f_able: Estimation of marginal effects for models with alternative variable transformations

by

Fernando Rios-Avila

Levy Economics Institute

Abstract

`margins` is a powerful post-estimation command that allows the estimation of marginal effects for official and community-contributed commands, with well-defined predicted outcomes (see `predict`). While the use of `factor variable` notation allows us to easily estimate marginal effects when interactions and polynomials are used, estimation of marginal effects of when other types of transformations such as splines, logs, or fractional polynomials, among others, are used remains a challenge. This paper describes how `margins` capabilities can be extended to analyze other variable transformations using the command `f_able`.

1. Introduction

`margins` is a powerful post-estimation command that was introduced in Stata 11, allowing the estimation of marginal effects for all official estimation commands, and any community-contributed command with a properly defined predicted outcome program (see `predict`). As described in Williams (2012) the introduction of margins, at its companion `marginsplot` (in Stata 12), provides a great tool for analyzing and providing meaningful information that is easier to interpret and describe. Furthermore, in combination with `factor variable` notation (also introduced in Stata 11), it also allows users to easily estimate marginal effects when interactions and polynomials of continuous variables, and interactions with discreet variables, are used without any additional work from the user.

Despite these features, the ability of `margins` to estimate marginal effects is limited to the use of simple polynomials and interactions. There is a reason for this. Unless further information is provided to a command, Stata cannot identify the possible interdependencies across explanatory variables.

As far as I know, there are only 2 commands where this limitation is not binding. As described in Poi (2008), `nl` can be used to easily obtain marginal effects when using functions other than interactions of polynomials, even when the model is nonlinear in parameters, but assuming an additive error. `npregress series`, one of the newest additions to Stata 16 nonparametric analysis, can also estimate marginal effects of arbitrary transformations of the independent variables (splines, B-splines and polynomials), based on numerical derivations. Unfortunately, both models are based on least-squares type of estimators, they cannot be used in the case of nonlinear models like logit or probit, and in the case of `npregress series` users cannot freely choose what transformations to use. The only other program that proposes a strategy that may help to obtain marginal effects, more specifically marginal means, using transformed covariates was proposed by Royston (2013). Their command, `marginscontplot` uses a predefined mapping between original and transformed covariates to appropriately handle the constructed variables in `margins`.

In this article, I describe a simple strategy, implemented by `f_able`, that allows the estimation of marginal effects of a larger set of nonlinear transformations using `margins` and the option `nochain [numerical noestimcheck]`. Section 2 provides a review of how marginal effects should be estimated, emphasizing on some of the computational challenges that are bypassed by the use of factor notation. Section 3 provides a simple approach for the estimation of marginal effects for the case of Spline regressions and compares it to a simplified case using `npregress series`. Section 4 introduces 3 commands that would help to prepare the data for the estimation of marginal effects for any arbitrary variable transformation and shows how it can be used to estimate marginal effects using any variable transformation. Section 5 concludes.

2. Marginal effects: Theoretical approach

As mentioned before, marginal effects are useful statistics to measure the impact that a change in the independent variables will have on the dependent variable, assuming other covariates remaining constant. At the beginning of most introductory econometric courses, very little emphasis is put on understanding this concept because, for the case of linear regressions, marginal effects are typically equal to the coefficient associated with the variable analyzed. Consider the following linear regression model:

$$y_i = b_0 + b_1 x_{1i} + b_2 x_{2i} + e_i \qquad (1)$$

Under the standard assumptions of exogeneity and correct model specifications (see for example Wooldridge (2016)), the coefficients of this model can be estimated using Ordinary Least Squares (OLS). In this simple model, the marginal effect of x_1 on y and x_2 on y will be given by b_1 and b_2, respectively. Mathematically, it would be:

$$\frac{\partial y_i}{\partial x_{1i}} = b_1 \,; \frac{\partial y_i}{\partial x_{2i}} = b_2$$

This implies that for a simple linear regression (like equation(1)), where each variable appears only once, and without any transformation, marginal effects are directly identified by the estimated coefficients.

Soon after, students are introduced to the idea that nonlinear transformations and interactions of the dependent variables can also be included in the linear regression model. Because the model is still linear in parameters it can be estimated using OLS. However, additional care needs to be taken when estimating the marginal effects, to account for the interdependence of variable transformations. Consider, for example, the following model:

$$y_i = b_0 + b_1 x_{1i} + b_2 x_{1i}^2 + b_3 x_{2i} + b_4 x_{1i} x_{2i} + e_i \qquad (2)$$

In this model, the marginal effect of x_1 on y is no longer a constant, and it now depends on the values of x_1 and also of x_2. Using calculus, however, it is easy to derive the marginal effects from this model:

$$\frac{\partial y_i}{\partial x_{1i}} = b_1 + 2b_2 x_{1i} + b_4 x_{2i}; \frac{\partial y_i}{\partial x_{2i}} = b_3 + b_4 x_{1i}$$

Because these effects are no longer constant, one has to decide what to report as the presentative marginal effect. While one can always report plots showing all possible values these marginal effects will take, the standard practice is presenting average marginal effects or the marginal effects at the mean. For the example above, they both will be the same:

$$E\left(\frac{\partial y_i}{\partial x_{1i}}\right) = E(b_1 + 2b_2 x_{1i} + b_4 x_{2i}) = b_1 + 2b_2 \bar{x}_1 + b_4 \bar{x}_2$$

$$E\left(\frac{\partial y_i}{\partial x_{2i}}\right) = E(b_3 + b_4 x_{1i}) = b_3 + b_4 \bar{x}_1$$

With this information in hand, and assuming x_1 and x_2 are non-stochastic, standard errors associated with this average marginal effects can be estimated right away,[1] and the technical part of the analysis would be done.

The problem with most software (including Stata) is that, unless additional information is provided, it may not recognized that some variables are interrelated to each other by constructions and that the assumption of "everything else remaining constant" is incorrect. In the case of Stata, until factor variables came along, it could not automatically adjust for these interrelations, providing incorrect estimations of marginal effects, unless further steps were considered. It still fails to grasp these interrelations when we step beyond simple interactions or polynomials.

3. Marginal effects: Empirical approach

For this section I will use the dataset "Fictional data on monthly drunk driving citations", available online. Consider now the following model:

$$citations_i = b_0 + b_1 fines_i + b_2 fines_i^2 + e_i \qquad (3)$$

Before Stata 11 and factor notation, if we wanted to estimate a model like this, we would need to create all variables before including them in the model. For example, creating a variable named `fines2` to be equal to $fines_i^2$:

```
webuse dui,clear
gen fines2=fines^2
```

and fit the model using the command regress:

```
regress citations fines fines2
```

```
      Source |       SS           df       MS      Number of obs   =       500
-------------+----------------------------------   F(2, 497)       =    189.57
       Model |    20750.38         2    10375.19   Prob > F        =    0.0000
```

[1] A detailed explanation of how standard errors are estimated using margins can be found https://www.stata.com/support/faqs/statistics/compute-standard-errors-with-margins/

```
    Residual |   27200.458       497  54.7292917   R-squared       =    0.4327
-------------+----------------------------------   Adj R-squared   =    0.4305
       Total |   47950.838       499  96.0938637   Root MSE        =    7.3979

------------------------------------------------------------------------------
   citations |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       fines |  -47.10883   7.691611    -6.12   0.000    -62.22091   -31.99674
      fines2 |    1.98084   .3885844     5.10   0.000      1.21737    2.744311
       _cons |   293.0067   37.94286     7.72   0.000     218.4585    367.5549
------------------------------------------------------------------------------
```

Because there is no way for Stata to know that `fines2=fines^2`, using `margins` to calculate marginal effect of fines on citations would produce the wrong answer, since `fines2` would not be handled correctly.

```
margins, dydx(fines)

Average marginal effects                        Number of obs     =        500
Model VCE    : OLS

Expression   : Linear prediction, predict()
dy/dx w.r.t. : fines

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       fines |  -47.10883   7.691611    -6.12   0.000    -62.22091   -31.99674
------------------------------------------------------------------------------
```

However, using calculus, average marginal effects can be easily derived by hand by taking the derivative of equation 3 with respect to fines, estimating the average as the point of reference, and use a command like `lincom` to calculate the marginal effects and standard errors:

```
sum fines, meanonly

lincom _b[fines]+2*_b[fines2]*`r(mean)'

 ( 1)  fines + 19.7904*fines2 = 0

------------------------------------------------------------------------------
   citations |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         (1) |  -7.907201   .4236816   -18.66   0.000    -8.739629   -7.074773
------------------------------------------------------------------------------
```

Of course, ever since Stata 11, estimating marginal effects for a model like this is much easier. Using factor notation we simply add the squared parameter, and let `margins` handle the interaction on its own:

```
qui:regress citations fines c.fines#c.fines

margins, dydx(fines)

Average marginal effects                        Number of obs     =        500
Model VCE    : OLS

Expression   : Linear prediction, predict()
dy/dx w.r.t. : fines

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       fines |  -7.907201   .4236816   -18.66   0.000    -8.739629   -7.074773
------------------------------------------------------------------------------
```

My understanding of how this works is that whoever did the coding behind `margins` and factor notation was able to "teach" Stata how to take derivatives of polynomials. In other words, Stata recognizes that when there is an expression like "`c.var1#c.var1`", internally it "knows" the analytical derivative is `2*c.var1`. Thus, margins simply uses this information to handle the squared parameter (`c.fines#c.fines`), before providing you with the result.

While this has been a big improvement for a better understanding of marginal effects, it does have its limitations. For example, `margins` would not be able to estimate marginal effects for the following models:

$$\text{Model 1: } citations_i = b_0 + b_1(1/fines_i) + e_i \quad (4)$$

$$\text{Model 2: } citations_i = b_0 + b_1 fines_i^{0.5} + e_i$$

$$\text{Model 3: } citations_i = b_0 + b_1 fines + b_2 \max(fines - 9.9, 0) + e_i$$

Even though, mathematically, the average marginal effects (AME), and marginal effects at the means (MEM) can be derived straight forward:

	AME	MEM
Model 1:	$b_1 E\left(-\frac{1}{fines_i^2}\right)$	$-b_1 \frac{1}{E(fines_i)^2}$
Model 2:	$0.5 * b_1 E\left(fines_i^{-0.5}\right)$	$0.5 * b_1 E(fines_i)^{-0.5}$
Model 3:	$b_1 + b_2 E\left(1(fines_i > 9.9)\right)$	$b_1 + b_2 1(E(fines_i) > 9.9)$

Which can be used to estimate the average marginal effects by hand. For simplicity, I will concentrate on the estimation of the average marginal effects:

```
. webuse dui, clear
(Fictional data on monthly drunk driving citations)
. gen i_fines=1/fines
. gen ni_fines2=-1/fines^2
. gen fines05=fines^.5
. gen i_fines05=0.5*fines^-.5
. gen fines_99=max((fines-9.9),0)
. gen dfines_99=fines>9.9
* model 1
. qui:regress citations i_fines
. sum ni_fines2, meanonly
. lincom _b[i_fines]*`r(mean)'

 ( 1)  - .0104108*i_fines = 0
------------------------------------------------------------------------------
   citations |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         (1) |  -8.091099   .4224389   -19.15   0.000    -8.921081   -7.261117
------------------------------------------------------------------------------
* model 2
. qui:regress citations fines05
. sum i_fines05, meanonly
. lincom _b[fines05]*`r(mean)'

 ( 1)  .1593264*fines05 = 0
------------------------------------------------------------------------------
   citations |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         (1) |  -8.010351   .4314167   -18.57   0.000    -8.857972   -7.162729
------------------------------------------------------------------------------
* model 3
. qui:regress citations fines fines_99
. sum dfines_99, meanonly
. lincom _b[fines]+_b[fines_99]*`r(mean)'

 ( 1)  fines + .5*fines_99 = 0

------------------------------------------------------------------------------
   citations |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         (1) |  -7.926694   .4271729   -18.56   0.000    -8.765981   -7.087407
------------------------------------------------------------------------------
```

Of course, for these models, one also have the option of using **nl**, to estimate the marginal effects, which requires less work:[2]

```
qui:nl (citations = {b0}+{b1}/fines), variable(fines)
margins, dydx(fines)

qui:nl (citations = {b0}+{b1}*fines^.5), variable(fines)
margins, dydx(fines)

qui:nl (citations = {b0}+{b1}*fines+{b2}*max((fines-9.9),0)), variable(fines)
margins, dydx(fines)
```

This flexibility of `nl` to estimate marginal effects with non-standard transformations motivates the questions: why is it that `nl` can "correctly" estimate marginal effects, whereas `regress` can't? The answer is rather simple. We are not using a constructed variable in the model, instead, we are using the original variable and letting `nl` handle the construction of the new variable.

This means that, while we see this model is being estimated:

```
citations = {b0}+{b1}*fines^.5
```

What may be happening in the background is that Stata identifies what elements of this code are parameters to be estimated (those within brackets), and what elements need to be created (`fines^.5`), before fitting the model. In other words, Stata is simply estimating the following:

```
citations = {b0}+{b1}*__000000
```

where `__000000` is a temporary variable, you never see, that was constructed as `fines^.5`. The difference is that `nl` knows that `__000000= fines^.5`.

But, how is it that `nl` knows what the derivative of $fines^{0.5}$ is $0.5fines^{-0.5}$. The answer is that `nl` does not know. The only type of analytical derivatives Stata may know about is when you have interactions (again factor notation). However, because it "remembers" how a variable was constructed, it can use numerical derivatives to make a reasonable approximation for the analytical derivative.

[2] For this to work, we need to indicate to nl that fines is a variable in the model. Outputs are identical to the ones produced by hand.

For the simplified case above, the numerical derivative for the transformation $fines^{0.5}$ can be approximated as follows:

$$\frac{\partial\left(fines_i^{0.5}\right)}{\partial fines_i} = \frac{(fines_i + h)^{.5} - (fines_i - h)^{.5}}{2h} \text{ for a sufficiently small h}$$

This expression is surprisingly accurate.[3] For this example, when $h = 1$ the largest absolute difference between the numerical and analytical derivative is 0.000423, whereas when $h = 1/2\text{^}16$, the largest difference is 6.58e-09. This simply implies that `margins` does not need to know how to obtain analytical derivatives, since it can use numerical derivatives instead, and use this information to estimate the appropriate marginal effects.

4. `f_able`: marginal effects for arbitrary variable transformations

In the previous sections, I sketched out how marginal effects should be estimated, and how to use that information to obtain average marginal effects, comparing the step-by-step procedure with what `margins` does. I also described that `nl` is capable to estimate marginal effects for variable transformations other than interactions, using numerical derivatives to approximate the analytical derivatives. As a matter of fact, when nonlinear models are estimated, numerical derivatives are already used to estimate marginal effects.[4]

In other words, Stata already has the capabilities to estimate marginal effects when transformations other than interactions and polynomials are used in a model. There is only one aspect that needs to be addressed. How to tell Stata that a variable Z is constructed based on another variable X within the model specification?. One possible solution, which I suggest in this paper, is to use the variable

[3] As a matter of fact, as indicated on its help file, the function `deriv()` in `mata` uses this approximation to compute numerical derivatives.

[4] This is partially correct. Some of nonlinear models implemente in Stata, like logit, probit, poisson, among others, have been programmed so they obtain marginal effects using analytical derivates rather than numerical ones. However, one can still request marginal effects to be estimated using numerical derivatives.

label to “store” the transformation used to create the variable of interest. To automatize this procedure, I propose two small programs that “wrap” around Stata’s built-in commands `generate` and `replace`:

```
program fgen
      syntax newvarname =/exp [if] [in]
      gen `typelist' `varlist'=`exp'
      label var `varlist' "`exp'"
end
program frep
      syntax varname =/exp [if] [in]
      replace   `varlist'=`exp'
      label var `varlist' "`exp'"
end
```

These two commands simply do one thing. When a new variable is created, it will label it with the expression used to create it, and if the values are replaced, it will change the label to the new expression used.

Consider model 3 from equation (4). I can create the variable of interest using the command defined above. Since I made a mistake while creating this code, I will rectify it, using the second command and replace the values `fines2` with the correct content:

```
. fgen fines2=max(fines-9,0)

. describe fines2

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------
fines2          float   %9.0g                 max(fines-9,0)

. frep fines2=max(fines-9.9,0)
(420 real changes made)

. describe fines2

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------
fines2          float   %9.0g                 max(fines-9.9,0)
```

In addition to setting up the data for the next step of the estimation of marginal effects, these commands may also be useful for keeping track of how variables are created or modified.

The final step is to explicitly tell Stata that a particular variable is constructed based on other variables in the model, and that when marginal effects are estimated, the constructed variable needs to be updated every time the original variable changes. This can be done with the two following programs:

```
program f_able, eclass
	syntax, [* NLvar(varlist)]
	_ms_dydx_parse `nlvar'
	if "`e(predict_old)'"=="" {
		ereturn local predict_old  `e(predict)'
		ereturn local predict      f_able_p
	}
	foreach i of varlist `nlvar' {
		local fnc:variable label `i'
		ereturn hidden local _`i' `fnc'
	}
	ereturn local nldepvar `nlvar'
end

program f_able_p
	syntax newvarname [if] [in], [*]
	local idepvar `e(nldepvar)'
	foreach i of local idepvar {
			tempvar _`i'
			qui:clonevar `_`i''=`i'
			qui:recast double `i'
			qui:replace `i'=`e(_`i')'
	}
	`e(predict_old)' `0'
	foreach i of local idepvar {
	    if "`i'"!="_cons" {
			qui:replace `i'=`_`i''
		}
	}
end
```

The first program, `f_able`, does three things. First, it adds information, to any previously estimated model, indicating which variables are constructed variables using `e(nldepvar)`. It also adds hidden macros with information regarding the data transformation used.[5] Lastly, it redirects `predict` from the original `e(predict)` command to the one I defined below as `f_able_p`, but keeping the original information in `e(predict_old)`.

The second program, `f_able_p`, has the only purpose of updating the constructed variables, identified in `e(nldepvar)`, before proceeding to obtain the predicted values

[5] While this step is unnecessary in most cases I have considered, in some examples `margins` "drops" the variable label information. In those cases, storing the information as part of the estimation commands seems to be better alternative.

appropriate for the estimated command `e(predict_old)`, using the information previously stored in the hidden macros.

With these two pieces of code, the last step is to simply call `margins` for the estimation of the marginal effects, using the option `nochainrule`.[6] This is an option often ignored when using official Stata commands. However, as its help file indicates:

"`nochainrule` is safer because it makes no assumptions about how the parameters and covariates join to form the response."

This implies that when a model like the following is estimated:

```
regress citations fines fines2

f_able, nl(fines2)
```

Marginal effects concerning will be calculated using the coefficient of `fines`, and the coefficient of `fines2` times the numerical derivative of `fines2` with respect to `fines` if `fines2` was declared as a constructed variable. Let see how this works.

```
. qui:regress citations fines fines2
. f_able, nlvar(fines2)
. margins, dydx(fines) nochainrule
Average marginal effects                        Number of obs     =        500
Model VCE    : OLS

Expression   : Fitted values, predict()
dy/dx w.r.t. : fines

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       fines |  -7.926694   .4271729   -18.56   0.000    -8.763937   -7.089451
------------------------------------------------------------------------------
```

The first line estimates the model with `citations` as a dependent variable and `fines` and `fines2` as independent variables. `fines2` is the one we defined previously as `max(fines-9.9,0)`. The second line calls `f_able` to identify that `fines2` is a constructed variable, and the last step estimates the

[6] Some commands, like `logit`, `probit` and `poisson`, also require to include the option `numerical`, to use numerical derivatives.

marginal effects using `nochainrule`. The results are the same as the ones done by hand, or those using `nl` command. They can also be replicated using the `npregress series`.

```
. npregress series citations fines, spline(1) knots(1)

Computing approximating function

Computing average derivatives

Linear-spline estimation                    Number of obs      =            500
                                            Number of knots    =              1
------------------------------------------------------------------------------
             |               Robust
   citations |     Effect   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       fines |  -7.926694   .4772213   -16.61   0.000     -8.86203   -6.991357
------------------------------------------------------------------------------
Note: Effect estimates are averages of derivatives.
```

Notice that while the point estimates are the same, the standard errors produced by `npregress series` are somewhat larger than those produced with `nl` or with the proposed strategy.

We can also compare `f_able` to the output we could obtain using factor notation:

```
qui:regress citations c.fines##c.fines##c.fines
. margins, dydx(fines)
Average marginal effects                        Number of obs     =        500
Model VCE    : OLS
Expression   : Linear prediction, predict()
dy/dx w.r.t. : fines
------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       fines |  -7.928817   .4226225   -18.76   0.000    -8.759168   -7.098465
------------------------------------------------------------------------------


. frep fines2=fines^2
(500 real changes made)
. fgen fines3=fines^3
. qui:regress citations fines fines2 fines3
. f_able, nlvar(fines2 fines3)
. margins, dydx(fines) nochainrule

Average marginal effects                        Number of obs     =        500
Model VCE    : OLS
Expression   : Fitted values, predict()
dy/dx w.r.t. : fines

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       fines |  -7.928819   .4226228   -18.76   0.000    -8.757144   -7.100493
------------------------------------------------------------------------------
```

As can be seen, this method replicates the results when using factor notation up to 5 decimal places, but some degree of precision is lost due to forced use of numerical derivatives.

Lastly, we do a more challenging estimation that combines the use of factor notation with `f_able` and compares it to the output of the `npregress series`.

```
. npregress series citations fines i.csize, spline(2) knots(1)

Computing approximating function

Computing average derivatives

Quadratic-spline estimation               Number of obs      =            500
                                          Number of knots    =              1
------------------------------------------------------------------------------
             |               Robust
   citations |     Effect   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       fines |  -7.590817   .3533786   -21.48   0.000    -8.283427   -6.898208
             |
       csize |
(medium vs small)  |    5.48074   .5827045     9.41   0.000      4.33866    6.622819
 (large vs small)  |   10.69879   .6043375    17.70   0.000     9.514311    11.88327
------------------------------------------------------------------------------
Note: Effect estimates are averages of derivatives for continuous covariates and averages
of contrasts for factor covariates.
```

The same can be replicated using regress and `f_able`:

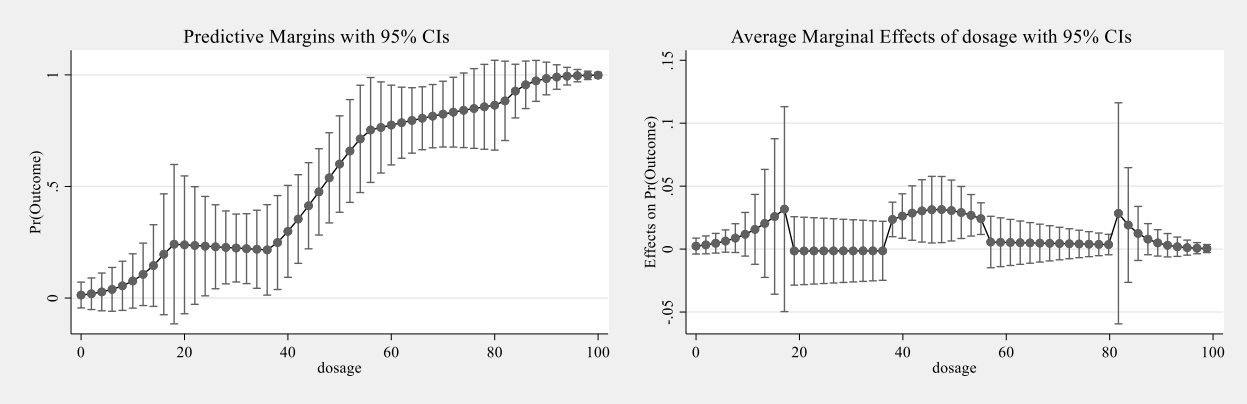
```
. webuse dui,clear
. fgen double fines2=fines^2
. fgen double fines3=max(fines-9.9,0)^2
. qui:regress citations c.(fines fines2 fines3)##i.csize
. f_able, nlvar(fines2 fines3)
. margins, dydx(fines csize) nochainrule

Average marginal effects                        Number of obs     =        500
Model VCE    : OLS

Expression   : Fitted values, predict()
dy/dx w.r.t. : fines 2.csize 3.csize

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       fines |  -7.590817   .3350831   -22.65   0.000    -8.247568   -6.934066
             |
       csize |
      medium |   5.480738   .6522748     8.40   0.000     4.202303    6.759174
       large |   10.69879   .6171921    17.33   0.000     9.489118    11.90847
------------------------------------------------------------------------------
Note: dy/dx for factor levels is the discrete change from the base level.
```

Once again, we can replicate the output with very small differences in the point estimates, but some differences in terms of the standard errors, possibly due to how degrees of freedom are used in npregress.

We can also use this strategy for models other than OLS. Consider, for example, the dataset `mksp2`. Similar to the example provided in the help file for the command `mkspline`, I create 4 variables to allow for a linear spline with 4 knots, and estimate a logit model of outcome against `dosage` and the constructed variables. Margins need to include the options `nochain` and `numerical`. I produce both the predicted probabilities and the marginal effects across various values of `dosage`:

```
webuse mksp2, clear
fgen dos1=max(dosage-17.5,0)
fgen dos2=max(dosage-36.5,0)
fgen dos3=max(dosage-55.5,0)
fgen dos4=max(dosage-81.5,0)
qui:logit outcome dosage dos1 dos2 dos3 dos4
f_able, nl(dos1 dos2 dos3 dos4)
qui:margins ,   nochain numerical at(dosage=(0(2)100))
marginsplot, name(m1)
qui:margins ,  dydx(dosage) nochain numerical at(dosage=(0(2)100))
marginsplot, name(m2)
graph combine m1 m2, xsize(8) scale(1.5)
```

Figure 1. Predicted probabilities and Marginal effects



5. Conclusion

In this article, I have described how marginal effects can be estimated based using analytical derivatives, as well as numerical derivatives. I have also introduced 2 small programs that enable margins to estimate marginal effects when using transformations beyond variable interactions and polynomials.

These commands can be used to estimate marginal effects for other official Stata commands, as well as other community-contributed commands that can produce sensible predicted outcomes. The strategy does have four limitations: first, the estimated marginal effects depend on the precision of the forced numerical derivatives; second, it requires the original variable to be present in the model specification so that marginal effects can be used; third, the information for the variable construction is limited to 80 characters, based on the limit of variables labels length; and forth, because `f_able` is forcing `margins` to do something it is not meant to do, one may experience difficulties estimating marginal effects and standard errors when the request is particularly complex.

While the limitation regarding the precision of the estimates is unavoidable, the other limitations can be circumvented to some extent. First, the original variable can be added to the list of explanatory variables using "`o.`". This omits the original variable from the estimation but keeps it in the list of explanatory variables, allowing the estimation of margins. Second, it is possible to modify the programs to store and gather the transformation information as a variable `note`. This has fewer limits on the length of the stored information.

The last limitation can be addressed with careful troubleshooting. While the option `nochain` forces most commands to use numerical derivatives for the estimation of marginal effects, some commands like probit, logit, and Poisson require you to use the options `nochain numerical`. If point estimates appear as missing, the option `noestimcheck` can be used to bypass some of the safety checks in `margins` that produce this error. Finally, if standard errors are missing when estimating

marginal effects for multiple points of interest, they could still be estimated by using fewer points of interest or even using one point of interest at the time.

6. References

Poi, B. P. 2008. "Stata tip 58: nl is not just for nonlinear models." *Stata Journal* 8 (1):139-141.

Royston, Patrick. 2013. "marginscontplot: Plotting the marginal effects of continuous predictors." *The Stata Journal* 13 (3):510-527.

Williams, Richard. 2012. "Using the margins command to estimate and interpret adjusted predictions and marginal effects." *The Stata Journal* 12 (2):308-331.

Wooldridge, Jeffrey M. 2016. *Introductory econometrics: A modern approach*: Nelson Education.