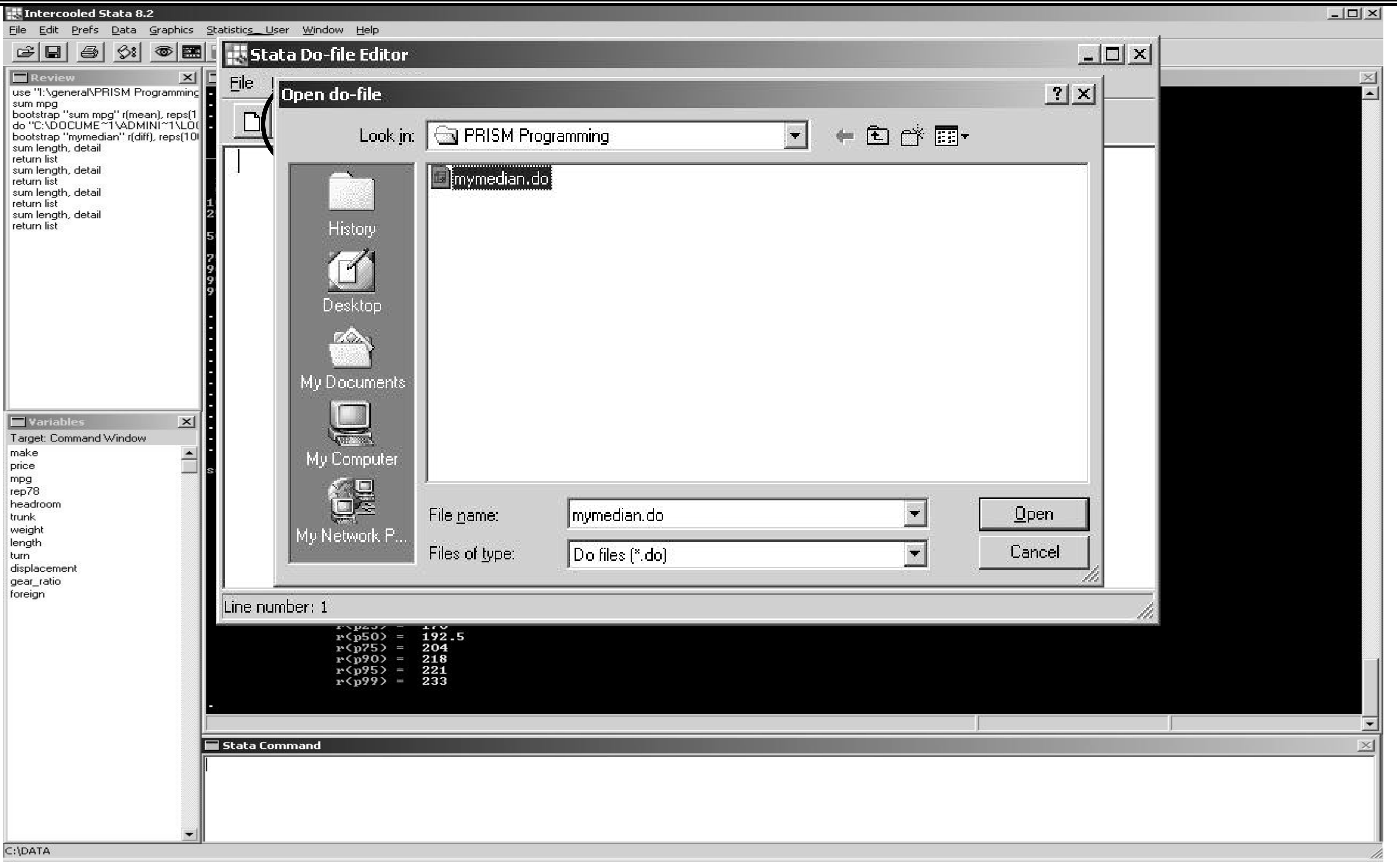
```
use "I:\general\PRISM Programming
sum mpg
bootstrap "sum mpg" r(mean), reps(1
do "C:\DOCUME~1\ADMINI~1\LO(
bootstrap "mymedian" r(diff), reps(10(
sum length, detail
return list
sum length, detail
return list
sum length, detail
return list
sum length, detail
return list
```

Target: Command Window

```
make
price
mpg
rep78
headroom
trunk
weight
length
turn
displacement
gear_ratio
foreign
```

```
.
.
.
. sum length, detail

                       Length (in.)
-------------------------------------------------------------
      Percentiles      Smallest
 1%          142            142
 5%          154            147
10%          157            149       Obs                  74
25%          170            154       Sum of Wgt.          74

50%        192.5                      Mean           187.9324
                        Largest       Std. Dev.      22.26634
75%          204            221
90%          218            222       Variance       495.7899
95%          221            230       Skewness      -.0409746
99%          233            233       Kurtosis        2.04156

.
.
.
.
.
.
.
.
.
.
.
.
. return list

scalars:
                  r(N) =  74
              r(sum_w) =  74
               r(mean) =  187.9324324324324
                r(Var) =  495.7898926323584
                 r(sd) =  22.26633990202158
           r(skewness) =  -.0409745547984551
           r(kurtosis) =  2.041559782872945
                r(sum) =  13907
                r(min) =  142
                r(max) =  233
                 r(p1) =  142
                 r(p5) =  154
                r(p10) =  157
                r(p25) =  170
                r(p50) =  192.5
                r(p75) =  204
                r(p90) =  218
                r(p95) =  221
                r(p99) =  233

.
```

Example: Difference of Medians Test



```
use "I:\general\PRISM Programming
sum mpg
bootstrap "sum mpg" r(mean), reps(1
do "C:\DOCUME~1\ADMINI~1\LO(
bootstrap "mymedian" r(diff), reps(10
sum length, detail
return list
sum length, detail
return list
sum length, detail
return list
sum length, detail
return list
```

Target: Command Window

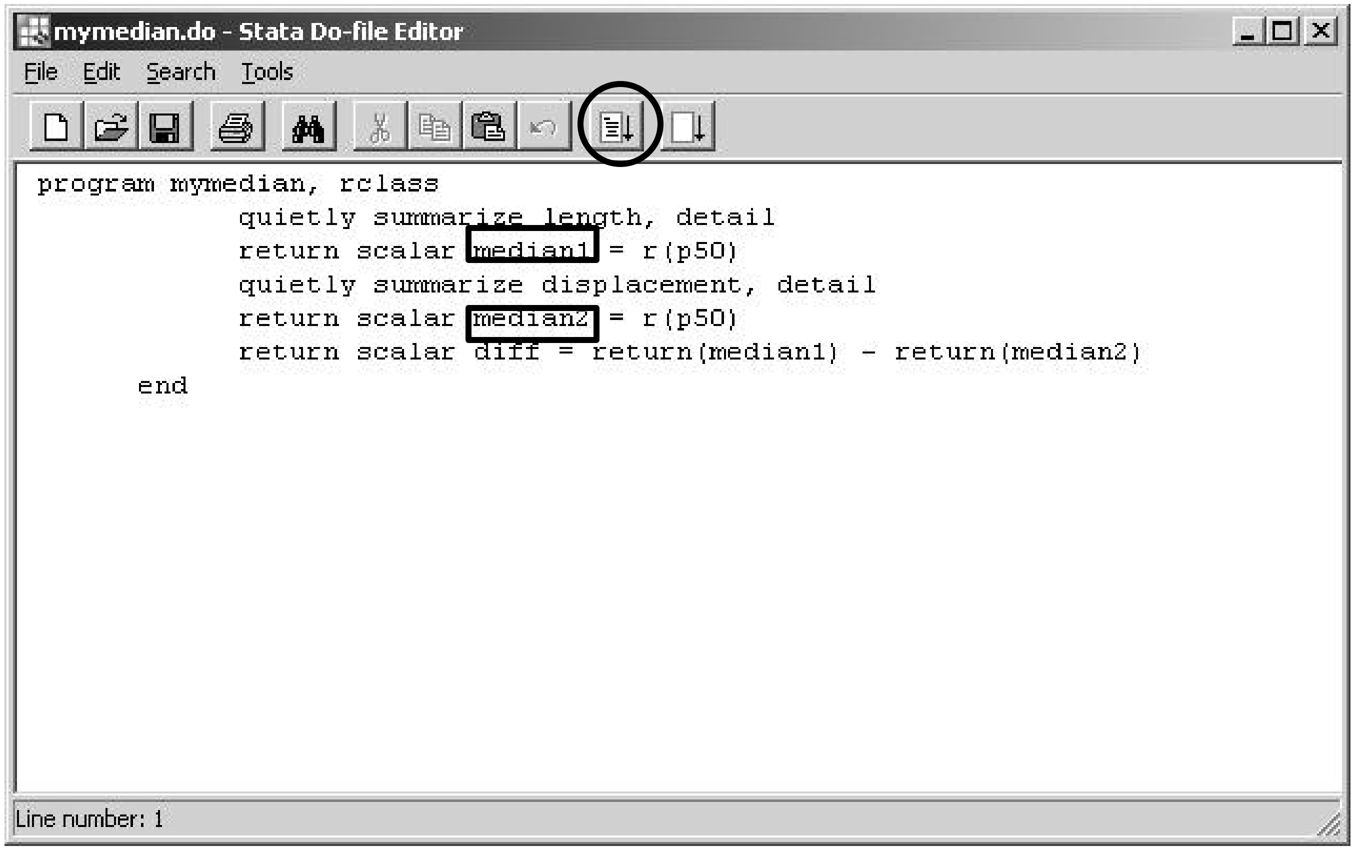
```
make
price
mpg
rep78
headroom
trunk
weight
length
turn
displacement
gear_ratio
foreign
```

File name: mymedian.do

Files of type: Do files (*.do)

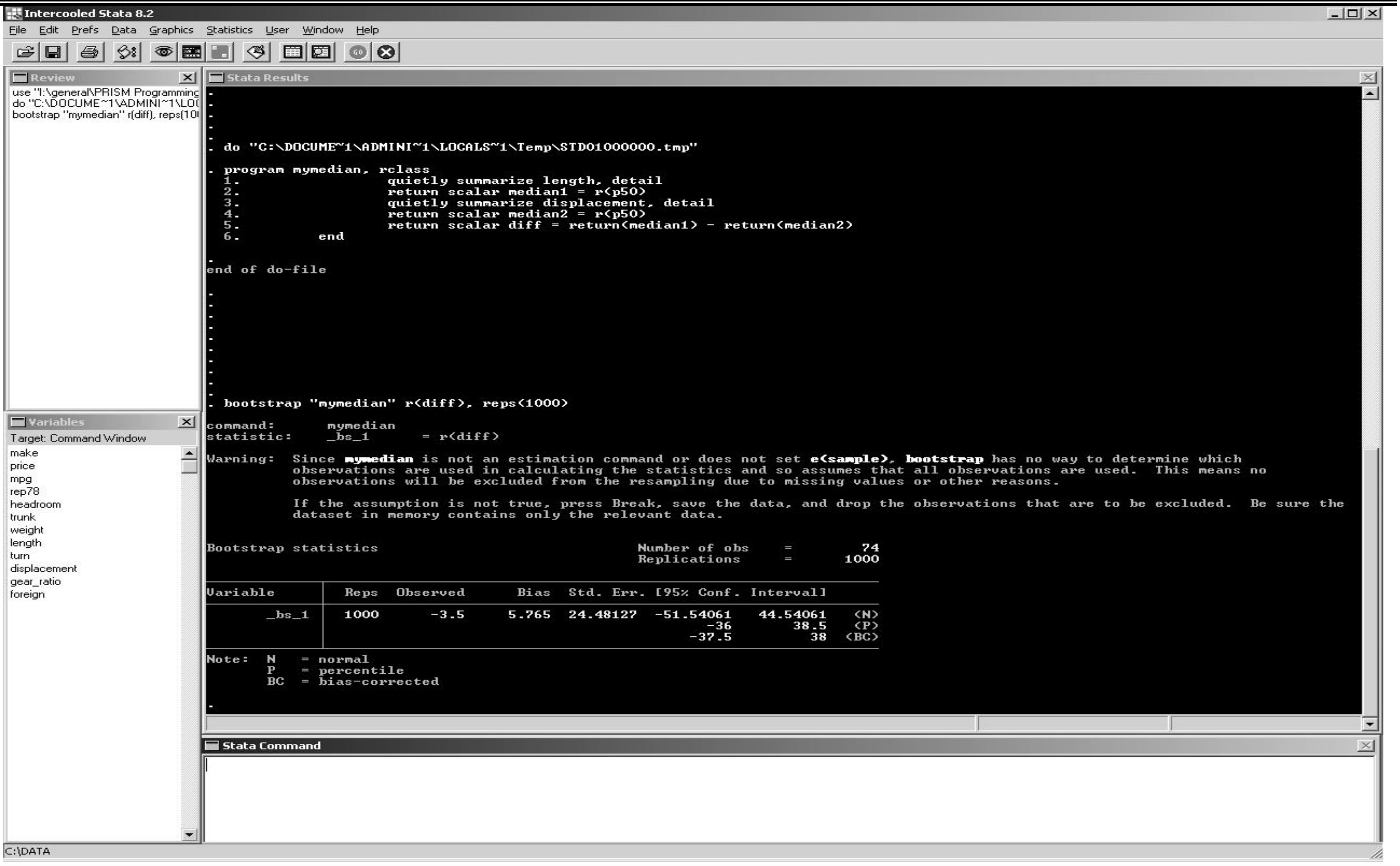
```
r(p50) =   192.5
r(p75) =   204
r(p90) =   218
r(p95) =   221
r(p99) =   233
```

Example: Difference of Medians Test



```
program mymedian, rclass
            quietly summarize length, detail
            return scalar median1 = r(p50)
            quietly summarize displacement, detail
            return scalar median2 = r(p50)
            return scalar diff = return(median1) - return(median2)
      end
```

Example: Difference of Medians Test



```
use "I:\general\PRISM Programming
do "C:\DOCUME~1\ADMINI~1\LO(
bootstrap "mymedian" r(diff), reps(10(
```

```
make
price
mpg
rep78
headroom
trunk
weight
length
turn
displacement
gear_ratio
foreign
```

```
.
.
.
.
.
. do "C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\STD01000000.tmp"

. program mymedian, rclass
  1.                   quietly summarize length, detail
  2.                   return scalar median1 = r(p50)
  3.                   quietly summarize displacement, detail
  4.                   return scalar median2 = r(p50)
  5.                   return scalar diff = return(median1) - return(median2)
  6.           end

.
end of do-file

.
.
.
.
.
.
.
.
.
. bootstrap "mymedian" r(diff), reps(1000)

command:      mymedian
statistic:    _bs_1       = r(diff)

Warning:  Since mymedian is not an estimation command or does not set e(sample), bootstrap has no way to determine which
          observations are used in calculating the statistics and so assumes that all observations are used.  This means no
          observations will be excluded from the resampling due to missing values or other reasons.

          If the assumption is not true, press Break, save the data, and drop the observations that are to be excluded.  Be sure the
          dataset in memory contains only the relevant data.

Bootstrap statistics                              Number of obs    =        74
                                                  Replications     =      1000

------------------------------------------------------------------------------
Variable     |  Reps  Observed       Bias  Std. Err. [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _bs_1 |  1000      -3.5      5.765  24.48127  -51.54061   44.54061   (N)
             |                                            -36       38.5   (P)
             |                                          -37.5         38  (BC)
------------------------------------------------------------------------------
Note:  N   = normal
       P   = percentile
       BC  = bias-corrected

.
```

What is Monte Carlo Simulation?

- Uses the observation of random samples from known populations of simulated data to track the behavior of a statistic.
- If the sampling distribution of a statistic is the density function of values it could take on in a given population, then its estimate is the relative frequency distribution of the values that were actually observed in many samples drawn from that population.
- Since we can generate the population to have any characteristics we wish, Monte Carlos are very flexible.

How do I do it?

1. Determine what the “population” is.
2. Sample from the “population”
3. Calculate the estimator of interest $(\hat{q})$. Save this value.
4. Repeat steps 2 and 3 many times.
5. Construct the frequency distribution of the $(\hat{q})$ values. This is the Monte Carlo estimate of the sampling distribution of q under the conditions you specified for the population.

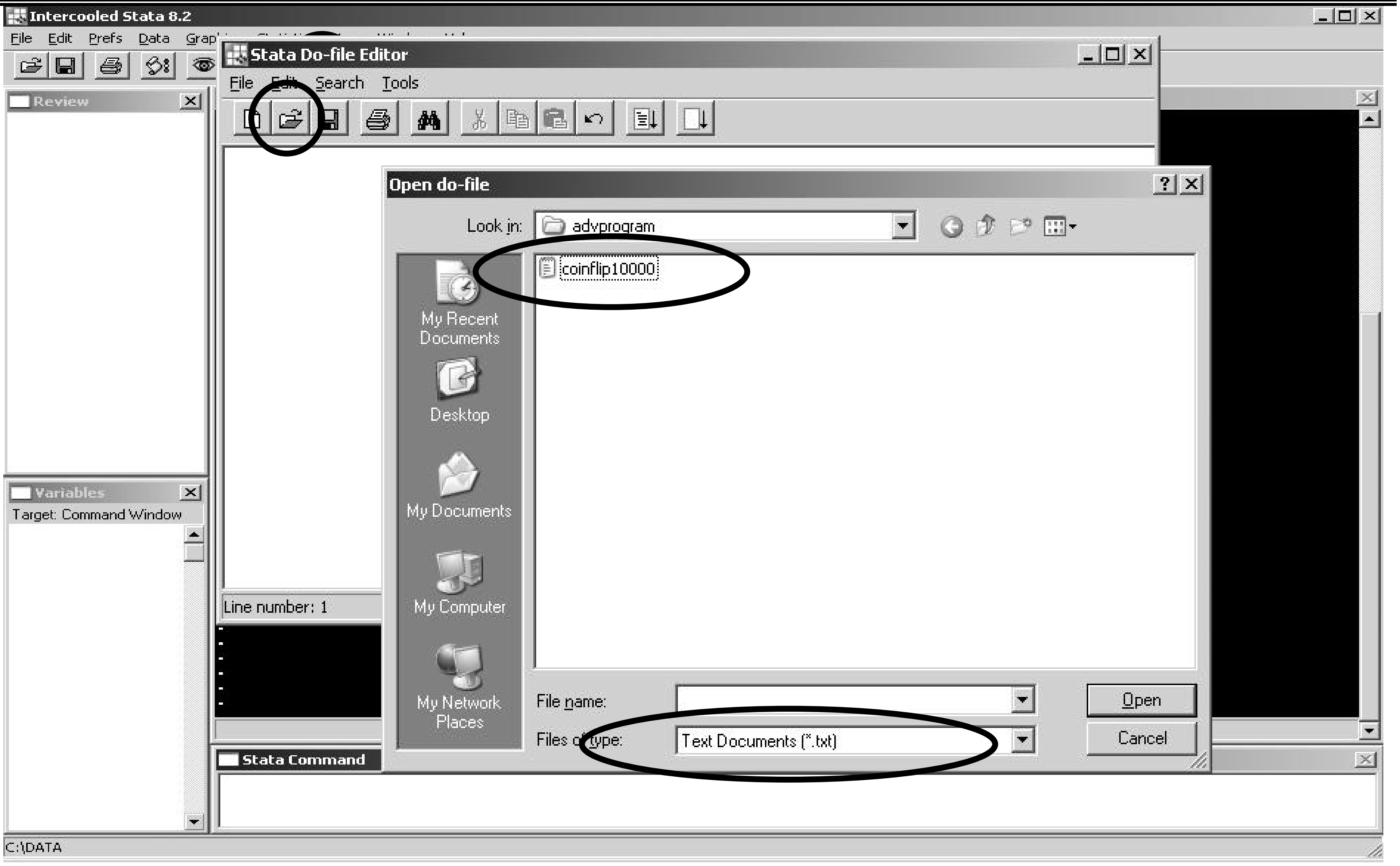
An Example: Coin Toss

- If you toss a fair coin 10 times, what is the probability of obtaining exactly 3 heads?
- By the binomial probability distribution:

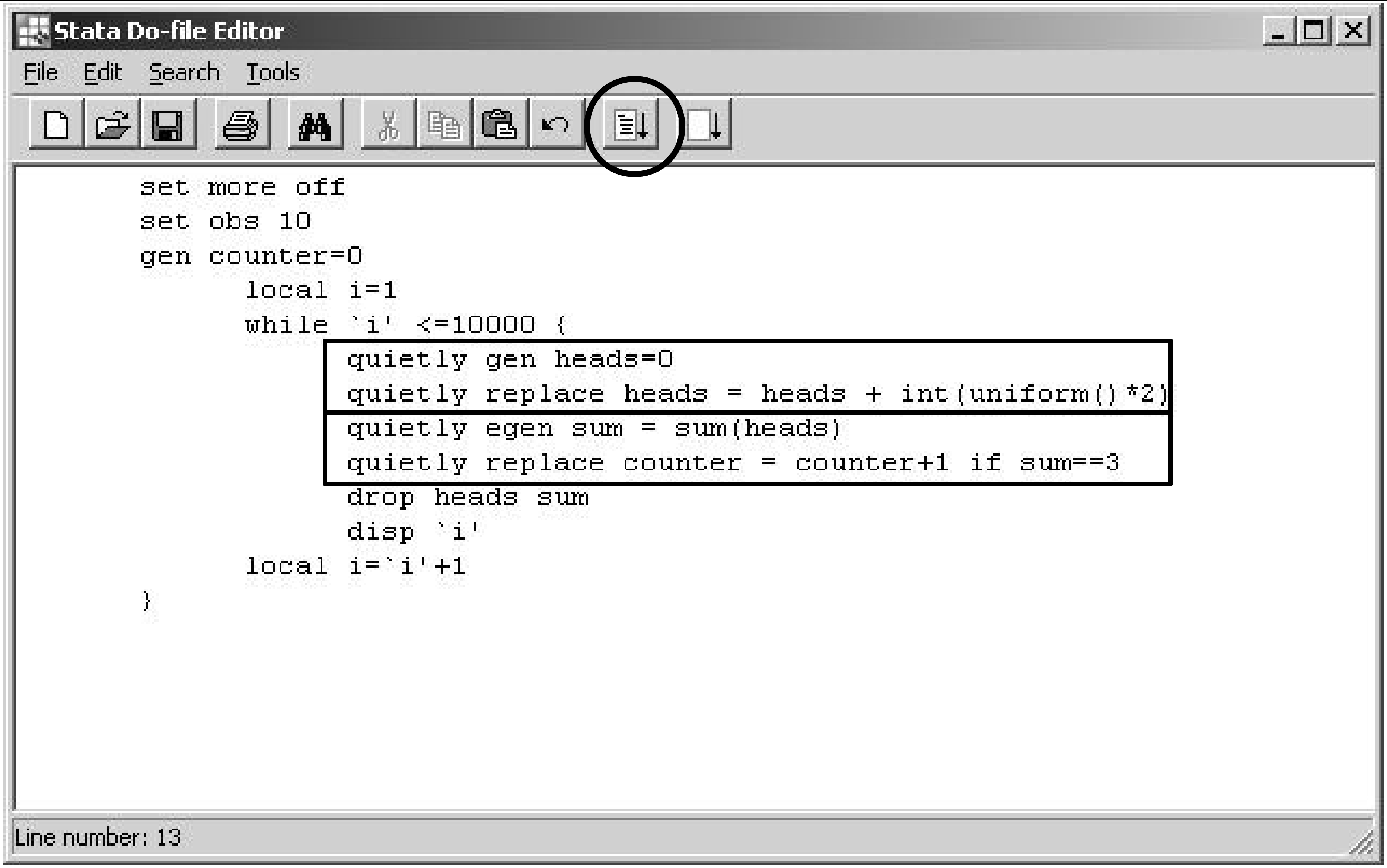
$$= \frac{10!}{3!\left(10-3\right)!} 0.5^{3} * 0.5^{7} \approx 0.1172$$

However, if we didn’t know how to use the binomial distribution, we could toss a fair coin 10 times in a repeated number of trials and simulate this probability.

Coin Toss Monte Carlo



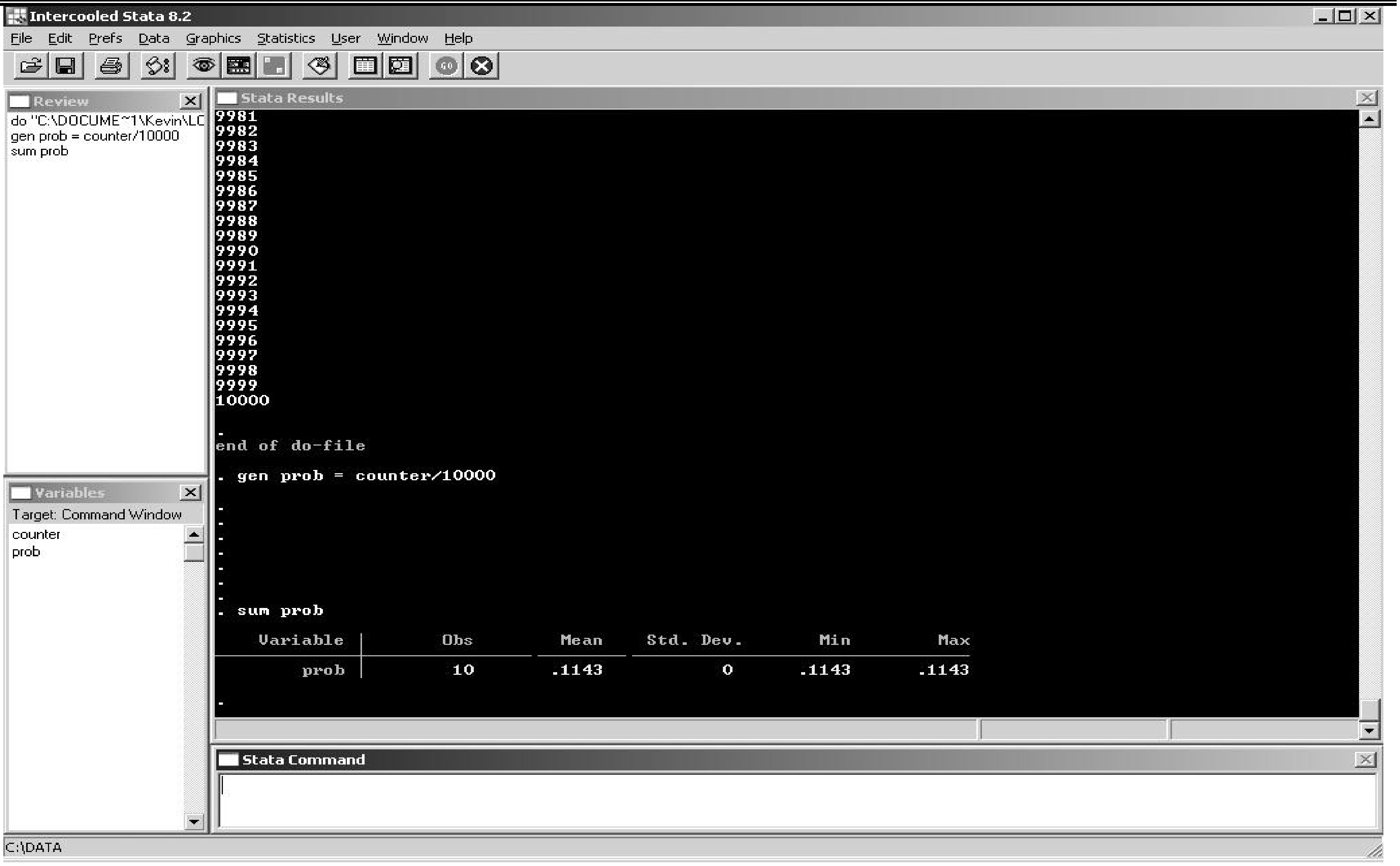
Coin Toss Monte Carlo



```
        set more off
        set obs 10
        gen counter=0
              local i=1
              while `i' <=10000 {
                    quietly gen heads=0
                    quietly replace heads = heads + int(uniform()*2)
                    quietly egen sum = sum(heads)
                    quietly replace counter = counter+1 if sum==3
                    drop heads sum
                    disp `i'
              local i=`i'+1
        }
```

Line number: 13

Coin Toss Monte Carlo



```
do "C:\DOCUME~1\Kevin\LC
gen prob = counter/10000
sum prob
```

```
9981
9982
9983
9984
9985
9986
9987
9988
9989
9990
9991
9992
9993
9994
9995
9996
9997
9998
9999
10000

.
end of do-file

. gen prob = counter/10000

.
.
.
.
.
.
.
. sum prob

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        prob |        10       .1143           0      .1143      .1143

.
```

Programming the Coin Toss Monte Carlo

- Allows us to automate the experiment, controlling:

✓ The number of tosses

(e.g. 10, or something else)

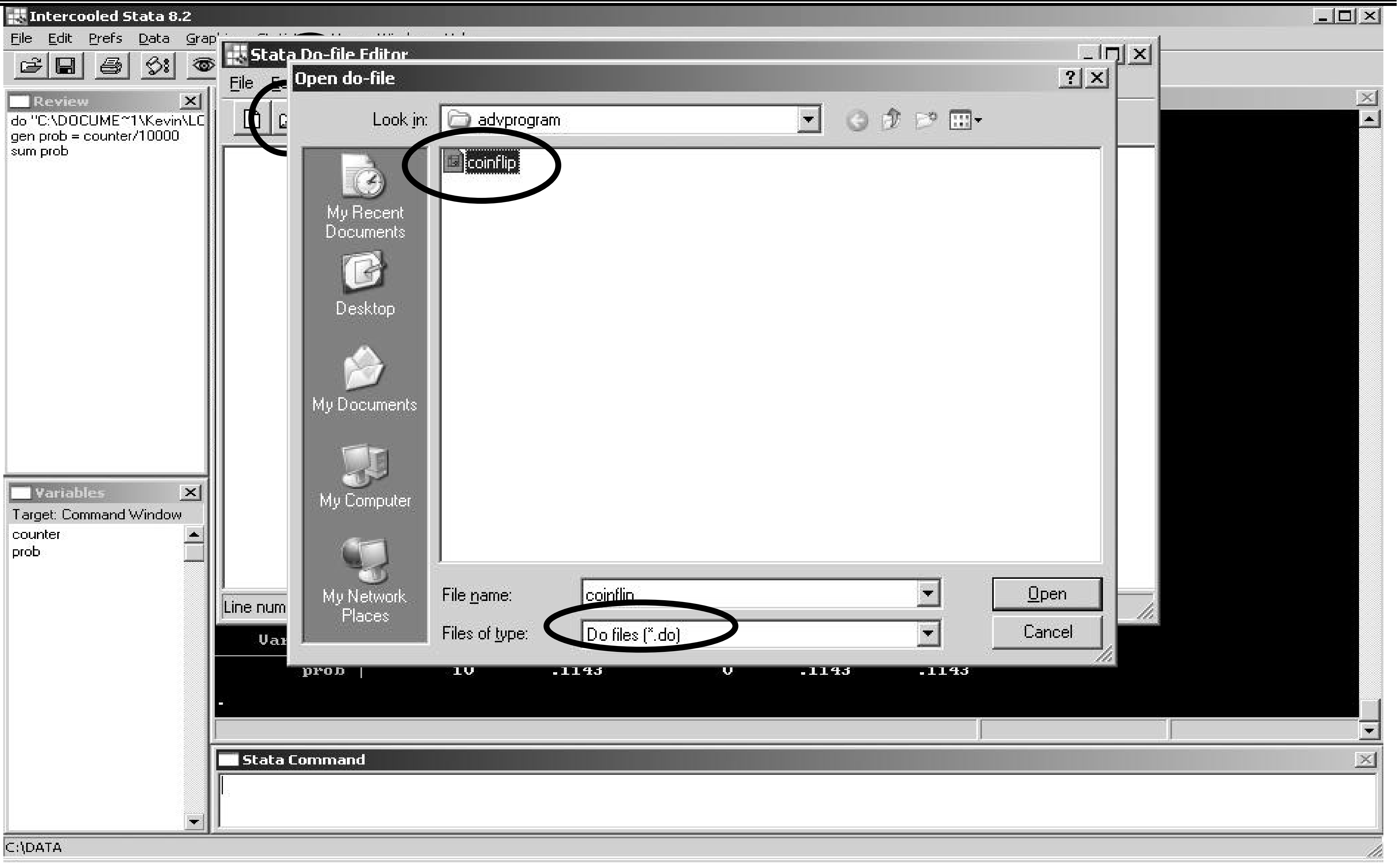
✓ The number of trials

(e.g. 10,000 or something else)

✓ The number of successes

(e.g. 3 or something else)

Coin Toss Program



Intercooled Stata 8.2

Review

```
do "C:\DOCUME~1\Kevin\LC
gen prob = counter/10000
sum prob
```

Variables

Target: Command Window

counter
prob

Stata Do-file Editor

Open do-file

Look in: advprogram

coinflip

File name: coinflip

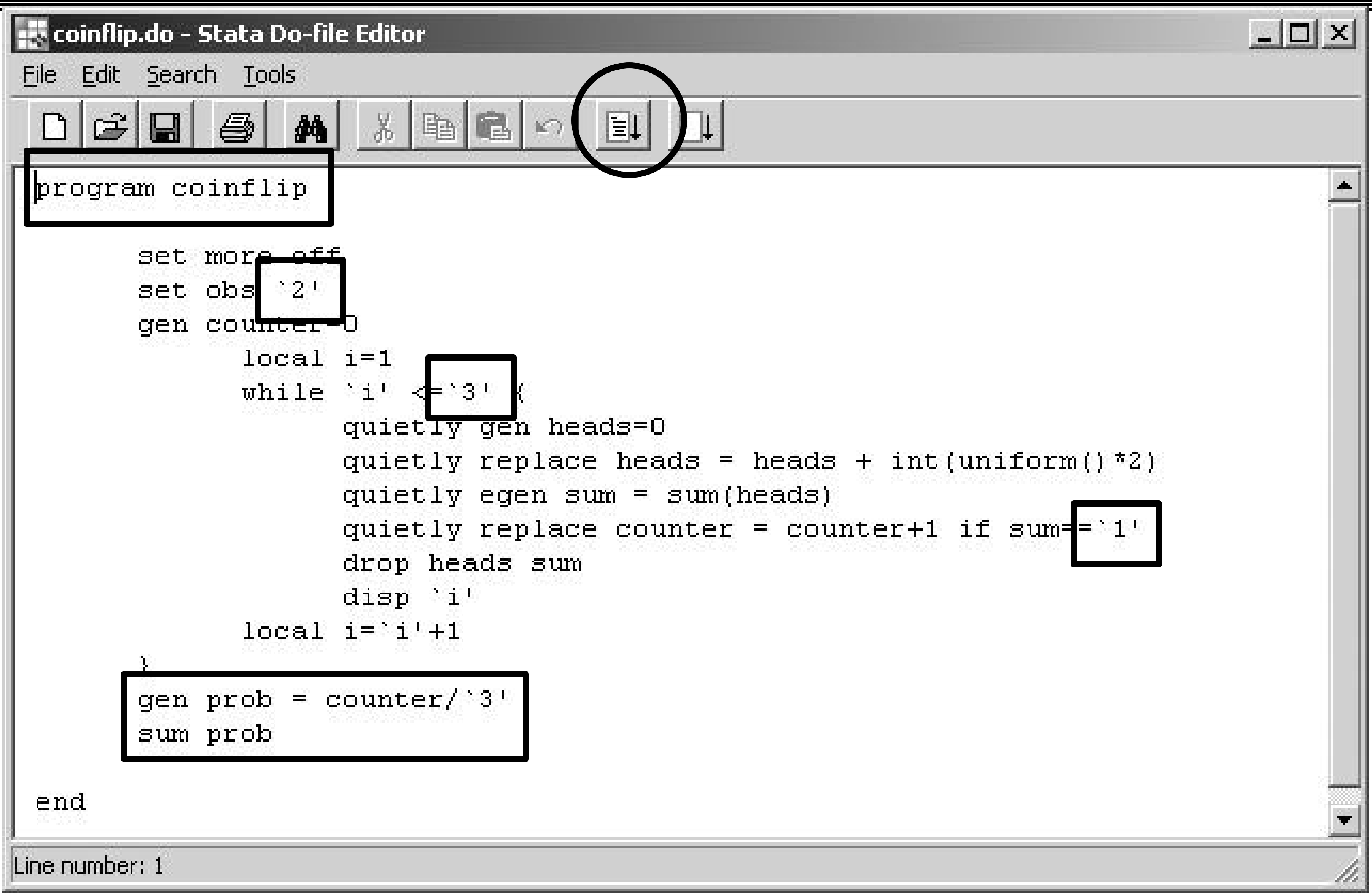
Files of type: Do files (*.do)

```
prob |        10        .1143           0       .1143       .1143
```

Stata Command

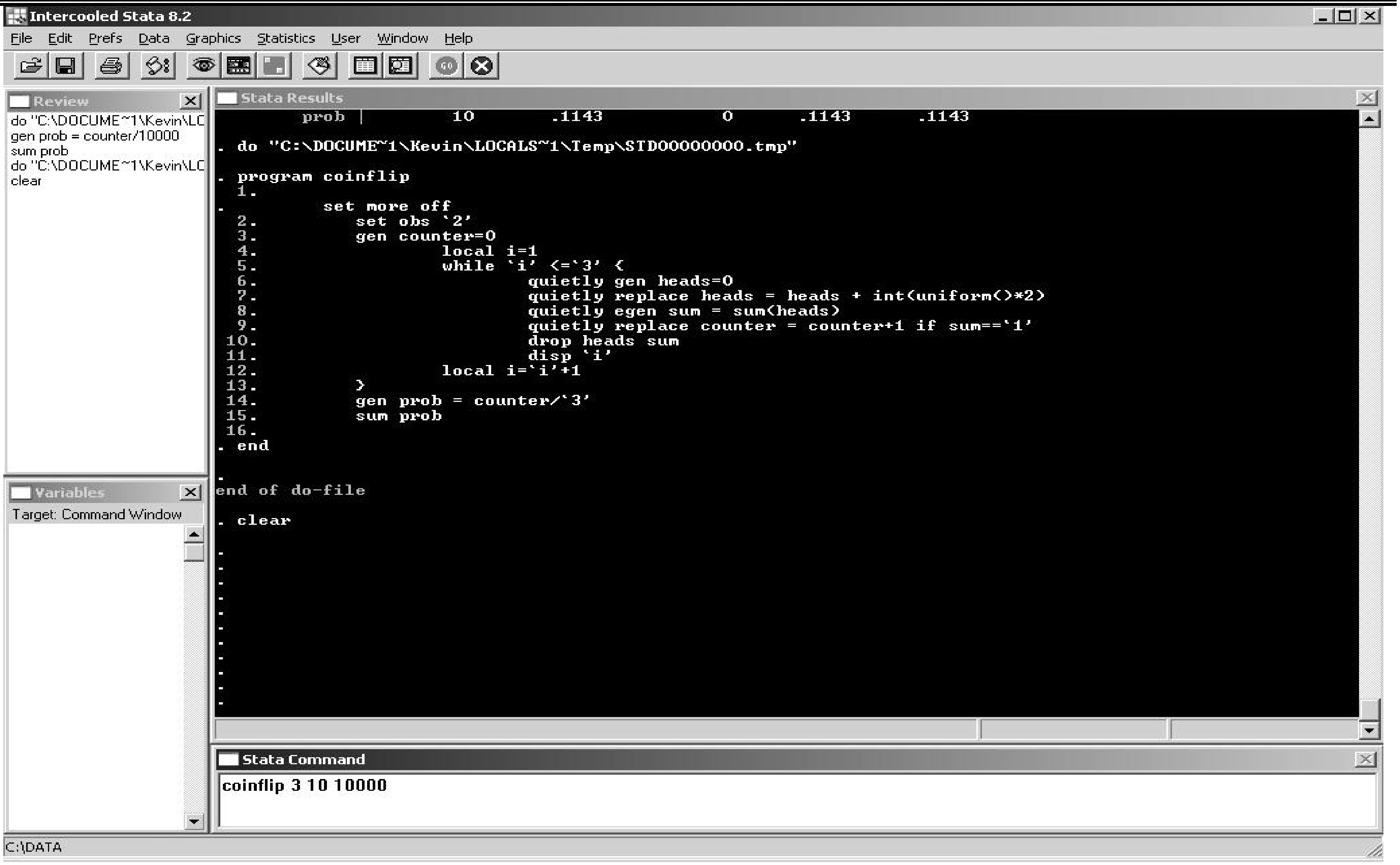
C:\DATA

Coin Toss Program



```
program coinflip

      set more off
      set obs `2'
      gen counter=0
            local i=1
            while `i' <=`3' {
                  quietly gen heads=0
                  quietly replace heads = heads + int(uniform()*2)
                  quietly egen sum = sum(heads)
                  quietly replace counter = counter+1 if sum==`1'
                  drop heads sum
                  disp `i'
            local i=`i'+1
      }
      gen prob = counter/`3'
      sum prob

end
```

Coin Toss Program

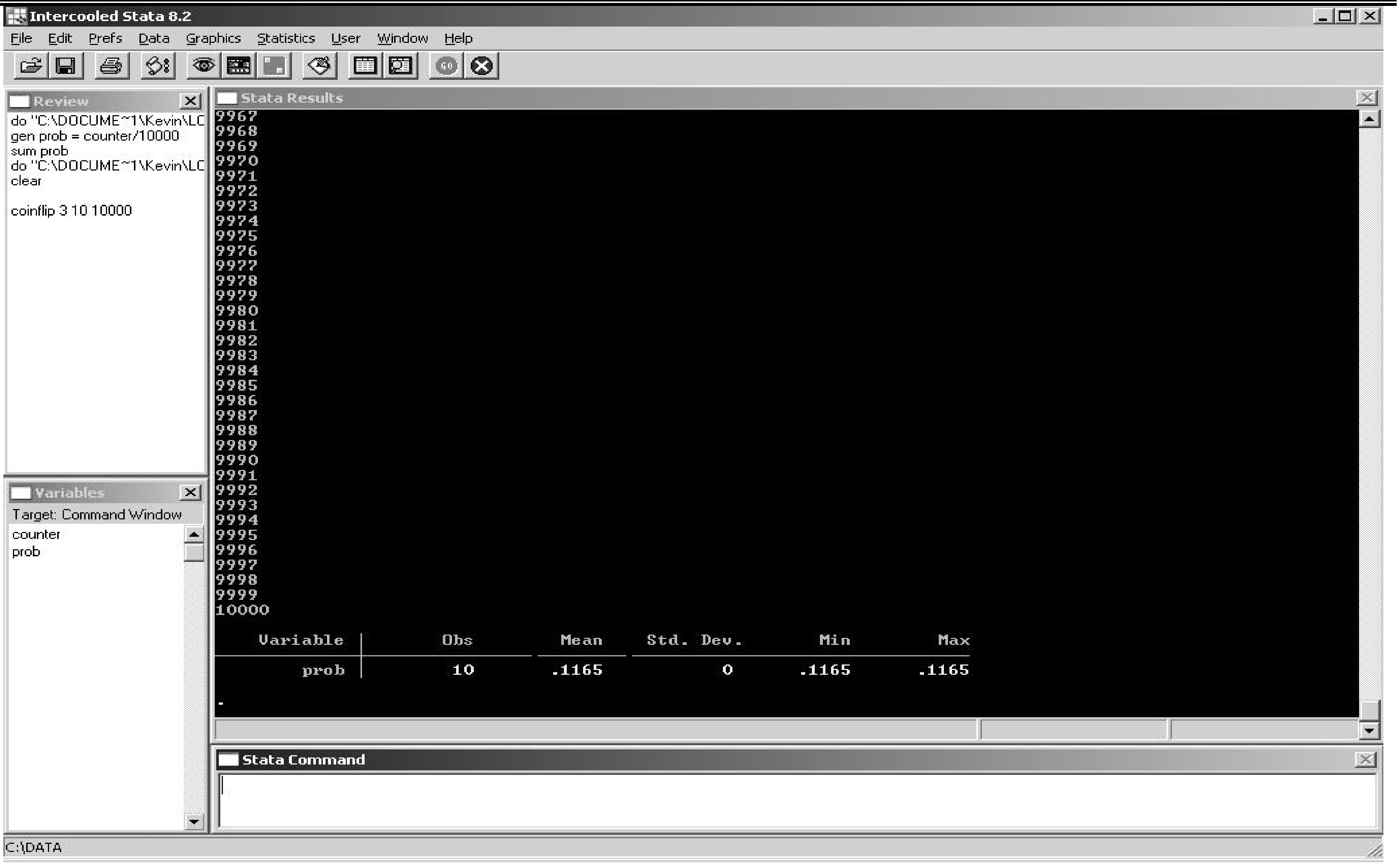


```
do "C:\DOCUME~1\Kevin\LC
gen prob = counter/10000
sum prob
do "C:\DOCUME~1\Kevin\LC
clear
```

```
        prob |        10        .1143           0       .1143       .1143

. do "C:\DOCUME~1\Kevin\LOCALS~1\Temp\STD00000000.tmp"

. program coinflip

.         set more off
            set obs `2'
            gen counter=0
                    local i=1
                    while `i' <=`3' {
                            quietly gen heads=0
                            quietly replace heads = heads + int(uniform()*2)
                            quietly egen sum = sum(heads)
                            quietly replace counter = counter+1 if sum==`1'
                            drop heads sum
                            disp `i'
                    local i=`i'+1
            }
            gen prob = counter/`3'
            sum prob

. end

.
end of do-file

. clear

.
.
.
.
.
.
.
.
.
.
.
```

```
coinflip 3 10 10000
```

Coin Toss Program



```
do "C:\DOCUME~1\Kevin\LC
gen prob = counter/10000
sum prob
do "C:\DOCUME~1\Kevin\LC
clear

coinflip 3 10 10000
```

```
9967
9968
9969
9970
9971
9972
9973
9974
9975
9976
9977
9978
9979
9980
9981
9982
9983
9984
9985
9986
9987
9988
9989
9990
9991
9992
9993
9994
9995
9996
9997
9998
9999
10000

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        prob |        10       .1165           0      .1165      .1165

.
```

Another Monte Carlo Application:

The Logic of Post Estimation Simulation

So, you estimate a model… and you want to say something *substantive* with quantities of interest:

Predicted or Expected Values of DV = $X_m \hat{b}$

First Differences = $X_{+s} \hat{b} - X_m \hat{b}$

The problem is that our $\hat{b}s$ are uncertain!

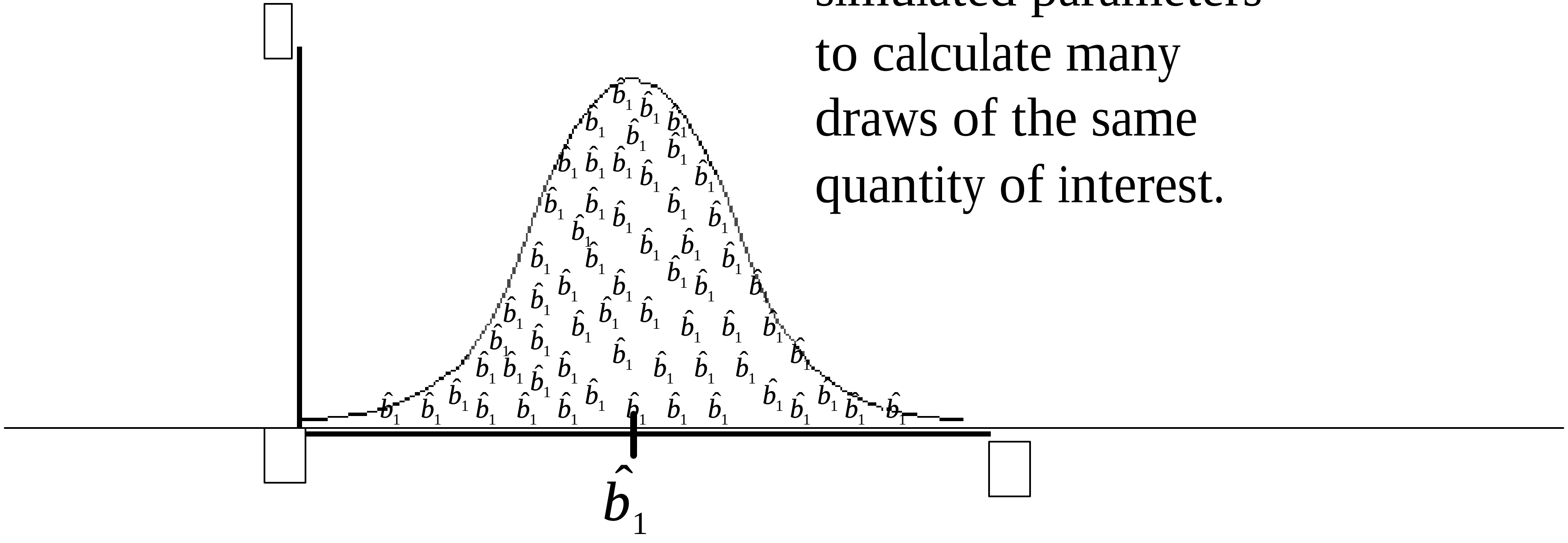
The solution is we know how uncertain.

$$\frac{\hat{b}_1}{\hat{s}_1}$$

Monte Carlo Simulation of Parameters

In order to capture the uncertainty, we draw simulated $\hat{b}s$ from the multivariate* normal distribution.

Then we use these simulated parameters to calculate many draws of the same quantity of interest.



Standard Deviation = $\hat{s}_1$

Simulating Quantities of Interest

In practice…

$$Y_i \sim f(q_i, a), \qquad q_i = g(X_i, b)$$

$$Y_i \sim N(m_i, s^2), \qquad m_i = g(X_i, b) = b_0 + X_{i1} b_1 + X_{i2} b_2 + \cdots$$

$$\hat{g} = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \vdots \\ \hat{a}_1 \end{bmatrix} \qquad \hat{V}(\hat{g}) = \begin{bmatrix} v_{\hat{b}_{11}} & v_{\hat{b}_{12}} & \cdots & v_{\hat{b}_2 \hat{a}} \\ v_{\hat{b}_{21}} & v_{\hat{b}_{22}} & \cdots & v_{\hat{b}_2 \hat{a}} \\ \vdots & \vdots & & \vdots \\ v_{\hat{a}\hat{b}_1} & v_{\hat{a}\hat{b}2} & \cdots & v_{\hat{a}} \end{bmatrix}$$

we simulate parameters with M draws from the multivariate normal distribution… $\tilde{g} \sim N(\hat{g}, \hat{V})$

$$\begin{bmatrix} \tilde{b}_{11} \\ \tilde{b}_{21} \\ \vdots \\ \tilde{a}_1 \end{bmatrix} \begin{bmatrix} \tilde{b}_{12} \\ \tilde{b}_{22} \\ \vdots \\ \tilde{a}_2 \end{bmatrix} \cdots \begin{bmatrix} \tilde{b}_{1M} \\ \tilde{b}_{2M} \\ \vdots \\ \tilde{a}_M \end{bmatrix}$$

- Choose a starting scenario, Xc .
- Draw one value of $\tilde{g}$, and compute $\tilde{q}_c = g(X_c, \tilde{b})$.
- Simulate the outcome $\tilde{Y}_c$, by taking a random draw from $f(\tilde{q}_c, \tilde{a})$.
- Repeat M times to get the distribution of Y_c .

Clarify (King et. al. *AJPS* 1999)

- *estsimp* – <u>est</u>imates the model and <u>sim</u>ulates the <u>p</u>arameters
  - This command **<u>must</u>** precede your regression command
  - e.g.: *estsimp logit y x1 x2 x3 x4*
  - This will save simulated βs to your dataset!
- *setx* – <u>set</u>s the values for the IVs (the <u>X</u>s)
  - Used <u>after</u> model estimation to set values of the Xs
  - e.g.: *setx x1 mean x2 p20 x3 .4 x4[16], nocwdel*
  - functions = mean|median|min|max|p#|math|#|‘macro’|varname[#]
  - reset values by re-issuing the command, e.g.: *setx x1 median*
- *simqi* – <u>sim</u>ulates the <u>q</u>uantities of <u>i</u>nterest
  - Automates the simulation of quantities of interest for the X values you just set.
  - e.g.: *simqi, prval(1)*
  - e.g.: *simqi, fd(prval(1)) changex(x4 p25 p75)*

You Can Use Clarify, but you Don’t have to.

Models Currently Supported by Clarify

regress	mlogit
logit	poisson
probit	nbreg
ologit	sureg
oprobit	weibull

But, you really don’t need Clarify to do this, so you can simulate quantities of interest for *any* model!

✓ Easy to simulate parameters because Stata saves them after estimation!

✓ <u>Program</u> the correct link function yourself!

An Example:
The Censored Probit Model

Selection Equation:

$$y1_j = z_j g + u_{2j}$$

Outcome Equation:

$$y2_j = x_j b + u_{1j}$$

Where:

$$u_1 \sim N(0,1)$$
$$u_2 \sim N(0,1)$$
$$corr(u_1, u_2) = r$$

/*Programming Step One:
Estimate Model*/

heckprob y2 x1 x2 x3 x4,
sel(y1 = z1 z2 z3 z4)
robust

An Example: The Censored Probit Model

Simulate the model parameters by drawing from the multivariate normal distribution.

Note: there are 11 – 4 Xs, 4 Zs, 2 constants, and ρ (the correlation between the errors).

/*Programming Step Two: Draw $\tilde{b}$ from multivariate normal, mean $\hat{b}$ and Covariance Matrix $\hat{\Sigma}$.*/

matrix params = e(b)

matrix P = e(V)

drawnorm b1-b11, means(params) cov(P) double

An Example:
The Censored Probit Model

Stata estimates the hyperbolic arctangent of ρ, so we must simulate to get the actual ρ.

/*Programming Step Three: Generate Simulated Rho*/

gen simrho = (exp(2*b11)-1)/(exp(2*b11)+1)

An Example:
The Censored Probit Model

Initiate a looping structure to generate *m* (in this case 1,000) simulated first differences for the effect of x1 on y2 comparing when x1 is at its mean (the base model) to when x1 is at a value two standard deviations (denoted _m2sd) below its mean.

```
/*ProgramStep Four: The Loop*/
local i =1
 /*A. Generate variables that will be used to fill
in a cell of Substantive Table*/
generate base_y2=.
generate x1_m2sd=.
while `i' <=1000 {
 /*B. Generate zγ  for the selection equation.*/
quietly generate select = b6[`i'] +
(b7[`i']*z1) + (b8[`i']*z2) +
(b9[`i']*z3) + (b10[`i']*z4)
 /*C.  Generate xb for the outcome equation.*/
quietly generate outcome =
b1[`i'] + (b2[`i']*x1) +
(b3[`i']*x2) + (b4[`i']*x3) +
(b5[`i']*x4)
```

An Example: The Censored Probit Model

This is the meat of the simulation. The first three commands generate the probability of being selected and experiencing the outcome (p_11) for the base model. For the censored probit, this probability is (Greene 2000, 857):

$$\Phi_2[b'x, g'z, r]$$

```
/* Generate first difference*/
quietly generate p_11 =
 binorm(outcome,select,simrho)
quietly summarize p_11, meanonly
quietly replace base_y2=r(mean) in `i'
quietly generate x1_m2sd=outcome -
 (b1[`i']*x1) + (b1[`i']*-0.2)
quietly generate p11_x1_m2sd
 =binorm(x1_m2sd,select,simrho)
quietly summarize p_x1_m2sd,
 meanonly
quietly replace x1_m2sd=r(mean) in `i'
```

An Example:
The Censored Probit Model

To get the other 999 we drop the three variables we just generated and repeat the loop until `i’ = 1,000.

When we’re done with the 1,000 simulations, we can use the centile command to get the relevant distributions. To do this for each variable in the model, we would embed this loop within a larger looping structure.

```
/*Step 5: Do the Loop Again*/
drop select outcome p_11
    x1_m2sd  p11_x1_m2sd
disp `i'
local i=`i'+1
}
centile base_y2 x1_m2sd,
    centile(2.5 50 97.5)
```