Notes on Simulation[1]

1 Uncertainty

Remember that there are two types of uncertainty we deal with when trying to make inferences from our sample to the population:

1. **Estimation Uncertainty:** This is our uncertainty about what the true parameters of the model are. We can think of it as being caused by small samples. In other words, if N were infinity (not the entire sample, but infinity), then there would be no estimation uncertainty.
2. **Fundamental Uncertainty:** This is the stochastic component of the real world - our disturbance ϵ. This means that our measure of fundamental uncertainty is σ^2, the variance of ϵ.

Clarify: Essentially this is a tool for (i) computing quantities of interest and (ii) for computing confidence intervals around those quantities.

You should have already learned how to calculate these quantities for some basic models in your earlier classes. For example, you probably learned to calculate confidence intervals around β or around the predicted value of Y. How did you do this?

2 Calculating 95% Confidence Intervals Around β

Let's start with the following model:

$$Y = \beta_0 + \beta_1 X_1 + \epsilon \tag{1}$$

We might want to calculate a confidence interval around β.

We know from the Gauss-Markov assumptions that

$$\hat{\beta} \sim \mathcal{N}(\beta, \text{var}(\hat{\beta})) \tag{2}$$

STOP: What does it mean to say that $\hat{\beta}$ is distributed in some way? Don't we only have one observation of $\hat{\beta}$? Implicitly there are millions of samples of the data that we could have drawn from the population - we happened to get our particular sample. However, if we had drawn a different set of observations, we would have got a different set of values for $\hat{\beta}$.[2] What Eq. 2 is saying is that

[1] These notes are based on notes by Jonathan Nagler.

[2] It is important to remember that our estimate of $\hat{\beta}$ is an unbiased estimate of β, but it will not necessarily equal β. We hope that if the sample we are using is somehow 'typical' that our estimate will be 'near' the population value. Unfortunately, it is always possible that we could obtain in any given sample an estimate $\hat{\beta}$ far from β, and we can never know for sure. The point here is to start thinking in terms of the sampling distribution of $\hat{\beta}$. Recall that the sampling distribution of a statistic can be thought of as the theoretical distribution of some statistic that we would obtain through repeated sampling.

the βs that we estimate from all of the possible sets of observations (samples) that we could draw from the population follow a multivariate normal distribution with a mean of β and a variance of $\text{var}(\hat{\beta})$. As we'll see, it is this that is at the basis of *Clarify* and other forms of simulation.

So, how do we calculate (large sample) confidence intervals? One method you are probably familiar with is the *pivotal method*. We can use the sample value $\hat{\beta}$ as the "center" or "pivot" of our confidence interval. We must choose a level of confidence – tradition suggests that we set $1-\alpha = 0.95$ (a "95 percent level of confidence"), though there's nothing special about this number. This means that we want to create a confidence interval such that

$$\Pr(\hat{\beta}_L \leq \beta \leq \hat{\beta}_U) = 0.95$$

One way of calculating the bounds of the confidence interval is to choose $\hat{\beta}_L$ and $\hat{\beta}_U$ so that

$$\Pr(\beta < \hat{\beta}_L) = \int_{-\infty}^{\hat{\beta}_L} \phi_{\hat{\beta}}(u)\, du = 0.025$$

and

$$\Pr(\beta > \hat{\beta}_H) = \int_{\hat{\beta}_H}^{\infty} \phi_{\hat{\beta}}(u)\, du = 0.025.$$

Since we know the parameters of $\phi_{\hat{\beta}}$ – that is, the distribution is $\mathcal{N}(\beta, \text{var}(\hat{\beta})$ – calculating values for the upper and lower limits of the confidence interval is straightforward. In effect, we can make the following probabilistic statement about β.

$$\Pr(\beta - 1.96\sigma_{\hat{\beta}} < \hat{\beta} < \beta + 1.96\sigma_{\hat{\beta}}) = 0.95 \tag{3}$$

where $\sigma_{\hat{\beta}}$ is the standard error of $\hat{\beta}$. If we subtract β and $\hat{\beta}$ from all sides of the inequality, we have

$$\Pr(-\hat{\beta} - 1.96\sigma_{\hat{\beta}} < -\beta < -\hat{\beta} + 1.96\sigma_{\hat{\beta}}) = 0.95 \tag{4}$$

Now multiplying by -1 (which flips the inequalities) and rearranging terms, we have the familiar equation giving us the 95% confidence intervals around β.[3]

$$\Pr(\hat{\beta} - 1.96\sigma_{\hat{\beta}} < \beta < \hat{\beta} + 1.96\sigma_{\hat{\beta}}) = 0.95 \tag{5}$$

[3]Just as a reminder, Equation (5) does NOT mean that there is a 95% chance that the true population β is in this interval. The population parameter is a fixed constant and is either in the confidence interval or it is not. In other words, once our sample has been observed, β is either in it or it isn't - it is not something that is probabilistic at this point. The correct interpretation is that over a large number of repeated samples, approximately 95% of all intervals constructed in this way will include β, the true population parameter. This is why we sometimes say that we are "95% confident" that the interval contains β.

3 Calculating 95% Confidence Intervals Around $E[Y]$

A similar process can be used to calculate confidence intervals around the expected value of Y. We know that

$$\hat{Y} \sim N(E[Y], \text{var}(\hat{Y})) \tag{6}$$

As a result, we can make a probabilistic statement about $E[Y]$.

$$\Pr(E[Y] - 1.96\sigma_{\hat{Y}} < \hat{Y} < E[Y] + 1.96\sigma_{\hat{Y}}) = 0.95 \tag{7}$$

where $\sigma_{\hat{Y}}$ is the standard error of $\hat{Y}$.[4] If we subtract $\hat{Y}$ and $E[Y]$ from all sides of the inequality, we have

$$\Pr(-\hat{Y} - 1.96\sigma_{\hat{Y}} < -E[Y] < -\hat{Y} + 1.96\sigma_{\hat{Y}}) = 0.95 \tag{11}$$

Now multiplying by -1 (which flips the inequalities) and rearranging terms, we have the equation giving us the 95% confidence intervals around $E[Y]$.

$$\Pr(\hat{Y} - 1.96\sigma_{\hat{Y}} < E[Y] < \hat{Y} + 1.96\sigma_{\hat{Y}}) = 0.95 \tag{12}$$

4 Calculating Large Sample Confidence Intervals

In general, the rules for calculating large sample confidence intervals around some statistic $\hat{\theta}$ are the following:

1. Select your level of confidence $1 - \alpha$.
2. Calculate the sample statistic $\hat{\theta}$.
3. Calculate the z-value associated with the $1 - \alpha$ level of confidence.
4. Multiply that z-value by $\sigma_{\hat{\theta}}$, the standard error of the sampling statistic.
5. Construct the confidence interval according to $[\hat{\theta}_L, \hat{\theta}_U] = \left[\hat{\theta} - z_{\alpha/2}\sigma_{\hat{\theta}},\ \hat{\theta} + z_{\alpha/2}\sigma_{\hat{\theta}}\right]$.

[4] We know that

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 \tag{8}$$

This means that

$$\text{var}(\hat{Y}) = \text{var}(\hat{\beta}_0) + X^2\text{var}(\hat{\beta}_1) + 2X\text{cov}(\hat{\beta}_0, \hat{\beta}_1) \tag{9}$$

It follows form this that

$$\sigma_{\hat{Y}} = \sqrt{\text{var}(\hat{\beta}_0) + X^2\text{var}(\hat{\beta}_1) + 2X\text{cov}(\hat{\beta}_0, \hat{\beta}_1)} \tag{10}$$

5 Simulation and *Clarify*

What we have just seen is that we can calculate quantities of interest and confidence intervals around these quantities for certain basic models without the need for things like *Clarify*. However, in cases that are more complicated than simple linear models such as OLS, we may need to use simulation methods. This is essentially what *Clarify* does.

Say we were interested in a probit model. We start with the following basic model

$$Y_i^* = X_i\beta + \epsilon \tag{13}$$

where

$$Y_i = \begin{cases} 1 & \text{if } Y_i^* > 0 \\ 0 & \text{if } Y_i^* \leq 0 \end{cases}$$

We might be interested in several quantities of interest from a probit model. We know that

$$\Pr(Y_i = 1) = \Phi(X_i\beta) \tag{14}$$

where Φ is the cdf of the normal distribution and $X_i\beta$ is just our linear model i.e. $\beta_0 + \beta_1 X_1 + \beta_2 X_2 \ldots$. We might want to know the predicted probability of $Y_i = 1$ for a given value of X and compute a confidence interval around this. We can do this through simulation.

5.1 The Basics of Simulation

The following example assumes that we want some quantity of interest involving X and $\hat{\beta}$. We'll look at a specific example in a moment.

- Step 1: Draw M values of $\hat{\beta}$.
  1. Estimate your model
     This will involve estimating the $k \times 1$ vector of $\hat{\beta}$ coefficients and a $k \times k$ variance-covariance matrix $\text{var}(\hat{\beta})$.
  2. Draw a value of $\hat{\beta}$ from the distribution $N(\hat{\beta}, \text{var}(\hat{\beta}))$
     Note that the coefficients from ML estimators will always be multivariate normal.
  3. Repeat (2) M times, where M is normally about a 1,000.
     M is the number of draws.
- Step 2: Choose the values of X that you are interested in.
- Step 3: Compute the M values of the quantity of interest that you want (in this case $\Pr(Y_i = 1)$) by combining each value of $\hat{\beta}_m$ with X. Then take the normal of this sum. This gives us M values of our quantity of interest - call it P_m.
- Step 4: Look at the distribution of P_m to determine our confidence intervals.

This might sound rather complicated but it really isn't.

5.2 An Example in *Stata*

Say our underlying linear model was the following:

$$Y^* = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \tag{15}$$

In other words, we have two independent variables and a constant.

- Step 1: Draw M values of $\hat{\beta}$.

  1. Estimate your model

     · probit y X_1 X_2
     · preserve

  2. Draw M values of $\hat{\beta}$ from the distribution $N(\hat{\beta}, \text{var}(\hat{\beta}))$

     · set seed 10101
     · drawnorm $MG\_b1 - MG\_b3$, n(1000) means(e(b)) cov(e(V)) clear
     · save simulated_betas, replace
     · restore
     · merge using simulated_betas
     · summarize _merge
     · drop _merge
     · summarize

- Step 2: Choose the values of X that you are interested in.

  · scalar h_$X_1 = 2$
  · scalar h_$X_2 = 2.5$
  · scalar h_constant= 1

- Step 3: Compute the M values of the quantity of interest that you want (in this case $\Pr(Y_i = 1)$) by combining each value of $\hat{\beta}_m$ with X. Then take the normal of this sum. This gives us M values of our quantity of interest - call it P_m.

  · generate x_betahat1 = MG_b1*h_X_1+ MG_b2*h_X_2 + MG_b3*h_constant
  · generate prob_hat1 = normal(x_betahat1)[5]

[5]To change the code for logit, you would obviously run a logit model at the beginning. At this point of the code, you would type: generate prob_hat1 =1/(1+exp(-(x_betahat1))).

- Step 4: Look at the distribution of P_m to determine our confidence intervals.

  · sum prob_hat1
  · centile prob_hat1, centile(2.5 97.5)

5.3 Another Example in *Stata*

Say we had the same probit model, but now we wanted to know how the predicted probability of $Y_i = 1$ changes if we increase X_1 by one unit. The code would look like that shown below.

· probit y X_1 X_2
· preserve
· set seed 10101
· drawnorm $MG\_b1 - MG\_b3$, n(1000) means(e(b)) cov(e(V)) clear
· save simulated_betas, replace
· restore
· merge using simulated_betas
· summarize _merge
· drop _merge
· summarize
· scalar h_X_1 = 2
· scalar h_X_2 = 2.5
· scalar h_constant= 1
· scalar h_X_{12} = 3
· generate x_betahat1 = MG_b1*h_X_1+ MG_b2*h_X_2 + MG_b3*h_constant
· generate prob_hat1 = normal(x_betahat1)
· generate x_betahat2 = MG_b1*h_X_{12}+ MG_b2*h_X_2 + MG_b3*h_constant
· generate prob_hat2 = normal(x_betahat2)
· generate prob_diff = prob_hat2-prob_hat1
· sum prob_hat1 prob_hat2 prob_diff
· centile prob_hat1 prob_hat1 prob_diff, centile(2.5 97.5)

5.4 *Clarify*

All *Clarify* does is automate some of this underlying code so that you do not have to write it.[6]

For example, to produce the predicted probability and confidence intervals that we calculated in the first example above, all we would type using *Clarify* is:

· estsimp probit X_1 X_2
· setx X_1 2 X_2 2.5
· setx

[6]To get *Clarify*, open *Stata* and type: net from http://gking.harvard.edu/clarify.

· simqi

In order to produce the change in predicted probability and confidence intervals that we calculated in the second example above, all we would type using *Clarify* is:

· estsimp probit X_1 X_2
· setx X_1 2 X_2 2.5
· setx
· simqi, fd(pr) changex(X_1 2 3) level(95)

While *Clarify* does simplify matters, I want you to be very comfortable with the underlying code since *Clarify* does not do everything.

5.5 Simulation with Interaction Terms

```
*     ************************************************************  *;
*                          Estimate Probit Model                    *;
*      Y* = b0 + b1X + b2Z + b3XZ + b4Control1 + b5Control2 + epsilon   *;
*     ************************************************************  *;

probit Y   X   Z   XZ  Control1 Control2;

*     ************************************************************  *;
*       Take 10,000 draws from the estimated coefficient vector and    *;
*       variance-covariance matrix.                                    *;
*     ************************************************************  *;

preserve;
set seed 10101;
drawnorm MG_b1-MG_b6, n(10000) means(e(b)) cov(e(V)) clear;

*     ************************************************************  *;
*       To calculate the desired quantities of interest we need to set  *;
*       up a loop.  This is what we do here. First, specify what       *;
*       quantities should be saved and what these quantities should be  *;
*       called.                                                        *;
*     ************************************************************  *;

postutil clear;
postfile mypost prob_hat0 lo0 hi0 prob_hat1 lo1
        hi1 diff_hat diff_lo diff_hi using sim , replace;
noisily display "start";
```

```
*     ************************************************************  *;
*       Start loop.  Let ‘a’ be the modifying variable Z and let this   *;
*       run from min to max in the desired increments.                  *;
*     ************************************************************  *;

local a=0 ; while ‘a’ <= 35 { ;

    {;

scalar h_X=5;

scalar h_Control1=2;

scalar h_Control2=3.4;

scalar h_constant=1;

    generate x_betahat0 = MG_b1*h_X
                            + MG_b2*(‘a’)
                            + MG_b3*h_X*(‘a’)
                            + MG_b4*h_Control1
                            + MG_b5*h_Control2
                            + MG_b6*h_constant;


    generate x_betahat1 = MG_b1*(h_X+1)
                            + MG_b2*‘a’
                            + MG_b3*(h_X+1)*(‘a’)
                            + MG_b4*h_Control1
                            + MG_b5*h_Control2
                            + MG_b6*h_constant;

    gen prob0=normal(x_betahat0);
    gen prob1=normal(x_betahat1);
    gen diff=prob1-prob0;

    egen probhat0=mean(prob0);
    egen probhat1=mean(prob1);
    egen diffhat=mean(diff);

    tempname prob_hat0 lo0 hi0 prob_hat1 lo1 hi1 diff_hat diff_lo diff_hi;

    _pctile prob0, p(2.5,97.5);
    scalar ‘lo0’ = r(r1);
    scalar ‘hi0’ = r(r2);
```

```
    _pctile prob1, p(2.5,97.5);
    scalar ‘lo1’= r(r1);
    scalar ‘hi1’= r(r2);

    _pctile diff, p(2.5,97.5);
    scalar ‘diff_lo’= r(r1);
    scalar ‘diff_hi’= r(r2);

    scalar ‘prob_hat0’=probhat0;
    scalar ‘prob_hat1’=probhat1;
    scalar ‘diff_hat’=diffhat;

    post mypost (‘prob_hat0’) (‘lo0’) (‘hi0’) (‘prob_hat1’) (‘lo1’) (‘hi1’)
                (‘diff_hat’) (‘diff_lo’) (‘diff_hi’) ;
    };
    drop x_betahat0 x_betahat1 prob0 prob1 diff probhat0 probhat1 diffhat ;
    local a=‘a’+ 1 ;
    display "." _c;
    } ;

display "";

postclose mypost;

*     ************************************************************  *;
*       Call on posted quantities of interest and graph effect of       *;
*       changing X from 5 to 6 on Pr(Y=1) at different values of Z when *;
*       the control variables are set to specific values.               *;
*     ************************************************************  *;

use sim, clear;

gen MV = _n-1;

graph twoway line diff_hat MV, clwidth(medium) clcolor(black)
        ||   line diff_lo  MV, clpattern(dash) clwidth(thin) clcolor(black)
        ||   line diff_hi  MV, clpattern(dash) clwidth(thin) clcolor(black)
        ||  ,
            xlabel(0 10 20 30, labsize(3))
            ylabel(-.1 0 0.1 0.2, labsize(3))
            yscale(noline)
            xscale(noline)
            yline(0) legend(off)
            scheme(s2mono) graphregion(fcolor(white));
```

```
*     ************************************************************  *;
*                 Figure can be saved in a variety of formats.          *;
*     ************************************************************  *;

graph export  h:\figure1.eps, replace;

translate @Graph h:\figure1.wmf;
```