f_able: Estimation of marginal effects with transformed covariates

Flexible model specifications: How far will margins take us

Rios-Avila, Fernando[1]

[1]friosavi@levy.org
Levy Economics Institute

Belgian Stata Conference, June 2021

Table of Contents

Introduction

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

$$E(y|x,z) = b_0 + b_1 x + b_2 z$$

- Under credible assumptions ($E(e|x,z) = 0$), this allows us to identify causal effects.
- For linear models, with no interactions or polynomials, marginal effects are equal to their coefficients:

$$\frac{dE(y|.)}{dx} = b_1 \& \frac{dE(y|.)}{dz} = b_2$$

- 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 account for the interrelationship across (constructed) variables:

$$E(y|.) = b_0 + b_1 x + b_2 x^2 + b_3 z + b_4 zx$$

$$\frac{dE(y|.)}{dx} = b_1 + 2b_2 x + b_4 z$$

$$\frac{dE(y|.)}{dz} = b_3 + b_4 x$$

- Main difference with simple linear model?
  - Marginal effects no longer constant (observed heterogeneity)
  - Coefficients alone are (often) not useful
  - Partial derivatives are needed to identify partial/marginal effects.

Estimating Marginal Effects: Non-linear model

- When the model is nonlinear, the problem is :

$$E(y|.) = G(b_0 + b_1 x + b_2 x^2 + b_3 z + b_4 zx) \rightarrow E(y|.) = G(XB)$$

$$\frac{dE(y|.)}{dx} = \frac{dG(XB)}{d(XB)} * (b_1 + 2b_2 x + b_4 z)$$

- In addition to obtaining derivatives of XB wrt x, we also need to find the derivative of $G()$ wrt XB
- $G()$ is the link function between the index XB and the outcome of interest. (Probit, fractional model, poisson, etc)

Estimating Marginal Effects

How to proceed in this case?
Marginal effects will vary with observed values. (observed heterogeneity), so what should one report?
There are many options:

$$APE = E\left(\frac{dE(y|.)}{dx}\right)$$

$$PEA = \frac{dE(y|.)}{dx}|X = \bar{x}; z = \bar{z}$$

$$PE\_at\_X = \frac{dE(y|.)}{dx}|X = X; z = Z$$

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 fines
. 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.

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 recognized they all depend on fines.
- One solution, estimate the derivatives manually, and calculate corresponding SE.
- Same as before factor notation.

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.

Storing Information: fgen and frep

- fgen and frep are wrappers around generate and replace that stores how a variable is generated, as a label or note.

```
. ssc install f_able
. qui:fgen fines2=fines^2
. describe fines2
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
fines2          double  %10.0g                fines^2

. qui:frep fines2=fines*fines
. describe fines2
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
fines2          double  %10.0g                fines*fines
```

Identifying constructed variables: f_able

- f_able is a post estimation command that is used to declare 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 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"
```

Updating Constructed Vars: f_able_p

- f_able_p is 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.
- Note: when calling margins we need to include the option nochain, so numerical derivatives are used.

```
. qui:reg citations fines fines2
. f_able 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
reg, probit, logit, qreg, etc.

* Step 3: Declare constructed variables:
f_able, nl(fx1 fx2 ... fxk)
f_able fx1 fx2 ... fxk, auto

* Step 4: Margins
margins, dydx(x1 x2 ..) [nochain numerical] [other options]

* Step 5: Additional post estimation (if no SE produced)
f_symev/f_symrv
```

f_able NEW syntax

```
* Step 1: Generate variables
fgen/frep fx1= "gen-able" function of x's
f_spline/f_rcspline Spline creation/restricted cube spline

* Step 2: Model estimation with f_reg
f_reg cmd depvar [indepvar], options

* Step 3: Margins
margins, dydx(x1 x2 ..) [other options]

* Step 4: Additional post estimation (if no SE produced)
f_symev/f_symrv
```

Example:A model of Charity

```
ssc install bcuse
bcuse charity, clear
fgen lavggift=log(avggift)
fgen lweekslast=log(weekslast)
fgen lmailsyear=log(mailsyear)
fgen lpropresp=log(propresp)

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

*OLS with LOG(Var)
f_reg reg gift resplast weekslast mailsyear propresp avggift ///
l*, robust
margins, dydx(*)  post
est sto model2
```

Example:A model of Charity

```
*Poisson with LOG(var)
f_reg poisson gift resplast weekslast mailsyear propresp avggift //
l*, robust
margins, dydx(*)  post
est sto model3

*Tobit with LOG(var)
f_reg tobit gift resplast weekslast mailsyear propresp avggift ///
l*, vce(robust) ll(0)
margins, dydx(*)  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)
```

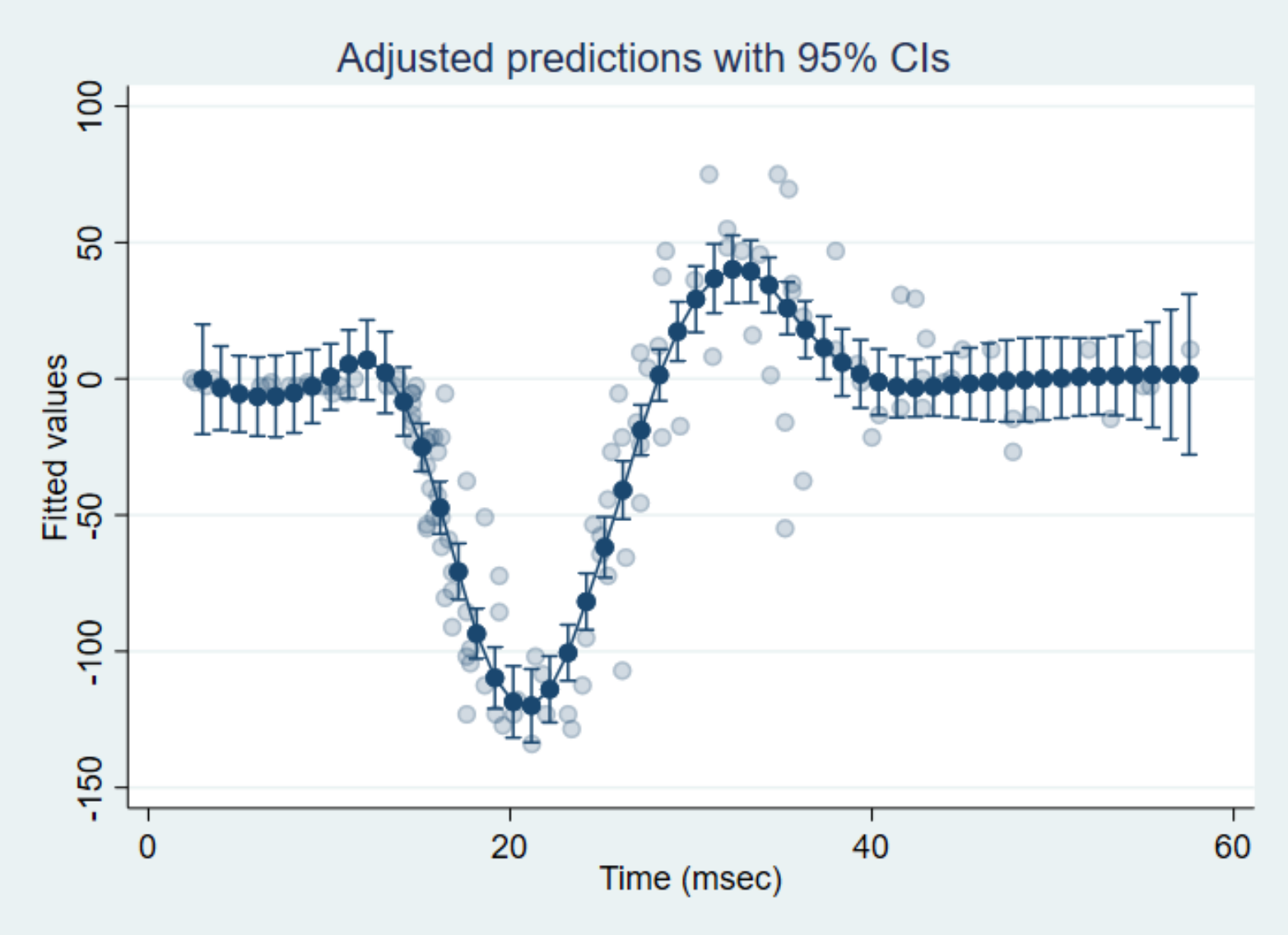
	(1) S OLS	(2) OLS w/Logs	(3) Poisson	(4) Tobit
resplast	1.514**	3.527***	2.743***	3.094***
	(0.719)	(0.990)	(0.634)	(0.605)
weekslast	-0.0186***	0.0755***	0.105***	0.0953***
	(0.00590)	(0.0212)	(0.0178)	(0.0182)
mailsyear	1.992***	0.605	1.241***	0.913***
	(0.396)	(0.464)	(0.339)	(0.309)
propresp	11.64***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

Standard errors in parentheses
* p<.1, ** p<.05, *** p<.01

Example:Motorcycle acceleration

```
webuse motorcycle,clear
f_spline spt=time, degree(3) np(7)
f_reg reg accel time spt*
margins, at(time=(3(1.01)58)) plot
```

Example:Motorcycle acceleration



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 (numerical derivatives)
  - it may be less precise because it relies on FORCED numerical differentiation.
- 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"
and get the latest files here: https://tinyurl.com/fablebe

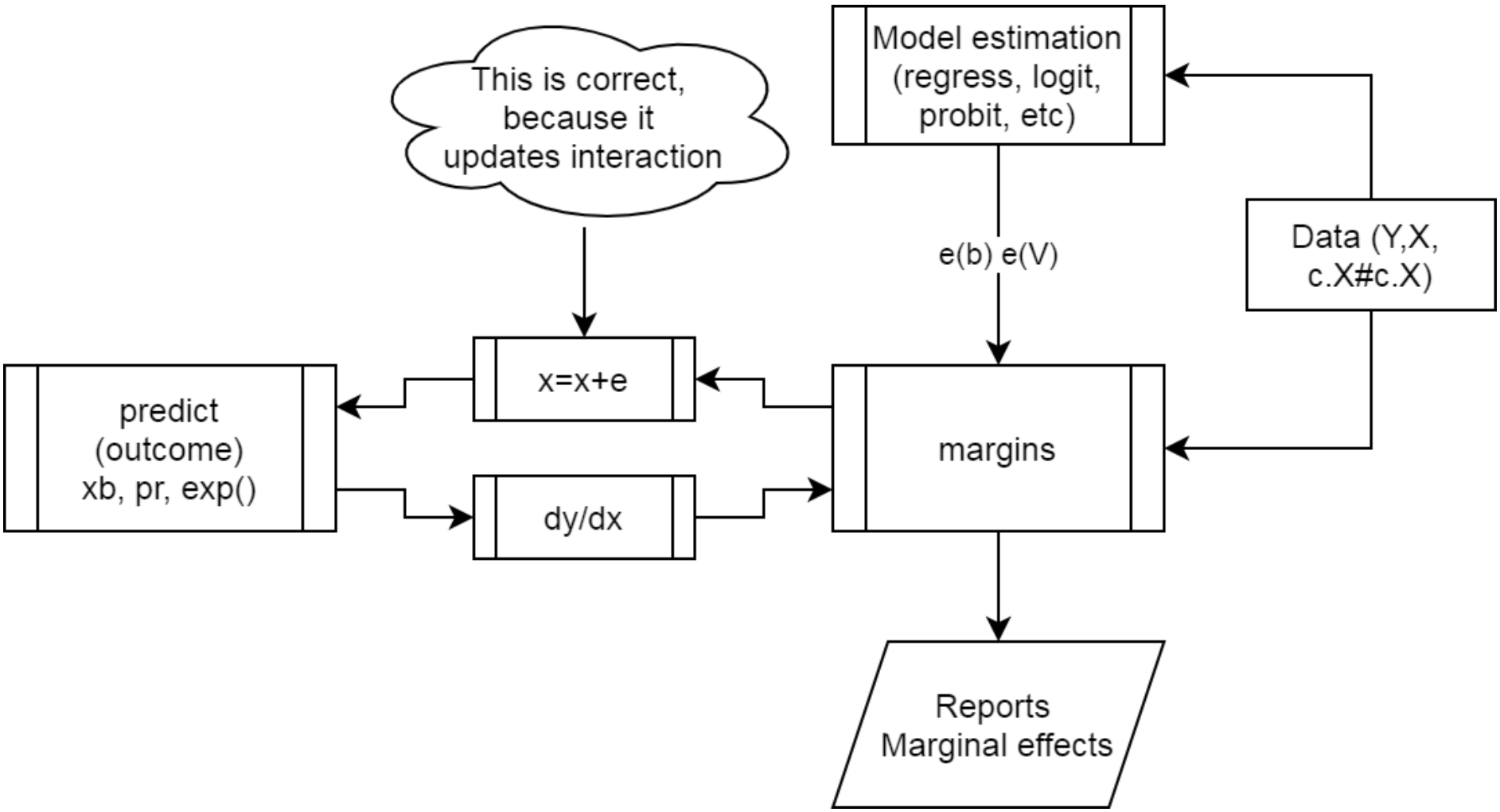
Thank you!

Questions or comments welcome

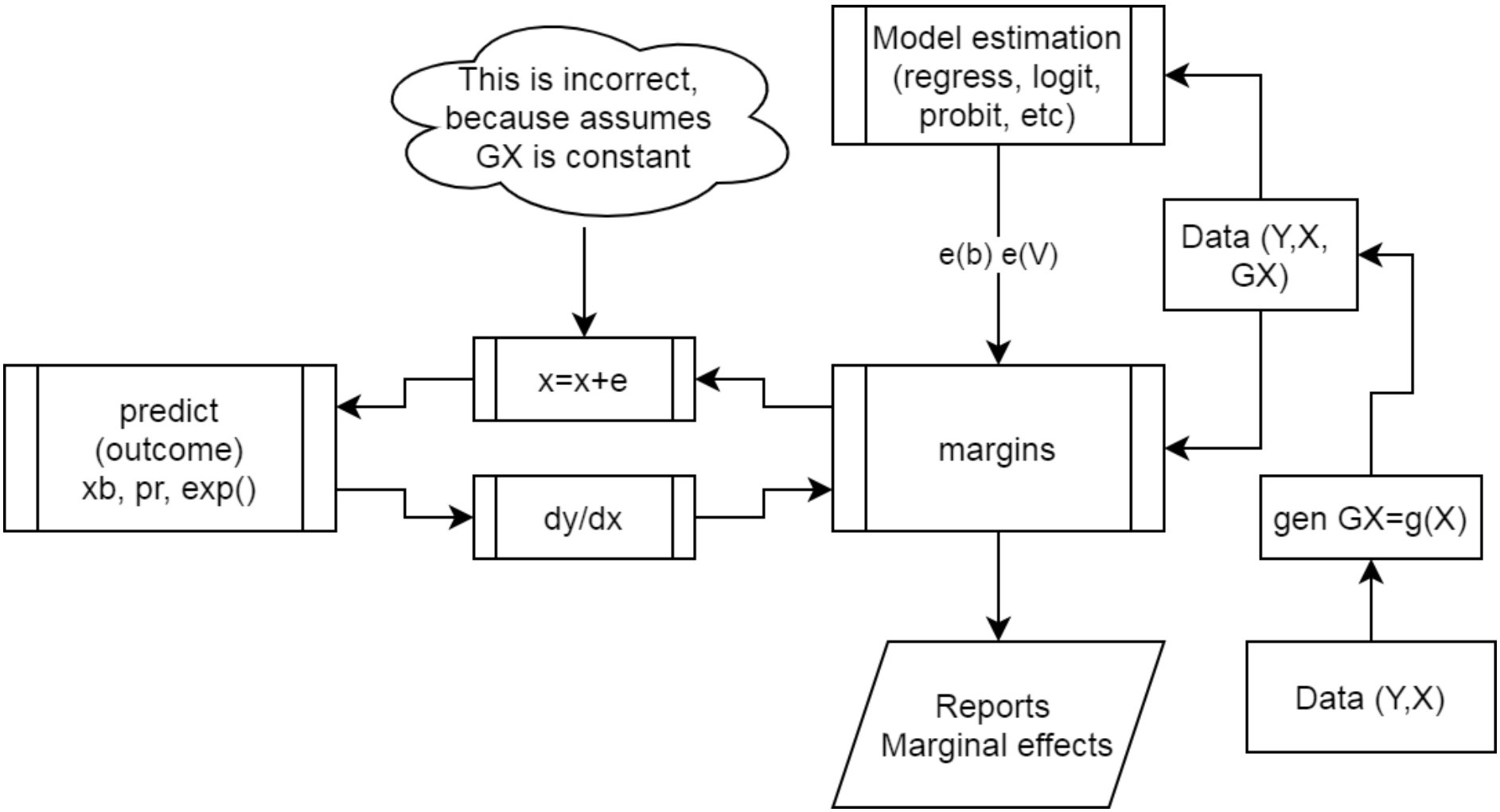
References

Rios-Avila, F. 2021. Estimation of marginal effects for models with alternative variable transformations. Stata Journal 21: 81-96.

How margins Works?



Why does it fail?



How does f_able works?