Detailed Explanation of Stata Code for a "Marginal Effect" Plot for X

Below, I go through the Stata code for creating the equivalent of a marginal effect plot for X from a probit model with an interaction taking the following basic form:[1]

$$\Pr(Y = 1) = \Phi(\beta_0 + \beta_1 X + \beta_2 Z + \beta_3 XZ). \tag{1}$$

```
version 11.0
#delimit ;
log using "C:\matt\publications\jop2\webpage\interaction3.log", replace;
set more off;
```

The first line simply indicates the version of Stata that is being used. The second line tells Stata that all command lines are finished with a semi-colon. The third line opens a log file and indicates where it will be saved. The fourth line tells Stata not to pause when running the do-file.

```
*     ************************************************************ *;
*     ************************************************************ *;
*       File-Name:      interaction3.do                             *;
*       Date:           1/6/2013                                    *;
*       Author:         MG                                          *;
*       Purpose:        Do-file to produce a marginal effect plot for   *;
*                       X when the interaction model is:            *;
*                       Pr(Y=1) = Phi(b0 + b1X + b2Z + b3XZ)        *;
*       Input File:     interaction3.dta                            *;
*       Output File:    interaction3.log                            *;
*       Data Output:    None                                        *;
*       Previous file:  None                                        *;
*       Machine:        laptop                                      *;
*     ************************************************************  *;
*     ************************************************************  *;
```

This part of the code is how I like to start all do-files and is not strictly necessary to create the sample marginal effect plot. It indicates the name of the do-file, the date on which it was written, the author, the purpose of the do-file, the data used with the do-file, the name of the associated log file, the name of any new data set that is created by the do-file, the name of any previous do-file or data set, and where the do-file was run.

```
use "C:\matt\publications\jop2\webpage\interaction3.dta", clear;
```

This tells Stata what data set to open and where to find it. It also clears any existing data in Stata 's memory.

[1] I am using the term "marginal effect plot" loosely here since the code does not actually produce a plot showing the *marginal effect* of X on $\Pr(Y = 1)$. Instead, the plot shows how a discrete change (in this case a *one unit change*) in X influences $\Pr(Y = 1)$.

```
*     ************************************************************  *;
*                           Estimate Probit Model                    *;
*     ************************************************************  *;

xtprobit pec polarization threshold polarization_threshold
        seatshare seatshare_2 incompatibility  asymmetry asym_seat, re i (ident);
```

This line estimates the chosen probit specification. In this case, `pec` is the dependent variable, Y, `polarization` is our X, `threshold` is our Z, `polarization_threshold` is our $X \times Z$, `seatshare` `seatshare_2` `incompatibility` `asymmetry` `asym_seat` are control variables, and `re i (ident)` indicates the use of random effects clustered on `ident`.

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

preserve;

set seed 339487731;

drawnorm SN_b1-SN_b10, n(10000) means(e(b)) cov(e(V)) clear;
```

The first line "preserves" the original data set in the memory. The second line specifies the seed to be used when drawing from the estimated coefficient vector and variance-covariance matrix. Setting the seed allows you to be able to replicate your results in the future. The third line draws 10,000 values of each estimated model parameter from a multivariate normal distribution whose mean and variance is specified by the estimated coefficient vector and variance-covariance matrix from the probit model. `SN_b1-SN_b10` are the labels given to the 10 estimated parameters in our model, i.e., the 10,000 estimates of the first parameter, β_1, are called `SN_b1` ... and the 10,000 estimates of the last parameter, the constant β_0, are called `SN_b10`.

```
*     ************************************************************  *;
*       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 "C:\matt\publications\jop2\webpage\sim.dta", replace;

noisily display "start";
```

To calculate the desired quantities of interest necessary to construct the marginal effect plot, we need to create a loop. These first line closes all open postfiles. The second line specifies the names of all the quantities of interest you are going to calculate and indicates where these quantities will be posted. The third line will get Stata to indicate that your do-file has at least reached this point and is doing something.

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

{;
```

These lines specify the parameters of the loop. We are going to use `a` to capture all the values of Z, `threshold`, for which we want to calculate the effect of X, `polarization`, on $\Pr(Y = 1)$. The loop is going to calculate the effect of X on $\Pr(Y = 1)$ across the observed range of `threshold`, which is 0 to 35.

```
scalar h_polarization=28.58;
scalar h_threshold=8.59;
scalar h_seatshare=31.37;
scalar h_seatshare_2=(h_seatshare*h_seatshare);
scalar h_incompatibility= 24.41;
scalar h_asymmetry=.545;
scalar h_constant=1;
```

The effect of X on the $\Pr(Y = 1)$ depends on the specific values of all the variables in the model. These lines specify the values of the variables at which I want to calculate the effect of X. Some scholars specify the mean or modal value for each variable. Others set the values to those in a particular case of interest. You should choose the values that are of theoretical interest in your specific application.

```
    generate x_betahat0  = SN_b1*h_polarization
                           + SN_b2*'a'
                           + SN_b3*h_polarization*'a'
                           + SN_b4*h_seatshare
                           + SN_b5*h_seatshare_2
                           + SN_b6*h_incompatibility
                           + SN_b7*h_asymmetry
                           + SN_b8*h_asymmetry*h_seatshare
                           + SN_b9*h_constant;

    generate x_betahat1  = SN_b1*(h_polarization+1)
                           + SN_b2*'a'
                           + SN_b3*(h_polarization+1)*('a')
                           + SN_b4*h_seatshare
                           + SN_b5*h_seatshare_2
                           + SN_b6*h_incompatibility
                           + SN_b7*h_asymmetry
                           + SN_b8*h_asymmetry*h_seatshare
                           + SN_b9*h_constant;

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

You have to decide what the appropriate quantity of interest is that you want in your "marginal effect" plot. Here I am going to plot the *change* in the $\Pr(Y = 1)$ when I increase X, `polarization`, by one unit

from its mean of 28.58. To do this, I am going to calculate Pr($Y = 1$) in some baseline scenario (`0`) where X and the other covariates have their baseline values, and Pr($Y = 1$) in a counterfactual scenario (`1`) where X is one unit higher but the other covariates still have their baseline values. I am then going to calculate the change (`diff`) in Pr($Y = 1$) as I move from the baseline scenario to the counterfactual scenario. The loop is going to allow me to calculate this change in probability at each value of Z, `threshold`, that I specify. In a probit model, Pr($Y = 1$) = $\Phi(X_i\beta)$. As a result, these lines first calculate $X_i\beta$ in the baseline scenario, `x_betahat0`, and in the counterfactual scenario, `x_betahat1`. They then take the normal of these quantities of interest, `prob0` and `prob1`. And finally, they then calculate the difference in these probabilities, `diff`. Every time through the loop, these quantities are calculated for a different value of `a`, i.e., a different value of Z.

Note that you would change these lines of code if you were calculating a different quantity of interest or if you were estimating a different model (duration model, count model, selection model etc.). Basically, you need to know what the quantity of interest is for your particular model and then change these lines of code to calculate exactly what you want.

```
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');

};
```

These lines of code essentially take the mean, 2.5th percentile, and 97.5th percentile, of all the quantities of interest that you have calculated and stores them in `sim.dta`. In effect, we are storing (i) the mean, 2.5th percentile, and 97.5th percentile of Pr($Y = 1$) in the baseline scenario, (ii) the mean, 2.5th percentile, and 97.5th percentile of Pr($Y = 1$) in the counterfactual scenario, and (iii) the mean, 2.5th percentile, and 97.5th percentile of the difference in Pr($Y = 1$) across the two scenarios. These quantities will allow us to produce two-tailed 95% confidence intervals around the estimated effect of X. You will need to change the values in the `_pctile` command if you want to calculate different confidence intervals.

```
drop x_betahat0 x_betahat1 prob0 prob1 diff probhat0 probhat1 diffhat;
```

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

    display "." _c;

} ;
```

These lines of code essentially finish the loop. Once you have calculated and stored the values for the quantities of interest that you need for the first value of `a` in the loop, you need to drop them so that they can be recalculated and posted for the next value of `a` in the loop. This is what the first line does. The second line indicates the increment by which `a` increases in each run of the loop. You should choose an increment that is specific to your application. The third line tells Stata to print a `.` for each run through the loop. By doing this, you can see whether the code is working and how many times Stata has gone through the loop at a given point in time.

```
display "";

postclose mypost;
```

These lines close the data set that you posted containing the quantities of interest that were calculated.

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

restore;

merge using "C:\matt\publications\jop2\webpage\sim.dta";
```

These lines restore the original data set that we preserved earlier and merges in the `sim.dta` containing our quantities of interest.

```
gen yline=0;

gen MV = _n-1;

replace  MV=. if _n>36;
```

The first line creates a variable that will be used to produce a horizontal line at 0 in the marginal effect plot. The last two lines create a variable that will be used as the horizontal axis and runs from the lowest observed value for Z (0) to the highest observed value (35).

```
graph twoway hist threshold, percent color(gs14) yaxis(2)
        ||  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)
        ||  line yline MV,  clwidth(thin) clcolor(black) clpattern(solid)
        ||  ,
            xlabel(0 5 10 15 20 25 30 35, nogrid labsize(2))
            ylabel(0 .001 .003 .005 .007, axis(1) nogrid labsize(2))
            ylabel(0 5 10 15 20 25, axis(2) nogrid labsize(2))
            yscale(noline alt) yscale(noline alt axis(2)) xscale(noline) legend(off)
            yline(0, lcolor(black))
```

```
        yline(.001 .003 .005 .007, lcolor(white))
        xtitle(Effective Threshold, size(2.5))
        ytitle("Marginal Effect of Polarization", size(2.5))
        ytitle("Percent of Observations" , axis(2) size(2.5))
        xsca(titlegap(4)) ysca(titlegap(4)) ysca(axis(2) titlegap(4))
        scheme(s2mono) graphregion(fcolor(white) ilcolor(white) lcolor(white));
```

This produces the marginal effect plot for X. The first part of the command,

```
graph twoway hist threshold, percent color(gs14) yaxis(2)
```

overlays a histogram on the marginal effect and indicates that we want the bars to represent the *percent* of observations, sets the color of the bars, and indicates that the second vertical axis on the right will be associated with the histogram. The second part of the command,

```
line diff_hat MV, clwidth(medium) clcolor(blue) clcolor(black)
```

plots the change in $\Pr(Y = 1)$ associated with a one unit change in X across all values of Z. The third part of the command,

```
line diff_lo MV, clpattern(dash) clwidth(thin) clcolor(black)
```

plots the lower bound of the confidence interval. The fourth part of the command,

```
    ||  line diff_hi MV, clpattern(dash) clwidth(thin) clcolor(black)
```

plots the upper bound of the confidence interval. The fifth part of the command,

```
line yline MV,  clwidth(thin) clcolor(black) clpattern(solid)
```

plots the horizontal line at 0. The remaining lines serve various purposes, such as indicating how the axes should look.