On Binary Outcome models with Heteroscedasticity

Duong Trinh

March 2024

1 Introduction

When implementing Probit Regression, I found in the solution and Stata do-file, the option is added in the command most of the time to report "robust" standard errors (SEs). From my understanding, it isn't advantageous or even being misinterpreted, in contrast to the case of a linear regression model estimated by OLS:

- For a standard linear model, when the errors are heteroscedastic, OLS coefficient estimators remain unbiased and consistent. Heteroscedasticity-robust standard errors (White's sandwich formula, produced by option in Stata) are valid whether the errors are heteroscedastic or even homoscedastic. We use them as a good practice to obtain reliable tests. This concept was introduced and emphasized in ECON4003.
- For binary dependent variable models, we assume a functional form of the likelihood and estimate using MLE. The standard Probit model relies on homoscedastic errors and if this specification is correct, all estimators for coefficients, covariance matrix, and marginal effects are consistent. In this case, using command in Stata with default standard errors is good enough, and option isn't required. In the presence of heteroscedasticity (in true DGP), the MLE estimators of the above parameters are biased and inconsistent unless the likelihood function is modified to correctly account for the form of heteroscedasticity. The use of White's robust SEs for standard Probit doesn't help address this misspecification problem, as we would be consistently estimating SEs of inconsistent parameters. Thus, it might be better to opt for command in Stata instead.
- My concern here is invoked by a short note in Cameron, A. C., & Trivedi, P. K. (2010). Microeconometrics using stata (Vol. 2). College Station, TX: Stata press., with the screenshot in attachment.

14.3.4 Robust estimate of the VCE

Binary outcome models are unusual in that there is no advantage in using the robust sandwich form for the variance–covariance matrix of the estimator (VCE) of the MLE if data are independent over i and $F(\mathbf{x}'\boldsymbol{\beta})$ is correctly specified. The reason is that the ML default standard errors are obtained by imposing the restriction $\text{Var}(y|\mathbf{x}) = F(\mathbf{x}'\boldsymbol{\beta})\{1 - F(\mathbf{x}'\boldsymbol{\beta})\}$, and this must necessarily hold because the variance of a binary variable is always $p(1-p)$; see Cameron and Trivedi (2005) for further explanation. If $F(\mathbf{x}'\boldsymbol{\beta})$ is correctly specified, the `vce(robust)` option is not required. Hence, we may infer a misspecified functional form $F(\mathbf{x}'\boldsymbol{\beta})$ if the use of the `vce(robust)` option produces substantially different variances from the default.

At the same time, dependence between observations may arise because of cluster sampling. In that case, the appropriate option is to use `vce(cluster` *clustvar*`)`.

- Inspired by simulation exercise by Enrique Pinzon, I conduct extra simulations to justify these ideas.

2 Simulation results

DGP 1: Probit & Homokedasticity

Statistic	Probit	Probit,robust	Hetprobit	LMP	LMP,robust	Approx True Value
AME of x1	-0.138	-0.138	-0.138	-0.119	-0.119	-0.14
5% Rejection Rate x1	0.049	0.052	0.047	1.000	1.000	NA
ATE of x21	-0.152	-0.152	-0.152	-0.152	-0.152	-0.15
5% Rejection Rate x21	0.057	0.062	0.067	0.055	0.058	NA
ATE of x22	0.129	0.129	0.129	0.129	0.129	0.13
5% Rejection Rate x22	0.057	0.061	0.065	0.050	0.059	NA
ATE of x23	-0.152	-0.152	-0.152	-0.152	-0.152	-0.15
5% Rejection Rate x23	0.058	0.060	0.067	0.054	0.058	NA
AME of x3	0.138	0.138	0.138	0.089	0.089	0.14
5% Rejection Rate x3	0.052	0.054	0.053	1.000	1.000	NA

DGP 2: Probit & Heterokedasticity

Statistic	Probit	Probit,robust	Hetprobit	LMP	LMP,robust	Approx True Value
AME of x1	-0.099	-0.099	-0.210	-0.101	-0.101	-0.210
5% Rejection Rate x1	1.000	1.000	0.052	1.000	1.000	NA
ATE of x21	-0.193	-0.193	-0.191	-0.192	-0.192	-0.190
5% Rejection Rate x21	0.062	0.060	0.064	0.044	0.058	NA
ATE of x22	0.077	0.077	0.081	0.080	0.080	0.082
5% Rejection Rate x22	0.063	0.061	0.065	0.044	0.060	NA
ATE of x23	-0.192	-0.192	-0.191	-0.192	-0.192	-0.190
5% Rejection Rate x23	0.063	0.058	0.063	0.046	0.060	NA
AME of x3	0.061	0.061	0.166	0.059	0.059	0.166
5% Rejection Rate x3	1.000	1.000	0.062	1.000	1.000	NA

DGP 3: Linear model & Heterokedasticity

Metric	1	2
b1	5.214e-01	5.214e-01
b2	-4.897e-01	-4.897e-01
b3	-5.256e-01	-5.256e-01
b4	4.841e-01	4.841e-01
b5	-5.227e-01	-5.227e-01
b6	5.116e-01	5.116e-01
bias1	2.135e-02	2.135e-02
bias2	1.026e-02	1.026e-02
bias3	-2.557e-02	-2.557e-02
bias4	-5.159e-01	-5.159e-01
bias5	9.773e-01	9.773e-01
bias6	1.159e-02	1.159e-02
hatmax	2.475e-02	2.475e-02
hatmin	6.000e-04	6.000e-04
N	1.000e+04	1.000e+04
r_b1	3.800e-02	4.600e-02
r_b2	7.150e-01	3.750e-02
r_b3	3.650e-02	4.400e-02
r_b4	1.265e-01	5.290e-01
r_b5	5.030e-01	8.480e-01
r_b6	7.205e-01	7.205e-01
rc	0.000e+00	0.000e+00
rep	1.000e+03	1.000e+03
se_b1	6.539e-01	3.395e-01
se_b2	3.111e-02	2.052e-01
se_b3	6.680e-01	3.543e-01
se_b4	6.574e-01	3.515e-01
se_b5	6.571e-01	3.431e-01
se_b6	3.111e-02	1.926e-01

3 Discussion

- [S] Having said that, I am not convinced that heteroskedastic probit will solve model mis-specification issues. The variance to an extent but mean specification remains. But, yes it is preferrable to standard probit regression.

4 Codes

```
clear all
local L = 10000000
local R = 200
local N = 10000
set seed 222
```

```
program define mkdata
        syntax, [n(integer 10000)]
        clear
        quietly set obs `n'
        generate x1    = rchi2(1)-1
        generate x2    = int(4*rbeta(5,2))
        generate x3    = rchi2(1)-1
        generate sg    = exp(.3*(x1 -1.x2 + 2.x2 - 3.x2 + x3))
        generate e     = rnormal(0 , sg)
        generate xb    = .5*(1 - x1 - 1.x2 + 2.x2 - 3.x2 + x3)
        generate y     =  xb + e > 0

        generate m1  = normalden(xb/sg)*((-.5 -.3*xb)/sg)
        generate m3  = normalden(xb/sg)*((.5 -.3*xb)/sg)
        generate m21 = normal(.5*(-x1 + x3)/exp(.3*(x1 -1 + x3)))       ///
                   - normal(.5*(1 -x1 + x3)/exp(.3*(x1 + x3)))
        generate m22 = normal(.5*(2-x1 + x3)/exp(.3*(x1 + 1 + x3)))     ///
                                  -normal(.5*(1 -x1 + x3)/exp(.3*(x1+ x3)))
        generate m23 = m21
end

mkdata, n(`L')
summarize m1, meanonly
local m1  = r(mean)
summarize m3, meanonly
local m3  = r(mean)
summarize m21, meanonly
local m21 = r(mean)
summarize m22, meanonly
local m22 = r(mean)
summarize m23, meanonly
local m23 = r(mean)

display `m1'
display `m3'
display `m21'
display `m22'
display `m23'

postfile sims est hm1 hm1_r hm21 hm21_r hm22 hm22_r hm23 hm23_r hm3 hm3_r  ///
         rc cv using hetprobit, replace

forvalues i=1/`R' {
        quietly {
                mkdata, n(`N')
                capture probit y x1 i.x2 x3, iterate(200)
                local rc = _rc
```

```
        local cv = e(converged)
        if (`rc' | `cv'==0){
                local hm1    = .
                local hm1_r  = .
                local hm21   = .
                local hm21_r = .
                local hm22   = .
                local hm22_r = .
                local hm23   = .
                local hm23_r = .
                local hm3    = .
                local hm3_r  = .
        }
        else {
                margins, dydx(*) post
                local hm1 = _b[x1]
                test _b[x1] = `m1'
                local hm1_r   = (r(p)<.05)
                local hm21 = _b[1.x2]
                test _b[1.x2] = `m21'
                local hm21_r   = (r(p)<.05)
                local hm22 = _b[2.x2]
                test _b[2.x2] = `m22'
                local hm22_r   = (r(p)<.05)
                local hm23 = _b[3.x2]
                test _b[3.x2] = `m23'
                local hm23_r   = (r(p)<.05)
                local hm3 = _b[x3]
                test _b[x3] = `m3'
                local hm3_r   = (r(p)<.05)
        }
        post sims (1) (`hm1') (`hm1_r') (`hm21') (`hm21_r')       ///
                  (`hm22') (`hm22_r') (`hm23') (`hm23_r') (`hm3') ///
                  (`hm3_r') (`rc') (`cv')

        capture probit y x1 i.x2 x3, vce(robust) iterate(200)
        local rc = _rc
        local cv = e(converged)
        if (`rc' | `cv'==0){
                local hm1    = .
                local hm1_r  = .
                local hm21   = .
                local hm21_r = .
                local hm22   = .
                local hm22_r = .
                local hm23   = .
                local hm23_r = .
                local hm3    = .
                local hm3_r  = .
```

```
}
else {
        margins, dydx(*) post
        local hm1 = _b[x1]
        test _b[x1] = `m1'
        local hm1_r   = (r(p)<.05)
        local hm21 = _b[1.x2]
        test _b[1.x2] = `m21'
        local hm21_r   = (r(p)<.05)
        local hm22 = _b[2.x2]
        test _b[2.x2] = `m22'
        local hm22_r   = (r(p)<.05)
        local hm23 = _b[3.x2]
        test _b[3.x2] = `m23'
        local hm23_r   = (r(p)<.05)
        local hm3 = _b[x3]
        test _b[x3] = `m3'
        local hm3_r   = (r(p)<.05)
}
post sims (2) (`hm1') (`hm1_r') (`hm21') (`hm21_r')       ///
          (`hm22') (`hm22_r') (`hm23') (`hm23_r') (`hm3') ///
          (`hm3_r') (`rc') (`cv')

capture hetprobit y x1 i.x2 x3, het(x1 i.x2 x3) iterate(200)
local rc = _rc
local cv = e(converged)
if (`rc' | `cv'==0) {
        local hm1    = .
        local hm1_r  = .
        local hm21   = .
        local hm21_r = .
        local hm22   = .
        local hm22_r = .
        local hm23   = .
        local hm23_r = .
        local hm3    = .
        local hm3_r  = .
}
else {
        margins, dydx(*) post
        local hm1 = _b[x1]
        test _b[x1] = `m1'
        local hm1_r   = (r(p)<.05)
        local hm21 = _b[1.x2]
        test _b[1.x2] = `m21'
        local hm21_r   = (r(p)<.05)
        local hm22 = _b[2.x2]
        test _b[2.x2] = `m22'
        local hm22_r   = (r(p)<.05)
```

```
        local hm23 = _b[3.x2]
        test _b[3.x2] = `m23'
        local hm23_r   = (r(p)<.05)
        local hm3 = _b[x3]
        test _b[x3] = `m3'
        local hm3_r   = (r(p)<.05)
}
post sims (3) (`hm1') (`hm1_r') (`hm21') (`hm21_r')       ///
          (`hm22') (`hm22_r') (`hm23') (`hm23_r') (`hm3') ///
          (`hm3_r') (`rc') (`cv')

capture regress y x1 i.x2 x3
margins, dydx(*) post
local hm1 = _b[x1]
test _b[x1] = `m1'
local hm1_r   = (r(p)<.05)
local hm21 = _b[1.x2]
test _b[1.x2] = `m21'
local hm21_r   = (r(p)<.05)
local hm22 = _b[2.x2]
test _b[2.x2] = `m22'
local hm22_r   = (r(p)<.05)
local hm23 = _b[3.x2]
test _b[3.x2] = `m23'
local hm23_r   = (r(p)<.05)
local hm3 = _b[x3]
test _b[x3] = `m3'
local hm3_r   = (r(p)<.05)
post sims (4) (`hm1') (`hm1_r') (`hm21') (`hm21_r')       ///
          (`hm22') (`hm22_r') (`hm23') (`hm23_r') (`hm3') ///
          (`hm3_r') (`rc') (`cv')

capture regress y x1 i.x2 x3, vce(robust)
margins, dydx(*) post
local hm1 = _b[x1]
test _b[x1] = `m1'
local hm1_r   = (r(p)<.05)
local hm21 = _b[1.x2]
test _b[1.x2] = `m21'
local hm21_r   = (r(p)<.05)
local hm22 = _b[2.x2]
test _b[2.x2] = `m22'
local hm22_r   = (r(p)<.05)
local hm23 = _b[3.x2]
test _b[3.x2] = `m23'
local hm23_r   = (r(p)<.05)
local hm3 = _b[x3]
test _b[x3] = `m3'
local hm3_r   = (r(p)<.05)
```

```
                post sims (5) (`hm1') (`hm1_r') (`hm21') (`hm21_r')       ///
                          (`hm22') (`hm22_r') (`hm23') (`hm23_r') (`hm3') ///
                          (`hm3_r') (`rc') (`cv')
        }
        if (`i'/50) == int(`i'/50) {
        di ".                  `i'"
    }
    else {
        di _c "."
    }
}
postclose sims
use hetprobit, clear
label define est 1 "probit" 2 "probit-robust" 3 "hetprobit" 4 "LPM" 5 "LPM-robust"
label values est est
bysort est: summarize
```