Understanding The STATA Code for Plots With A Single Modifying Variable and a Limited Dependent Variable

Below, I go through the STATA code for creating a plot that illustrates the marginal effect of some variable in an interaction model when there is a single modifying variable and the dependent variable is limited in some sense. The example that follows is based on a probit model and so we are interested in the marginal effect of X on the probability of Y as the modifying variable Z changes. Specific lines of code are written in small font, whereas comments are written in larger font.

```
version 8.0
log using logfilename.log, replace
#delimit;
```

The first line of code says that we are using STATA 8.0. The second line opens a log file called 'logfilename.log' or whatever you want to call it. The third line simply tells STATA that all command lines are finished with a semi-colon.

```
* ****************************************************************** *;
* ****************************************************************** *;
* File-Name: limited.do                                               *;
* Date: Month/Day/Year                                                *;
* Author: MRG                                                         *;
* Purpose:Do-file for creating figures showing the                    *;
*         marginal effect of X on Y with a single                     *;
*         modifying variable where the dependent variable             *;
*         is limited. The estimated model is:                         *;
*         Y* = b0 + b1X + b2Z + b3XZ + epsilon                        *;
* Input File: name_of_datafile.dta                                    *;
* Output File: logfilename.log                                        *;
* Data Output: None                                                   *;
* Previous file: None                                                 *;
* Machine: Office                                                     *;
* ****************************************************************** *;
* ****************************************************************** *;
```

This part of the code is not required for creating the figures. I use it so that I can keep track of what I am doing. I keep track of the do-file name, when I last edited it, who wrote it, why I wrote it, what dataset the do-file uses, what the logfile is called, what any new dataset created by the do-file is called, what the names of any previous do-files were called, and what machine I last ran the do-file on.

```
use h: name_of_datafile.dta;
```

This tells STATA the path for finding the dataset used by this do-file.

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

probit Y X Z XZ Control1 Control2;
```

This is simply the estimation command. You replace the variable names Y, X, Z etc. with the names of your variables. 'Control1' and 'Control2' simply refer to other independent variables that

you might have that are not involved in the interaction. You should enter the independent variables in this order. In other words, the first independent variable should be the variable that you want to calculate the marginal effect of. The second independent variable should be the modifying variable. The third independent variable should be the interaction term.

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

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

The first line of this code says preserve the data that STATA already has in its memory. The second says that we want to take 10,000 draws from the estimated coefficient and variance-covariance matrix from the probit model. This will help us calculate measures of uncertainty a little later. Essentially we will get 10,000 β_1s, β_2s, etc. This is essentially what the CLARIFY procedure does. You can determine how accurate you want your results to be by changing how many draws you take. I think the default in CLARIFY is 1,000. In this example, we are using 10,000.

```
* ************************************************************** *;
* To calculate the desired quantities of interest we need to set *;
* up a loop. This is what we do here.                            *;
* ************************************************************** *;
* First, specify what you 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";
```

To calculate the desired quantities of interest it will be necessary to utilize a loop. We will have to calculate and store various quantities at each stage of the loop. We begin to specify what quantities we will store in these lines of code, what the quantities will be called, and where they will be stored. The first line basically says that if you have 'posted' or stored anything earlier, clear it. This is a safety step because you don't want to mix up quantities that you have posted from different procedures. If you are not worried about doing this, you can drop this first line. The next line of code basically tells STATA what the quantities that you are going to post should be called. I am going to post a set of quantities called 'prob_hat0', 'lo0', and 'hi0' – 'prob_hat0' is the probability of Y under scenario 0, 'lo0' is the lower limit of the 95% confidence interval for this probability, 'hi0' is the upper limit of the 95% confidence interval for this probability. I am also going to post similar quantities ('prob_hat0' 'lo0' hi0') for scenario 1. I will explain what the scenarios are in a moment. Finally, I will also be posting the change in the probability of Y as we move from scenario 0 to scenario 1. This is some kind of difference and is called 'diff_hat'. I also post the lower and upper bounds of the 95% confidence interval around this difference ('diff_lo' and 'diff_hi'). I am going to post these quantities to a dataset call 'sim' - because it is simulated data. Finally, the last line simply tells STATA to print on the screen 'start' when it has started the loop procedure. This last line is not really necessary but lets you know that STATA has actually started to work on the loop.

```
* ************************************************************ *;
* 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 { ;
```

Let 'a' be the modifying variable Z. The loop is going to run from when 'a' or the modifying variable Z is 0 until when 'a' (or Z) is 35. How do you determine what the range should be? This is the choice of the analyst. You might want to run the loop from the minimum observed value of Z to the maximum observed value of Z. In this example, 0 would be the minimum of Z and 35 would be the maximum. If you have one or two outliers you might want to choose a range that covers the majority of the observations of Z. This is totally a choice of the analyst. Simply plug in the desired numbers.

```
{;

scalar h_X=5;
scalar h_Control1=2;
scalar h_Control2=3.4;
scalar h_constant=1;
```

In limited dependent variables you can only look at the effect of some variable X when the other independent variables are set at specific levels. This is because the effect of X on Y depends on the values of the other independent variables. So, these lines of code set the values of the independent variables in the loop. What values should you choose? You might want to choose the mean for continuous variables or the median for dichotomous variables. You could choose values for the independent variables that satisfy some scenario of interest. These are all choices of the analyst. Note that you also have to specify the value of X even though this is the variable that you are interested in capturing the effect of. You should just plug in your values of interest. You also need to specify the value of the constant - this will always be 1 as you would expect.

```
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;
```

I would like to calculate the probability of Y when my variable X is at some level when the modifying variable 'a'=0. I would then like to calculate the probability of Y if I increase my variable X by 1 when the modifying variable 'a'=0. I would then like to calculate the change or difference in these probabilities. In probit models, the probability that Y=1 is $\Phi(x\hat{\beta})$. Thus, we first need to calculate $x\hat{\beta}$ for scenario 0 for the case when 'a'=0 and X=5 - this is x_betahat0. Second, we need to calculate $x\hat{\beta}$ for the second scenario for the case when 'a'=0 and X is increased by one unit to 6 - this is x_betahat1. Why do I increase X by 1? This is a decision that the analyst has to make. What is a meaningful increase in X? This might be a one unit increase as in this example. But you might want to look at a standard deviation increase as well. If X runs from 0 to 1 it might make little sense to look at a one unit increase because this is capturing a shift from its minimum to its

maximum - this is often substantively unrealistic. If X runs from 0 to 1, you might want to look at the effect of a 0.1 or 0.01 increase in X. It all depends on what is substantively meaningful in your particular case. It is important to remember when it comes to interpretation, though, that your baseline category is for a particular value of X. In this case, I chose X to be 5. The analyst has to decide what a substantively meaningful baseline category for X would be. Thus, you would interpret this particular example as the effect of a one unit increase in X from its mean (if its mean is 5) on the probability of Y for the case when 'a'=0.

```
gen prob0=norm(x_betahat0);
gen prob1=norm(x_betahat1);
gen diff=prob1-prob0;
```

So far, I have only calculated x_betahat0 and x_betahat1. I need to transform these into probabilities by taking the normal density of these quantities. This is what the first two lines of code do here. The third line then calculates the change in probability from scenario 0 to scenario 1.

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

Because I drew 10,000 values of $\hat{\beta}$, I have 10,000 prob0's, prob1's and diff's. Our best guess of the actual value of prob0, prob1, and diff is the mean of these 10,000 values. These three lines of code generate variables probhat0, probhat1, and diffhat that are equal to the mean value of prob0, prob1, and diff.

```
tempname prob_hat0 lo0 hi0 prob_hat1 lo1 hi1 diff_hat diff_lo diff_hi ;
```

This line of code simply tells STATA that I am going to create some other quantities of interest and they are called prob_hat0 lo0 hi0 prob_hat1 lo1 hi1 diff_hat diff_lo diff_hi. These are the quantities that I am actually going to post to 'sim.dta' from every loop.

```
_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);
```

So far, I have calculated the probability of Y for scenario 0, the probability of Y for scenario 1, and the difference between these probabilities. However, I have not calculated any measure of uncertainty around these probabilities and first difference. These lines of code calculate measures of uncertainty. Specifically, they calculate the lower and upper bound of the 95% confidence intervals for each of the three quantities of interest that I have already calculated. Note that if you want a different confidence interval you simply have to change 'p(2.5,97.5)' to something else. For example, a 90% confidence interval would be 'p(5,95)'. Again this all works because we have 10,000 values for each of the probabilities and first difference. 'p(2.5,97.5)' says take the value of the 2.5 percentile and call it the lower bound of the 95% confidence interval and take the value of the 97.5 percentile and call it the upper bound of the 95% confidence interval. 95% of the observations fall between these two numbers.

```
scalar 'prob_hat0'=probhat0;
scalar 'prob_hat1'=probhat1;
scalar 'diff_hat'=diffhat;
```

Before, I created variables that captured the probability of Y in scenario 0 (probhat0) and scenario 1 (probhat1). I also created a variable that was the difference between these probabilities (diffhat). The trouble is at the moment that these variables each have 10,000 observations - all of which are the same. I only want to save or post one value for each of the probabilities and the first difference for each stage of the loop. These lines of code create scalars that are equal to the probabilities and the first difference. It is these scalars that I post.

```
post mypost ('prob_hat0') ('lo0') ('hi0') ('prob_hat1') ('lo1') ('hi1') ('diff_hat') ('diff_lo') ('diff_hi');

};
```

Now that I have created all the quantities of interest for when the modifying variable 'a'=0, I post them to 'sim.dta' using this line of code.

```
drop x_betahat0 x_betahat1 prob0 prob1 diffprobhat0 probhat1 diffhat;
```

Once I have posted these values, I drop all calculated quantities of interest from the dataset that I am working with so that I can start again for the case when the modifying variable 'a' is different.

```
local a='a'+ 1 ;
```

This line increases the modifying variable 'a' by one unit and repeats the process that we have just gone through. Why increase 'a' by one unit? Well, you don't have to. Again, it all depends on the choice of the analyst. As I said at the beginning of the loop, this modifying variable Z ranges from a minimum of 0 to a maximum of 35. I decided to calculate the effect of X on the probability of Y across the observed range of Z as Z incrementally increases by one unit. You may want to calculate the effect of X as Z increases by 0.1 or 0.01. It is up to you.

```
display "." _c;

} ;
```

The first line here tells STATA to show a '.' on the screen every time it does one loop. This is just a helpful sign to show you that STATA is still working and hasn't stopped for some unknown reason.

```
display "";
postclose mypost;
```

These lines just say that the loop has finally finished and STATA should stop posting to 'sim.dta'.

```
* ************************************************************ *;
* Call on posted quantities of interest                        *;
* ************************************************************ *;

use sim, clear;
```

Now, call 'sim.dta' which is where you have posted all the quantities of interest. This is now the dataset in STATA.

```
gen MV = _n - 1;
```

This line creates your X-axis and acts as the modifying variable. It is a counter that starts at 0.

```
graph twoway line diff_hat MV, clwidth(medium) clcolor(blue) 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)
                title("Marginal Effect of X on Y As Z Changes", size(4))
                subtitle(" " "Dependent Variable: Y" " ", size(3))
                xtitle(Z, size(3.5) )
                ytitle("Marginal Effect of X", size(3.5))
                xsca(titlegap(2))
                ysca(titlegap(2))
                scheme(s2mono) graphregion(fcolor(white));
```

The desired quantity of interest to be plotted is the change in the probability of Y associated with a one unit increase in X (from its mean, which was 5) across the observed range of the modifying variable Z. The first line plots the change in probability against MV, while the second and third lines plots the upper and lower bounds of the 95% confidence interval about this change in probability. The 'xlabel' and 'ylabel' commands just format the relevant axes. 'yscale(noline)' and 'xscale(noline)' simply tells STATA that I don't want to actually see the x- and y-axes. The 'title' commands are self-explanatory. The remaining lines refer to various formatting issues.

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

graph export h: figure1.eps, replace;

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

There are several ways of exporting the plot so that you can work on it or import it into your document. I tend to use one of these commands. The first command exports the plot as an encapsulated postscript (eps) file. These can be imported directly into any LaTeX document. The second command exports the plot in wmf format. You can insert the plot into a word document simply by inserting it is a picture file.

```
* ************************************************************** *;
* THE END                                                        *;
* ************************************************************** *;

exit;
```

Finally, note that this example is based on a probit model. However, it should be easy to see that it can easily be adapted to take account of other limited dependent variables or non-linear models. All you have to do is calculate the predicted probability for scenario 0 and for scenario 1 using the relevant formula for the model you are using. These formulas can be found in most econometric textbooks. The quantity of interest may not be a predicted probability. For example, you may be interested in the expected duration. Again, just use the formula for the expected duration for

your particular model. Note also that although I have spoken about the marginal effect of X on Y throughout this example, we have not actually calculated the marginal effect of X. The marginal effect of X on Y is the instantaneous change in Y for a small change in X. What we have calculated is really a first difference. The formula for the marginal effect of X in a probit model can easily be found in a textbook if that is the quantity of interest. On the whole, first differences typically seem to be more substantively meaningful in non-linear models, but it depends on your goal.