The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–3

Stata tip #: Exploring model consequences by plotting different predictions

Maarten L. Buis
Department of Sociology
Tübingen University
Tübingen, Germany
maarten.buis@uni-tuebingen.de

A useful trick for exploring the consequences of a model is that [R] **predict** does not require the data to be the same as the data used during estimation. That way we can get predictions while fixing some of the explanatory variables at specific values. One area of application of this trick are situations in which something varies with two controls. In that case one can create a graph where one goes on the x axis and the other is presented by a set of example values. In this tip I will illustrate it using a linear regression model that includes an interaction between two variables where one of these was entered as a quadratic curve. This trick is not limited to this situation, it can be meaningfully used whenever one wants to simultaneously explore the relationship between three (or more) variables. This particular application was chosen because with the new factor variable notation ([U] **factor variables**) it has become very easy to include an interaction term or a quadratic term in your model. It is even possible to combine the two: interact two variable of which one (or both) is entered as a quadratic curve. However, how would one interpret the results of such a model? Consider the model below

```
. sysuse auto, clear
(1978 Automobile Data)
. reg price c.mpg##c.weight##c.weight i.foreign
  (output omitted)
```

In this model `weight` is entered as a quadratic curve and interacted with `mpg`. One way to interpret the results of this model is to fix `mpg` at a value — say the mean minus the standard deviation — predict the price for different values of weight, than do this again but fix `mpg` at another value — say the mean, etc. A graph of these predictions against weight would show how the relationship between `weight` and `price` changes when `mpg` changes.

The first step would be to choose the values at which one wants to fix `mpg`. In this example I choose for functions of the mean (m) and the standard deviation (sd): m − 2sd, m − sd, m, m + sd, m + 2sd. So I first collect the mean and the standard deviation and store them in the local macros `‘m’` and `‘sd’`.

```
. qui sum mpg if e(sample)
. local m = r(mean)
. local sd = r(sd)
```

The next step is to fix the explanatory variables at the chosen values. To do so we

will make quite some big changes to our data that we will probably not want to save. So we start this set of commands with [P] **preserve**. When we are done, we will type `restore` and we will get the data back at the stage it was when we typed `preserve`.

If control variables were included in the model, as in this example the variable `foreign`, then these should also be fixed. Here I fix `foreign` at the value 0, which means that we are looking at US ("domestic") cars. The `forvalues` loop below creates 5 variables (`yhat0` till `yhat4`) containing predicted prices while fixing `mpg` at different values. Consider the first iteration of that loop. In that case the local macro `'i'` will be −2, so all observations of `mpg` will contain m − 2sd. The local macro `'j'` will be 0, so the predictions will be stored in `yhat0`. During the second iteration of the loop the local macro `'i'` will be −1, so all observations of `mpg` will contain m − sd, and the predictions will be stored in `yhat1`, etc.

Finally, all these new variables are given the format `%8.0gc` ([D] **format**). This format makes large numbers easier to read by inserting commas. Such a format makes sense here since we are talking about predicted prices of cars, which we would expect to run in the 1000s of dollars.

```
. preserve
. qui replace foreign = 0
.
. forvalues i = -2/2 {
  2.     qui replace mpg = `m´ + `i´*`sd´
  3.     local j = `i´ + 2
  4.     qui predict yhat`j´
  5. }
. format yhat* %8.0gc
```

Now all that is left is graph these predicted prices. When graphing these predicted prices it would be nice to emphasize the ordinal nature of these predictions, `yhat1` represents predictions at lower values of `mpg` than `yhat2`, etc. This is done by graphing all predictions as a solid line (using the `lpatter()` option) and choosing darker grey tones for predictions at higher values of `mpg` (using the `lcolor()` option), see: [G] **connect options**. The resulting graph is shown in Figure 1.

```
. sort weight
. twoway line yhat0 yhat1 yhat2 yhat3 yhat4 weight, ///
>        ytitle("predicted price (US {c S|})")      ///
>        lpattern(solid solid solid solid solid)    ///
>        lcolor(gs13 gs10 gs7 gs4 gs1)              ///
>        legend( order( - "mpg"                     ///
>                       1 "mean - 2*sd"             ///
>                       2 "mean - 1*sd"             ///
>                       3 "mean"                    ///
>                       4 "mean + 1*sd"             ///
>                       5 "mean + 2*sd" ))
. restore
```

Figure 1: Predicted prices of US cars by weight and mpg