Simulation and Substantive Interpretation in Statistical Modeling

By Brandon Bartels & Kevin Sweeney

Discussion of the substantive *impact* of a variable on a dependent variable, especially for maximum likelihood models, requires more than reporting the sign and significance of coefficient. Substantive interpretation of MLE models is now practically required for publication in major journals. Post-estimation simulation of the model's parameters allows the analyst to calculate these substantive quantities of interest, and most importantly, *it allows the analyst to reflect the degree of uncertainty around those quantities*.

We present the logic of post-estimation simulation and express its importance for presenting both the estimate *and* the precision of a substantive quantity of interest. CLARIFY (hereafter) (Tomz, Wittenberg, and King 2003; see also King, Tomz, and Wittenberg 2000) is an easy-to-use post-estimation program that simulates parameters and calculates substantive quantities of interest and the degree of uncertainty around those estimates. can currently be used for linear regression, logit and probit, ordered logit and probit, multinomial logit, count models (Poisson and negative binomial), duration models (Weibull), and seemingly unrelated regressions. Importantly, we emphasize that analysts can move beyond to simulate parameters of interest from a model, and that researchers estimating models outside of the canned models should almost always execute post-estimation parameter simulation.

The Logic of Post-Estimation Parameter Simulation Using Clarify

The motivation for post-estimation parameter simulation is provided by the Monte Carlo principle: We can learn about the distributional properties of a random variable, *y*, by sampling many (*m*) times from the probability distribution that generated *y*. translates this principle to the parameters of a statistical model. Since maximum likelihood parameters have the statistical property of asymptotic normality, we can learn more about a coefficient by drawing *m* samples from a normal distribution underlying the coefficient. Post-estimation simulation used in , then, involves simulating a *distribution* of each parameter estimated by the model; think of the technique as attempting to simulate the *sampling distribution* of a parameter. uses Monte Carlo simulation to draw *m* values of each parameter from a multivariate normal distribution, where the mean of the distribution is the vector of point estimates of the coefficients from the model, and the variance is the variance-covariance matrix of those point estimates.[1]

 uses these simulated parameter estimates to generate quantities of interest and most importantly, to reflect uncertainty in those quantities. For example, consider a simple logit model of intended vote choice, where the dependent variable is intention to vote for Bush or not. We want to estimate the effect of economic perceptions—measured as *very bad*, *average*, or *very good*—on the intention of a Bush vote. First, we may be interested in the probability of an intended Bush vote for someone who perceives the economy as *very bad*, holding other variables constant at a baseline value (e.g., the mean). We first estimate the model, and using , we randomly draw *m*, e.g., 1000, values of each coefficient using the procedure discussed above. To generate the point prediction of an intended Bush vote when economic perceptions is set at *very bad* and the rest of the variables are held constant at their baselines, first estimates 1000 probability estimates via the simulated parameter estimates. The estimated point prediction reports is the *mean* of these 1000 probability estimates. Let's say that the mean is 0.15, so we can report that, all else equal, the estimated probability of voting for Bush for someone who perceives

[1] Note that the Clarify procedure differs from bootstrapping in that it is parametric, while bootstrapping is nonparametric. For a further discussion, see King et al. (2000, 352).

the economy as very bad is 0.15. Importantly, as we discussed above, parameter simulation allows one to report the *precision* of this quantity of interest. To report a 95% confidence interval of this point prediction, C simply sorts the 1000 simulated probability estimates from lowest to highest, and reports the 25th and 975th probability estimates as the lower and upper bounds of this confidence interval. Let's say that these upper and lower bounds are 0.09 and 0.21, respectively. We can now report that with 95% confidence, the probability of voting for Bush for one who perceives the economy as very bad, all else equal, is between 0.09 and 0.21.

We may also want to report an estimate and associated precision of a first difference, i.e., the change in the probability of voting for Bush given a change in economic perceptions. Say we wanted to estimate the change in the probability of an intended Bush vote as economic perceptions change from *average* to *very good*. C estimates *m* of these first differences, and then reports the mean and the 95% confidence interval of the estimate. Assume that the estimated first difference is 0.35. This information alone would lead us to conclude that a change in economic perceptions from *average* to *very good* increases the probability of an intended Bush vote by 0.35. Now, if the lower and upper bounds of the confidence interval are 0.25 and 0.45, respectively, we can conclude with 95% confidence that this first difference is 0.35, plus or minus about 0.10. Reporting this confidence interval is important because it conveys the degree of dispersion around the first difference and also allows one to conclude whether the first difference is statistically different from zero.

Simulation for Substance without Using Clarify

C is useful because it allows analysts to implement the simulation technique in a simple manner. However, the more important point behind the program is that simulation is a powerful technique to gain substantive leverage over our statistical results for *any* type of statistical model. The coefficients and their attendant directionality and statistical significance contain relatively little interesting information (especially in maximum likelihood models), and "substantive" results reported with the coefficients alone will hide the uncertainty that surrounds our point estimates. This holds for *all types* of statistical models, not just those canned in C. We might call this critical underlying point "simulation for substance."

A review of the recent (i.e., post King, Tomz, and Wittenberg 2000) literature in the major journals in our field shows that scholars have overwhelmingly failed to extend the notion of "simulation for substance" to statistical models that are not canned in C. This is a major problem not only because their results are not presented as informatively as they might have been and therefore the import of the piece is not capitalized on, but also because taking the additional step is relatively easy and *does not* require the C program. To show this here we will consider a member of the increasingly popular sample selection model family: censored probit.[2] Simulation for this model is not canned in C, but can easily be programmed by the analyst. The left column (on the next page) walks through the model and relevant simulation, the right column displays Stata code for achieving "simulation for substance" in the censored probit model.

In conclusion, since post-estimation parameter simulation maximizes the quality of the presentation of results from a statistical model, analysts should be encouraged to use this technique whenever possible,

[2] Space limitations preclude us from explaining sample selection models in more detail, but we chose this example because, as a two stage model, it may appear very complicated to simulate quantities of interest. It is not. You will likely need to be somewhat familiar in order to follow what is below. To bone up on your own: Heckman (1979) is the most cited reference for sample selection models, Dubin and Rivers (1989) extend the Heckman model to cases where the dependent variable in the outcome equation is binary (i.e. censored probit), and Timpone (2002) and Sweeney and Fritz (2004) provide examples of "simulation for substance" applied to the censored probit model.

even for models outside of C. Building on last quarter's session on post-estimation, PRISM will offer a session this quarter on advanced programming, which will cover the topic of post-estimation parameter simulation for models not canned in C. See below for more information on this session.

References

Dubin, Jeffery, and Douglas Rivers. 1989. "Selection Bias in Linear Regression, Logit, and Probit Models." *Sociological Methods and Research* 18:360-90.

Greene, William H. 2000. *Econometric Analysis*. 4th Edition. Upper Saddle River, New Jersey: Prentice Hall.

Heckman, James J. 1979. "Sample Selection Bias as a Specification Error." *Econometrica* 47:153-61.

King, Gary; Michael Tomz; and Jason Wittenberg. 2000. "Making the Most of Statistical Analyses: Improving Interpretation and Presentation." *American Journal of Political Science* 44: 341-355.

Sweeney, Kevin, and Paul Fritz. 2004. "Jumping on the Bandwagon: An Interest Based Explanation for Great Power Alliances." *Journal of Politics* 66(2):428-49.

Timpone, Richard J. 2002. "Estimating Aggregate Policy Reform Effects: New Baselines for Registration, Participation, and Representation." *Political Analysis* 10:154-77.

Tomz, Michael, Jason Wittenberg, and Gary King. 2003. *CLARIFY: Software for Interpreting and Presenting Statistical Results.* Version 2.1. Cambridge, MA: Harvard University. http://gking.harvard.edu

1. Selection Equation:

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

Outcome Equation:

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

Where:

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

2. 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).

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

4. 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.

D. 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):

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

Then, we do the same for the case where X1 is two standard deviations below its mean (x1_m2sd).

This gives us 1 simulated first difference.

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

6. 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 One: Estimate Model*/ [3]

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

/*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

/*Step Three: Generate Simulated Rho*/

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

/*Step 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 zg 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)

/*D. Generate the relevant 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'

/*Step Five: Do the Loop Again*/

>drop select outcome p_11 x1_m2sd p11_x1_m2sd
>disp `i'
>local i=`i'+1
>}

/*Step Six: Get the Median and Confidence Intervals for your first difference*/

>centile base_y2 x1_m2sd, centile(2.5 50 97.5)

[3] Stata commands are preceded by >.