

# f\_able: Estimation of marginal effects with transformed covariates

## Taking Margins a step further

Rios-Avila, Fernando<sup>1</sup>

<sup>1</sup>friosavi@levy.org  
Levy Economics Institute

UK Stata Conference, September 2020

# Table of Contents

- 1 Introduction
- 2 How to estimate marginal/partial effects
- 3 Margins and Factor
- 4 Limitations
- 5 f\_able. Going Beyond margins
- 6 Conclusions

# Introduction

- Marginal effects tells us how a dependent variable (outcome)  $y$  changes when an independent variable  $x$  changes, assuming everything else constant ( $e$  and  $z$ 's).

$$y = b_0 + b_1x + b_2z + e$$

- For linear models, with no interactions or polynomials, marginal effects are equal to their coefficients:

$$\frac{dy}{dx} = b_1 \& \frac{dy}{dz} = b_2$$

- However, when there are interactions, polynomials, or other transformations, further work is needed.

# Estimating Marginal effects

- When interactions or polynomials are used, marginal effects should be obtained estimating equation derivatives:

$$y = b_0 + b_1x + b_2x^2 + b_3z + b_4zx + e$$

$$\frac{dy}{dx} = b_1 + 2b_2x + b_4z$$

$$\frac{dy}{dz} = b_3 + b_4x$$

- Main difference with simple linear model?
  - Marginal effects no longer constant
  - Coefficients alone are not useful
  - Derivatives are needed to obtain the effects.

# Estimating Marginal effects: Non-linear model

- When the model is nonlinear, the problem is :

$$y = G(b_0 + b_1x + b_2x^2 + b_3z + b_4zx)$$

$$y = G(XB)$$

$$\frac{dy}{dx} = \frac{dG(XB)}{d(XB)} * (b_1 + 2b_2x + b_4z)$$

- In Addition to obtaining derivatives of XB wrt x, we also need to find the derivative of G() wrt XB

# Estimating Marginal effects

How to proceed in this case? what to report? There are many options:

$$APE = E \left( \frac{dy}{dx} \right)$$

$$PEA = \frac{dy}{dx} | X = \bar{x}; z = \bar{z}$$

$$PE\_at\_X = \frac{dy}{dx} | X = X; z = Z$$

Or report "ALL" effects for each observation in the data.  
Then "simply" estimate SE.

# Empirical Estimation of Marginal effects

- Before Stata 11, estimation of marginal effects for models with interactions was "hard".
- You needed to create the variables "by hand", and adjust marginal effects on your own:

```
. webuse dui, clear
. gen fines2=fines*fines
. reg citations fines fines2
. sum fines2
. lincom _b[fines]+2*_b[fines2]*'r(mean)'
```
- Otherwise, using the old `-mfx-` or the new `-margins-` would give you incorrect results.
- why? because Stata does not recognize that  $fines2 = fines^2$ . Fines2 is assumed constant.

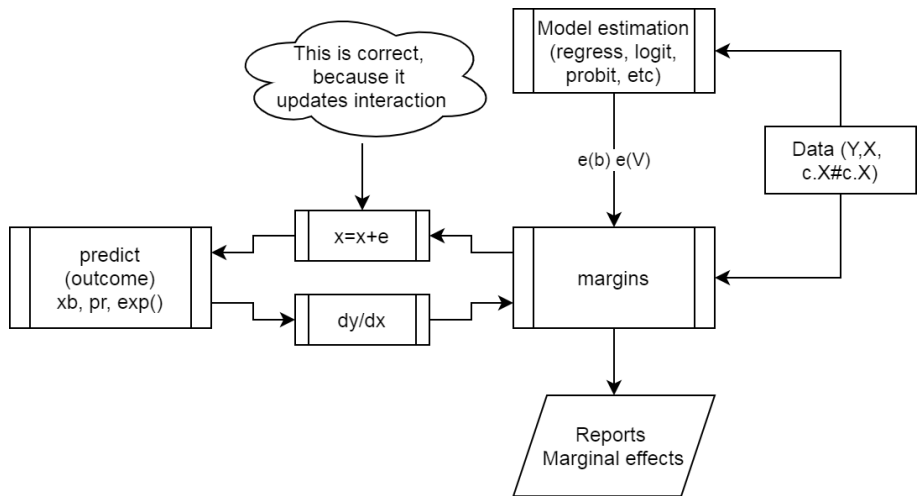
# Margins and Factor notation, and limitations

- Stata 11 introduced the use of factor notation, and margins.
- Factor notation (c. # i.) facilitates adding interactions to models, so that correct marginal effects can be estimated using `margins`
- Marginal effects for the previous model can be easily estimated:
 

```
. webuse dui, clear
. reg citations fines c.fines#c.fines
  (where c.fines#c.fines=fines^2)
. margins, dydx(fines)
```
- Internally, `margins` understand `c.fines#c.fines` depends on `fines`. (And probably estimates analytical derivatives to obtain the PE).
- when nonlinear models are involved `margins` calls on `predict` if one is interested on an outcome different from the linear index.



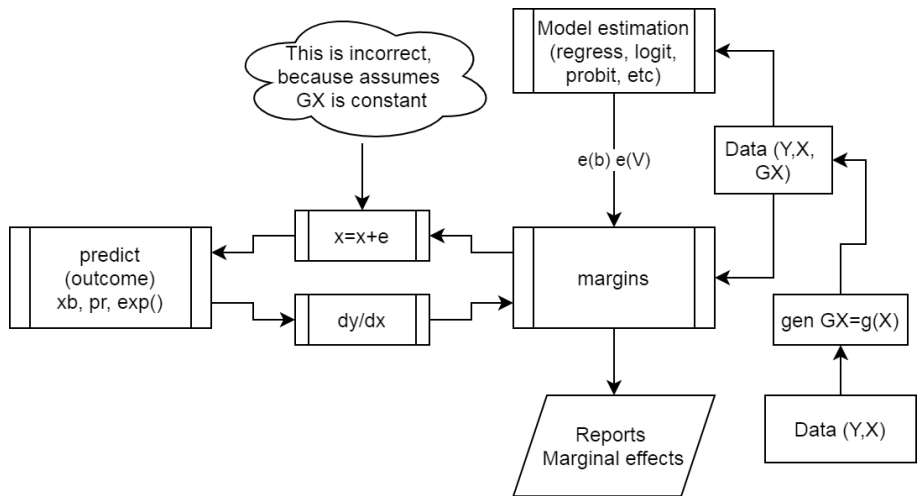
# How margins Works?



# Limitations of margins

- What If one is interested in using other variable transformations, for example:  $fines^5$ ,  $\log(fines)$ ,  $splines$ ,  $fracpoly$ , etc
- In any of these cases, margins will not work.
- why? Because these variables will have to be created manually, and Margin will not recognize they all depend on fines.
- One solution, estimate the derivatives manually, and calculate corresponding SE.
- Same as before [factor notation](#).

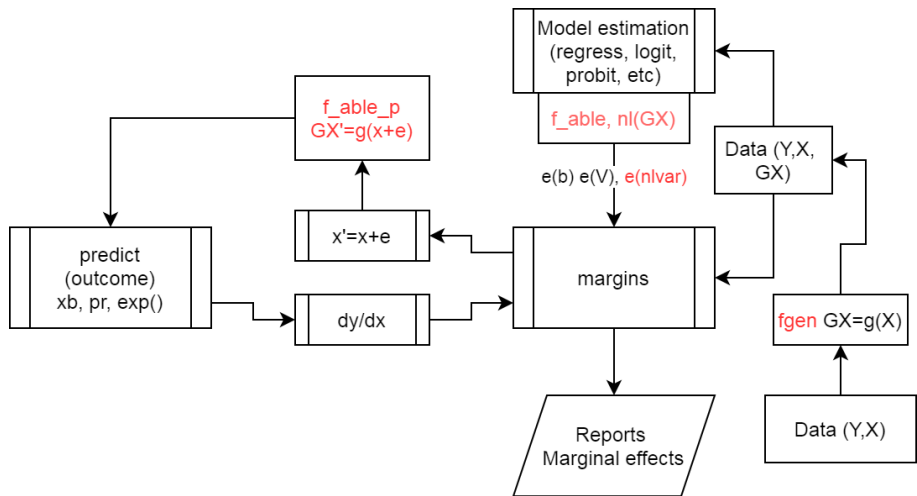
# Why does it fail?



## Beyond factor notation

- Some other commands in Stata are already able to control for "unusual" variable transformations (`nl` and `npregress series`).
- However, for any command being able to use those capabilities, one needs to solve three problems:
  - Store information of how a variable is created.
  - Identify that a variable is a `constructed` variable.
  - Use that information to update constructed variables, and obtain partial effects.
- Here is where `f_able` helps solving these problems.

# How does f\_able works?



## f\_able package: fgen and frep

- To solve the first problem, I propose fgen and frep. These commands are wrappers around generate and replace that stores how the variable was generated, as a label or note.

```
. ssc install f_able
. qui:fgen fines2=fines^2
. describe fines2
```

variable name	storage type	display format	value label	variable label
fines2	double	%10.0g		fines^2

```
. qui:frep fines2=fines*fines
. describe fines2
```

variable name	storage type	display format	value label	variable label
fines2	double	%10.0g		fines*fines

## f\_able package: f\_able

- To solve the second problem, I propose `f_able`. This is a post estimation command that identifies what variables in a model are "constructed" variables, adding information to any previously estimated model, and redirecting the `predict` sub-command to `f_able_p`.

```
. qui:reg citations fines fines2
. f_able, nl(fines2)
. ereturn list, all
scalars: (omitted)
macros: (other macros omitted)
      e(nldepvar) : "fines2"
      e(predict)  : "f_able_p"
      e(predict_old) : "regres_p"
Hidden macros: (other hidden macros omitted)
      e(_fines2) : "fines*fines"
```

## f\_able package: f\_able\_p

- To solve the third problem, I propose `f_able_p`. This passive command uses the information left by `f_able` to update all constructed values when the original variable changes, before using `predict` for the margins estimation.
- Only difference, when calling `margins` we need to include the option `nochain`, so numerical derivatives are used.

```
. qui:reg citations fines fines2
. f_able, nl(fines2)
. margins, dydx(fines) nochain
```

```
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	-7.907201	.4236816	-18.66	0.000	-8.737602	-7.0768



# f\_able syntax

\* Step 1: Generate variables

```
fgen/frep fx1= "gen-able" function of x's
```

```
fgen/frep fx2= "gen-able" function of x's
```

```
fgen/frep fxk= "gen-able" function of x's
```

\* Step 2: Model estimation: Any model

\* Step 3: Declare constructed variables:

```
f_able, nl(fx1 fx2 ... fxk)
```

\* Step 4: Margins

```
margins, dydx(x1 x2 ..) nochain numerical [other options]
```

\* Step 5: Additional post estimation (if no standard errors produced)

```
f_symev/f_symrv
```

## Example: A model of Charity

```

use charity, clear
fgen lavggift=log(avggift)
fgen lweekslast=log(weekslast)
fgen lmailsyearch=log(mailsyear)
fgen lpropresp=log(propresp)

*Simple OLS
reg gift resplast weekslast mailsyear propresp avggift , robust
margins, dydx(resplast weekslast mailsyear propresp avggift) post
est sto model1

*OLS with LOG(Var)
reg gift resplast weekslast mailsyear propresp avggift l*, robust
f_able, nl(lavggift lweekslast lmailsyearch lpropresp)
margins, dydx(resplast weekslast mailsyear propresp avggift) nochain post
est sto model2

```

## Example: A model of Charity

\*Poisson with LOG(var)

```
poisson gift resplast weekslast mailsyear propresp avggift l*, robust
f_able, nl(lavggift lweekslast lmailsyear lpropresp)
margins, dydx(resplast weekslast mailsyear propresp avggift) ///
nochain numerical post
est sto model3
```

\*Tobit with LOG(var)

```
tobit gift resplast weekslast mailsyear propresp avggift l*, vce(robust) ll(0)
f_able, nl(lavggift lweekslast lmailsyear lpropresp)
margins, dydx(resplast weekslast mailsyear propresp avggift) ///
nochain numerical predict(ystar(0,.)) post
est sto model4
```

# Example: A model of Charity

```
. esttab model1 model2 model3 model4, mtitle("S OLS" "OLS w/Logs" "Poisson" "Tobit") ///
se star(* .1 ** .05 *** .01)
```

	(1)	(2)	(3)	(4)
	S OLS	OLS w/Logs	Poisson	Tobit
resplast	1.514** (0.719)	3.527*** (0.990)	2.743*** (0.634)	3.094*** (0.605)
weekslast	-0.0186*** (0.00590)	0.0755*** (0.0212)	0.105*** (0.0178)	0.0953*** (0.0182)
mailyear	1.992*** (0.396)	0.605 (0.464)	1.241*** (0.339)	0.913*** (0.309)
propresp	11.64*** (1.283)	15.67*** (1.942)	11.08*** (1.224)	14.12*** (1.170)
avggift	0.0199 (0.0176)	0.847*** (0.0753)	0.437*** (0.0198)	0.394*** (0.0327)
N	4268	4268	4268	4268

Standard errors in parentheses

\* p<.1, \*\* p<.05, \*\*\* p<.01

# Conclusions

- This presentation introduces the package `f_able`, as a post estimation command that enables `margins` to estimate marginal effects with transformed covariates
- This strategy has some limitations.
  - It can be slow
  - it may be less precise because it relies on FORCED numerical differentiation.
  - Some commands may require additional "margin" options (`nochain & numerical`) and post estimation adjustment.
- However, it can provide researchers with a simple tool to make the best of more flexible model specifications.

For more examples see the help file "ssc install f\_able"

Working paper available at: [https://bit.ly/rios\\_fable](https://bit.ly/rios_fable)

Thank you!



# References

Rios-Avila, Fernando. (forthcoming). "f\_able: Estimation of marginal effects for models with alternative variable transformations". The Stata Journal