1 Estimating Treatment Effect Stochastic Frontier Models of Chen et al. (2020, JBES) using Stata

by Hung-Jen Wang, [wangh@ntu.edu.tw (mailto:wangh@ntu.edu.tw)](mailto:wangh@ntu.edu.tw), June 2022

1.1 What is it

This Jupyter notebook shows how to estimate the treatment effect stochastic frontier model of Chen et al. (2020) using Stata. The notebook mixes together the text of the paper and the associated Stata code, making the code easier to understand and to follow.

- Chen, Y.T., Hsu, Y.C. and Wang, H.J. (2020) "A Stochastic Frontier Model with Endogenous Treatment Status and Mediator," *Journal of Business and Economic Statistics*, 38, pp.243-256.

1.2 How to use it

- Method 1: You may read the notebook (or the PDF version of it), and copy-and-paste the code to a Stata .do file and work on the example from there.
- Method 2: You may run the Stata code directly and interactively in the notebook. For this purpose, you need to install Jupyter Notebook and the `Stata_kernel` following [the instructions here (https://kylebarron.dev/stata_kernel/)](https://kylebarron.dev/stata_kernel/).

1.3 Notes

- The code works with Stata versions 13, 14, 15, 16, and 17.
- The illustration is a simplified version of the empirical model presented in Chen et al. (2020).
  - The simplified version adopts fewer explanatory variables in some of the equations. Otherwise, it has all the characteristics of the original model and the estimation would follow the exact same steps.
  - Chen et al. (2020) estimates the full, original model using Stata 13.1 (works also with Stata 14). Unfortunately, for unknown reasons the original model does not converge smoothly (in the `gmm` step) using Stata 17. The simplified example avoids the problem, although results from different versions of Stata are still slightly different. The differences arise from Stata's `gmm` estimation command.
- For a replication of the result in Chen et al. (2020), see JBES2020_CHW_Table46.zip from my web page.

2 Program Start

In [ ]:

```
myclear
set more off
set trace off
capture log close
```

In [ ]:

```
use data4a, clear

set seed 1256

global xvar  lfertvalue lgca pc1     /* frontier vars for D=1 */
global xvarB lfertvalueB lgcaB pc1B  /* frontier vars for D=0; same as $xvar with different
global zvar  rain2                   /* ineff var for D=1 */
global zvarB rain2B                  /* ineff var for D=0, same as $zvar with different nam
global z2_var lriverkm  /* log of river length based on which z2 is constructed */
global Z_predictors dss_riverkm dss_km2   /* other vars; from which we use PCA to extract i
```

2.1 constructing Z_2, the continuous IV for M

> **(Section 2.5 The Dam Example: Variables and Effects**, *on the construction of* Z_2 **)**
>
> > The mediator M may also be endogenous in the dam example because the irrigation construction could also be confounded by the unexplained conditions of agricultural production. The endogeneity of M requires an additional continuous IV, which is denoted by Z_2, for parameter identification. All other things being equal, districts with larger river distributary systems, such as river basins (and hence longer total river lengths), are advantageous in building efficient irrigation systems. We set Z_2 to be the ordinary least squares (OLS) residual obtained by regressing the log of the district's river length on Z_1 and X to account for this consideration and to satisfy certain identification assumptions [...].
>
> **(Table 2)**
>
> > river length (for the geographical suitability of irrigation): the OLS residual obtained by regressing the log of the total river length on Z_1 and X for each district.

In [ ]:

```
**** Constructing Z_2, the continuous IV for M ****

   /* principal component */
quie pca  $Z_predictors, components(2)

global Z_predictors_pca
forvalues i = 1/2 {
   capture  drop pc`i'  /* drop the dummies that were created earlier */
   global Z_predictors_pca $Z_predictors_pca pc`i'  /* append the names */
}

predict double $Z_predictors_pca

 /* adjust lriverkm so that it satisfies the model's assumptions */
capture drop e
reg $z2_var Z1_inst $xvar $zvar $Z_predictors_pca  /* We set Z2 to be the ordinary least sq
                                                    /* regressing the log of the district's
                                                    /* account for this consideration and t
predict double e, resid
gen double Z2_inst = e
```

2.2 bivariate probit model of $P(M, D|Z_1, Z_2, X, \eta)$ in (4.11)*

> (**Section 4.2 Model Specification and Estimation,** *on the bivariate probit model of* $P(M, D|Z_1, Z_2, X, \eta)$ *in (4.11)*)
>
> > In addition, we specify $P(M, D|Z_1, Z_2, X, \eta)$ as a bivariate probit model and set the distribution $F_{U_M, U_D}(\cdot, \cdot, \rho_{md})$ as a bivariate normal distribution:
> >
> > $$\begin{bmatrix} U_M \\ U_D \end{bmatrix} \Bigg| (Z_1, Z_2, X) \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho_{md} \\ \rho_{md} & 1 \end{bmatrix} \right). \tag{4.11}$$
> >
> > As in Wooldridge (2010, p.596), this specification is useful because the maximum likelihood estimator (MLE) of η can be computed using an econometrics package that permits the ML estimation for a bivariate probit model. Let $\Phi(\cdot)$ and $\phi(\cdot)$ be respectively the distribution function and the PDF of $N(0, 1)$.

In [ ]:

```
*** bivariate probit model of P(M,D|Z1, Z2; X, \eta) **

quie sum Mvar
global nofobs = r(N)

biprobit (eq1: Mvar = Dvar Z2_inst $xvar $zvar  $Z_predictors_pca  ) (eq2: Dvar = Z1_inst $

mat bp0 = e(b)
mat bp0Mz2= bp0[1,"eq1:Z2_inst"]
scalar alpha_z = bp0Mz2[1,1] /* coefficient of alpha_z */

mat bp0Dz1 = bp0[1, "eq2:Z1_inst"]
  scalar gamma_z = bp0Dz1[1,1]  /* coefficient of gamma_z */
  if scalar(gamma_z) < 0 {
    di in red "Gamma_z cannot be <0."
    aaaaa
  }

mat bp0Md  = bp0[1,"eq1:Dvar"]
  scalar alpha_d = bp0Md[1,1]  /* coefficient of alpha_d */

mat bp0rho = bp0[1, "/athrho"]

scalar rho_e = e(rho)  /* rho_md? */



*** compute the first term on the RHS of Psi() in (4.12) for later use ****

predict double tem1, eq(eq2) xb
gen double Xgammax = tem1 - scalar(gamma_z)*Z1_inst

predict double tem2, eq(eq1) xb
gen double Xalphax = tem2 - scalar(alpha_d)*Dvar - scalar(alpha_z)*Z2_inst

drop tem1 tem2

gen double Psi_comp1 = 1/(normal(scalar(gamma_z) + Xgammax) - normal(Xgammax))  /* 1st term
```

2.3 Q_D and the Q_{z1} functions in (4.13) and (4.14)

> (**Section 4.2 Model Specification and Estimation,** *on the Q_D and the Q_{z1} functions in (4.13) and (4.14)*)
>
> > Furthermore, it implies the following specification of (3.2):
> >
> > $$Q_D(\gamma_{z1} Z_1 + X^T \gamma_x) = \Phi(\gamma_{z1} Z_1 + X^T \gamma_x). \qquad (4.13)$$
> >
> > Correspondingly, we specify the instrument propensity score model in (3.3) as a probit model:
> >
> > $$Q_{Z_1}(X^T \alpha_{z1}) = \Phi(X^T \alpha_{z1}). \qquad (4.14)$$

In [ ]:

```
** Q_D, Q_z1 of (4.13), (4.14) **

  quie probit Z1_inst $xvar $zvar  $Z_predictors_pca  /* Q_D of (4.13) */
  quie predict double Q_z1 /* Q_{z1} of (4.14) */

  local b_low = 0.1
  local b_up  = 1-0.1

  quie replace Q_z1 = cond(Q_z1<0.1, 0.1, cond(Q_z1>0.9, 0.9, Q_z1))  /* adjust for extreme
```

2.4 the mixture function of $f_{z2}(z_2, \alpha_f)$ in (4.15)

> (**Section 4.2 Model Specification and Estimation,** *on the mixture function of* $f_{z2}(z_2, \alpha_f)$ *in (4.15)*)
>
> > In this empirical example, the histogram of Z_2 indicates a bimodal distribution, so we use a mixture distribution of two normals to approximate the distribution and set
> >
> > $$f_{Z_2}(z_2, \alpha_f) = \frac{\alpha_{f(p)}}{\alpha_{f(s1)}}\phi\left(\frac{z_2 - \alpha_{f(m1)}}{\alpha_{f(s1)}}\right) + \frac{1-\alpha_{f(p)}}{\alpha_{f(s2)}}\phi\left(\frac{z_2 - \alpha_{f(m2)}}{\alpha_{f(s2)}}\right), \quad (4.15)$$
> >
> > where $(\alpha_{f(m1)},\ \alpha_{f(s1)})$ and $(\alpha_{f(m2)},\ \alpha_{f(s2)})$ are respectively the means and the standard deviations of the two normal distributions, and $\alpha_{f(p)}$ and $(1 - \alpha_{f(p)})$ are the distributions' weights.

In [ ]:

```
** f_{z2} of (4.15), a mixture model **

gen byte m1 = 1
gen byte m2 = 1
            /* estimating parameters of the mixed distribution */
ml model lf mixnormal (Z2_inst = m1, nocons) (Z2_inst= m2, nocons)  (p1: ) (s1: ) (s2: )  /
ml init   -1.368926  1.109845  -.2098054  -.0035635  .0067127, copy
ml max, diff
mat mixb0 = e(b)

scalar _m1 = [eq1]_b[m1]  /* mean of the first distribution */
scalar _m2 = [eq2]_b[m2]  /* mean of the 2nd distribution */
scalar _pb1  = exp([p1]_b[_cons])/(1 + exp([p1]_b[_cons]))  /* the probability */
scalar _sig1 = exp([s1]_b[_cons])  /* std dev of the 1st distribution */
scalar _sig2 = exp([s2]_b[_cons])  /* std dev of the 2nd distribution */
```

2.5 $\Psi_{d'}(Z_2, X, \eta)$ in (4.12), $m_{d'}(X, \alpha_m)$ in (3.4)

(Section 4.2 Model Specification and Estimation, *on $\Psi_{d'}(Z_2, X, \eta)$ in (4.12)* **)**

> In the supplementary appendix, we further show that this bivariate probit model implies the following formula of (3.5):
>
> $$\Psi_{d'}(Z_2, X, \eta) = \frac{1}{\Phi(\gamma_{z1} + X^T\gamma_x) - \Phi(X^T\gamma_x)} \times \int_{X^T\gamma_x}^{\gamma_{z1}+X^T\gamma_x} \Phi\left(\frac{\alpha_d d' + \alpha_z Z_2 + X^T\alpha_x + \rho_{md}u}{\sqrt{1-\rho_{md}^2}}\right)\phi(u)du. \tag{4.12}$$

(Section 3.1 Structural Model, *on $m_{d'}(X, \alpha_m)$ in (3.4)* **)**

> Given $P(M, D|Z_1, Z_2, X, \eta)$ and $f_{Z2}(\cdot, \alpha_f)$, we show that under suitable assumptions, the potential-mediator model in (2.16) can be expressed as
>
> $$m_{d'}(X, \alpha_m) \equiv \int_R \Psi_{d'}(z_2, X, \eta) f_{Z2}(z_2, \alpha_f)\mathrm{d}z_2, \tag{3.4}$$
>
> where $\alpha_m \equiv (\eta^T, \alpha_f^T)^T$.

(Section 4.2 Model Specification and Estimation, *on potential mediator function $E[M(d')|X, C] = m_{d'}(X, \alpha_m)$ in (3.4)* **)**

> Accordingly, the potential mediator model $m_{d'}(\cdot)$ in (3.4) is specified using (4.12) and (4.15). The integration in (3.4) is numerically computed using the Gauss-Hermite quadrature with 60 quadrature points.

In [ ]:

```
***  Psi(Z2,X,eta) and m(X,alpha_m)=int(Psi()*f_z2)******


  /* 1st step: The first term on the RHS of Psi() in (4.12). It does not involove z2 and u

    ** the result, which is Psi_comp1, has been computed previously

  /* 2nd step: double integration (in Psi and over Psi)  */

capture drop double_int_d0
capture drop double_int_d1
quie gen double double_int_d0 = .
quie gen double double_int_d1 = .

    /* this box is for examiniation */
    local hasdata = 0
    sort code61

    /* ************************** */

quie sum code61, meanonly
local nofobs = r(N)

  forvalues i = 1/`nofobs' {
     scalar bb1 = Xalphax[`i']
     scalar mylb = Xgammax[`i']
     scalar myub = scalar(gamma_z) + scalar(mylb)
     scalar zeroc = 0
       /* for m_{d'=1} */
     quie myint5a_jup, alphadd(alpha_d) alphaz(alpha_z) xalphax(bb1) rho(rho_e) mylow(mylb)
            quie replace double_int_d1 = my_res if _n == `i'
       /* for m_{d'=0} */
     quie myint5a_jup, alphadd(zeroc) alphaz(alpha_z) xalphax(bb1) rho(rho_e) mylow(mylb) m
            quie replace double_int_d0 = my_res if _n == `i'
  }

     ***** now, potential mediator function E[M(d')|X,C]=m_{d'}(X,α_m) in (3.4) ******

quie gen double m_d0 = Psi_comp1*double_int_d0 /* for m_{d'=0}  */
quie gen double m_d1 = Psi_comp1*double_int_d1 /* for m_{d'=1}  */

sort code61
```

2.6 calculating the weights in (3.7)

(Section 3.2 Identification of Parameters, *on the weights of the estimator* **)**

Specifically, β_d can be identified as the minimizer of the weighted MSE of actual outputs: for $d \in \{0, 1\}$,

$$\beta_d \equiv \arg\min_{b_d \in \mathcal{B}_d} \sum_{d'=0,1} E\Big[w(d, d', \alpha_w)\Big(Y - h_{d'}(X, \alpha_m, b^h_{d1}, b^h_{d0}) + g_{d'}(X, \alpha_m, b^g_{d1},$$

where the weights $w(1, 1, \alpha_w)$, $w(1, 0, \alpha_w)$, $w(0, 1, \alpha_w)$ and $w(0, 0, \alpha_w)$ are functions of $(D, Z_1, Z_2, X^T)^T$:

$$w(d, d', \alpha_w) \equiv \begin{cases} D\lambda(Z_1, X, \alpha_{z1}), & (d, d') = (1, 1), \\ D\varphi_1(Z_2, \alpha_\varphi)\lambda(Z_1, X, \alpha_{z1}), & (d, d') = (1, 0), \\ (D - 1)\varphi_2(Z_2, \alpha_\varphi)\lambda(Z_1, X, \alpha_{z1}), & (d, d') = (0, 1), \\ (D - 1)\lambda(Z_1, X, \alpha_{z1}), & (d, d') = (0, 0), \end{cases} \tag{3.7}$$

$$\lambda(Z_1, X, \alpha_{z1}) \equiv \frac{Z_1 - Q_{Z1}(X^T\alpha_{z1})}{Q_{Z1}(X^T\alpha_{z1})(1 - Q_{Z1}(X^T\alpha_{z1}))}, \tag{JN1}$$

$$\varphi_1(Z_2, \alpha_\varphi) \equiv \frac{f_{Z2}(Z_2 + \alpha_d/\alpha_{z2}, \alpha_f)}{f_{Z2}(Z_2, \alpha_f)}, \tag{JN2}$$

and

$$\varphi_2(Z_2, \alpha_\varphi) \equiv \frac{f_{Z2}(Z_2 - \alpha_d/\alpha_{z2}, \alpha_f)}{f_{Z2}(Z_2, \alpha_f)}, \tag{JN3}$$

with the identified parameter vectors $\alpha_w \equiv (\alpha_{z1}^T, \alpha_\varphi^T)^T$ and $\alpha_\varphi \equiv (\alpha_d, \alpha_{z2}, \alpha_f^T)^T$.

In [ ]:

```
*** (JN1), (JN2), (JN3), and the weight functions ***

  quie gen double lambda = (Z1_inst - Q_z1)/(Q_z1*(1-Q_z1))  /* (JN1) */

  quie gen double f_z2_b = scalar(_pb1)*normalden(Z2_inst, scalar(_m1), scalar(_sig1)) + /*
      */               (1-scalar(_pb1))*normalden(Z2_inst, scalar(_m2), scalar(_sig2))
  quie gen double f_z2_a = scalar(_pb1)*normalden(Z2_inst + scalar(alpha_d)/scalar(alpha_z)
      */               (1-scalar(_pb1))*normalden(Z2_inst + scalar(alpha_d)/scalar(alpha_z)
  quie gen double f_z2_c = scalar(_pb1)*normalden(Z2_inst - scalar(alpha_d)/scalar(alpha_z)
      */               (1-scalar(_pb1))*normalden(Z2_inst - scalar(alpha_d)/scalar(alpha_z)

quie gen double phi1 = f_z2_a/f_z2_b    /* (JN2) */
quie gen double phi2 = f_z2_c/f_z2_b    /* (JN3) */

           /* smooth the probability */
quie sum phi2 if phi2 <=15
quie replace phi2 = r(max) if phi2 > 15  /* 1 of the observations has extremely large value


*** The wegith function in (3.7) ***

quie gen double W11 = Dvar*lambda
quie gen double W10 = Dvar*phi1*lambda
quie gen double W01 = (Dvar-1)*phi2*lambda
quie gen double W00 = (Dvar-1)*lambda
```

In [ ]:

```
****** Prepare for Estimation **********

sort code61

keep code61 lprodmain $xvar $zvar Dvar Mvar Z1_inst Z2_inst $Z_predictors $Z_predictors_pca

quie gen myid = _n /* for sorting purpose */

capture drop mysample
capture drop wght

capture quie replace frann2 = frann2*10

mat b0_gmm1 = 0
mat b0_gmm2 = 0

capture drop hasmiss
findmis lprodmain $xvar $zvar W11 W10 W01 W00 Dvar m_d1 m_d0, gen(hasmiss)

foreach X in lprodmain $xvar $zvar {
 di " "
 di "`X'"
 ttest `X' if hasmiss == 0, by(Dvar) unpaired
}
```

2.7 estimating model parameters using (3.6)

(Section 4.2 Model Specification and Estimation, *on estimating model parameters using (3.6)* **)**

> Given $h(X, \beta_{dj}^{h})$, $g(X, \beta_{dj}^{g})$ and $w(d, d', \hat{\alpha}_w)$, based on (3.7) with (4.14) and (4.15), we further estimate the parameter vector β_d using a second-stage weighted non-linear least squares estimator (WNLSE) $\hat{\beta}_d = (\hat{\beta}_{d1}^{hT}, \hat{\beta}_{d0}^{hT}, \hat{\beta}_{d1}^{gT}, \hat{\beta}_{d0}^{gT})^T$ based on (3.6), for $d \in \{0, 1\}$.

(Section 3.2 Identification of Parameters)

$$\beta_d \equiv \arg\min_{b_d \in \mathcal{B}_d} \sum_{d'=0,1} E\Bigl[w(d, d', \alpha_w)\Bigl(Y - h_{d'}(X, \alpha_m, b_{d1}^{h}, b_{d0}^{h}) + g_{d'}(X, \alpha_m, b_{d1}^{g},$$

In [ ]:

```
***********************************
*** Estimate models with D=1 ******
***********************************

mat myini1 = (  .461449,  1.39053,  .3040748,  -7.853099,   /*
   */  .2839695,  .9664387,  .2020703,  -.8084254,  4.053684,   /*
   */  -9.814968,  13.1198,  -5.644003)

  scalar getout = 0

  forvalues i = 1/2 { /* run twice, the 2nd time using the first time's result as init. So

     if getout == 1 {
        exit
     }

     if `i' == 2 {
        mat myini1 = b0_gmm1  /* 2nd time, using results from model in i=1 */
     }

        /* the number of equations (nequations) equals the sum of the numbers of var in $xv
       gmm sftreat_GMM4_jup if Dvar == 1, nequations(12) parameters($xvar b0 $xvarB b0B $zv
           mylhs(lprodmain) myxvar($xvar) myzvar($zvar) mywght(W11) mywghtB(W10) ///
           myp(m_d1) mypB(m_d0) ///
           winitial(identity) ///
           from(myini1)     ///
           onestep


       **** further checking ***********

          local missingv = 0 /* flag */
          if _rc == 0 {
            mat checkv = e(V) /* V-COV matrix */
            local vsize = colsof(checkv)
            forvalues i = 1/`vsize' {
              local checkstd = checkv[`i', `i']
              if `checkstd' == 0 | `checkstd' > 2e+7 {
                 local missingv = 1
              }
            }
          }

       ************************

  if _rc == 0 & `missingv' == 0 {
    mat b0_gmm1 = e(b)
  }
  else {
    scalar getout = 1
    exit
  }

   if getout ==1 {
    exit
      }

}
```

```
  if getout ==1 {
   exit
   }


***************


  *** obtain H_11, G_11 etc., for LATE ****


    /* begin with the H_11 */

  foreach X in hb_11 hb_10 H_11 {
    capture drop `X'
  }

 quie gen double hb_11 = 0 if hasmiss ==0
 quie gen double hb_10 = 0 if hasmiss ==0

 foreach X in $xvar b0 {
   mat aa_11 = b0_gmm1[1, "`X':_cons"]
   scalar aa_11 = aa_11[1,1]
   mat aa_10 = b0_gmm1[1, "`X'B:_cons"]
   scalar aa_10 = aa_10[1,1]

   if "`X'" ~= "b0" {
     quie replace hb_11 = hb_11 + scalar(aa_11)*`X' if hasmiss ==0
     quie replace hb_10 = hb_10 + scalar(aa_10)*`X' if hasmiss ==0
   }
   else {
     quie replace hb_11 = hb_11 + scalar(aa_11) if hasmiss ==0
     quie replace hb_10 = hb_10 + scalar(aa_10) if hasmiss ==0
   }
 }
    quie gen double H_11 = (m_d1)*hb_11 + (1-m_d1)*hb_10 if hasmiss ==0


    /* Now G_11 */

  foreach X in gb_11 gb_10 G_11 {
    capture drop `X'
  }

 quie gen double gb_11 = 0 if hasmiss ==0
 quie gen double gb_10 = 0 if hasmiss ==0

 foreach X in $zvar g0 {
   mat aa_11 = b0_gmm1[1, "`X':_cons"]
   scalar aa_11 = aa_11[1,1]
   mat aa_10 = b0_gmm1[1, "`X'B:_cons"]
   scalar aa_10 = aa_10[1,1]

   if "`X'" ~= "g0" {
     quie replace gb_11 = gb_11 + scalar(aa_11)*`X' if hasmiss ==0
     quie replace gb_10 = gb_10 + scalar(aa_10)*`X' if hasmiss ==0
   }
   else {
     quie replace gb_11 = gb_11 + scalar(aa_11) if hasmiss ==0
     quie replace gb_10 = gb_10 + scalar(aa_10) if hasmiss ==0
   }
 }
    quie gen double G_11 = (m_d1)*exp(gb_11) + (1-m_d1)*exp(gb_10) if hasmiss ==0


  *************

if getout ==1 {
 exit
}
```

In [ ]:

```
***********************************
*** Estimate models with D=0 ******
***********************************

mat myini2 = (     .41535,  .6289736,  -.2667481,  .0098642,   /*
   */  .5633515,  .2420941,  -.0912889,  -1.035107,  2.650496,  /*
   */  -1.448135,  2.379549,  -2.76988  )

  scalar getout = 0

  forvalues i = 1/2 { /* run twice, the 2nd time using the first time's result as init. So

    if getout == 1 {
       exit
    }

     if `i' == 2 {
        mat myini2 = b0_gmm2
     }

         /* the number of equations (nequations) equals the sum of the numbers of var in $x
       gmm sftreat_GMM4_jup if Dvar == 0, nequations(12) parameters($xvar b0 $xvarB b0B $zv
           mylhs(lprodmain) myxvar($xvar) myzvar($zvar) mywght(W01) mywghtB(W00) ///
           myp(m_d1) mypB(m_d0) ///
           winitial(identity) ///
           from(myini2)   ///
           /* tracelevel(gradient) */ onestep technique(bfgs)


  **** further checking ***********

          local missingv = 0 /* flag */
          if _rc == 0 {
            mat checkv = e(V) /* V-COV matrix */
            local vsize = colsof(checkv)
            forvalues i = 1/`vsize' {
              local checkstd = checkv[`i', `i']
              if `checkstd' == 0 | `checkstd' > 2e+7 {
                 local missingv = 1
              }
            }
          }

 ************************************

  if _rc == 0 & `missingv' == 0 {
    mat b0_gmm2 = e(b)
  }
  else {

    scalar getout = 1

    exit
  }
    if getout == 1 {
      exit
    }
}
```

```
  if getout == 1 {
    exit
  }


  *** obtain H_01, H_00, G_01, G_00, etc., for LATE ****

  /* begin with H01, H00 */

  foreach X in hb_01 hb_00 H_01 H_00 {
    capture drop `X'
  }

 quie gen double hb_01 = 0 if hasmiss ==0
 quie gen double hb_00 = 0 if hasmiss ==0


 foreach X in $xvar b0 {
   mat aa_01 = b0_gmm2[1, "`X':_cons"]
   scalar aa_01 = aa_01[1,1]
   mat aa_00 = b0_gmm2[1, "`X'B:_cons"]
   scalar aa_00 = aa_00[1,1]

   if "`X'" ~= "b0" {
     quie replace hb_01 = hb_01 + scalar(aa_01)*`X' if hasmiss ==0
     quie replace hb_00 = hb_00 + scalar(aa_00)*`X' if hasmiss ==0
   }
   else {
     quie replace hb_01 = hb_01 + scalar(aa_01) if hasmiss ==0
     quie replace hb_00 = hb_00 + scalar(aa_00) if hasmiss ==0
   }
 }

    quie gen double H_01 = (m_d1)*hb_01 + (1-m_d1)*hb_00 if hasmiss ==0
    quie gen double H_00 = (m_d0)*hb_01 + (1-m_d0)*hb_00 if hasmiss ==0


  /* Now do G_01, G_00 */

  foreach X in gb_01 gb_00 G_01 G_00 {
    capture drop `X'
  }

 quie gen double gb_01 = 0 if hasmiss ==0
 quie gen double gb_00 = 0 if hasmiss ==0


 foreach X in $zvar g0 {
   mat aa_01 = b0_gmm2[1, "`X':_cons"]
   scalar aa_01 = aa_01[1,1]
   mat aa_00 = b0_gmm2[1, "`X'B:_cons"]
   scalar aa_00 = aa_00[1,1]

   if "`X'" ~= "g0" {
     quie replace gb_01 = gb_01 + scalar(aa_01)*`X' if hasmiss ==0
     quie replace gb_00 = gb_00 + scalar(aa_00)*`X' if hasmiss ==0
   }
   else {
     quie replace gb_01 = gb_01 + scalar(aa_01) if hasmiss ==0
     quie replace gb_00 = gb_00 + scalar(aa_00) if hasmiss ==0
   }
```

```
}

   quie gen double G_01 = (m_d1)*exp(gb_01) + (1-m_d1)*exp(gb_00) if hasmiss ==0
   quie gen double G_00 = (m_d0)*exp(gb_01) + (1-m_d0)*exp(gb_00) if hasmiss ==0
```

2.8 calculating the treatment effects

In [ ]:

```
   *** calculate LATE (see Identification of CLATE and LATE in the paper)

     quie probit Dvar Z1_inst $xvar $zvar  $Z_predictors_pca  if hasmiss == 0, iterate(50)

     foreach X in linear linear0 linear1 Delta Delta_i tem1 tem2 tem3 {
      capture drop `X'
     }
     quie predict double linear  if e(sample), xb
     quie gen double linear0 = linear - _b[Z1_inst]*Z1_inst + _b[Z1_inst] if e(sample)
     quie gen double linear1 = linear - _b[Z1_inst]*Z1_inst  if e(sample)
     quie gen double Delta_i = normal(linear0) - normal(linear1) if e(sample)
     quie sum Delta_i if e(sample)
     scalar Delta = r(mean)

     foreach X in 11 01 00 {
        capture drop tem1
        quie gen double tem1 = Delta_i*(H_`X')/(scalar(Delta))
        quie sum tem1
        scalar H_`X'_hat = r(mean)

        capture drop tem1
        quie gen double tem1 = Delta_i*(G_`X')/(scalar(Delta))
        quie sum tem1
        scalar G_`X'_hat = r(mean)
     }

      scalar LATE = scalar(H_11_hat) - scalar(G_11_hat) -scalar(H_00_hat) + scalar(G_00_hat
      scalar DLATE = scalar(H_11_hat) - scalar(G_11_hat) -scalar(H_01_hat) + scalar(G_01_ha
      scalar ILATE = scalar(H_01_hat) - scalar(G_01_hat) -scalar(H_00_hat) + scalar(G_00_ha

      scalar LATE  = scalar(H_11_hat) - scalar(G_11_hat) -scalar(H_00_hat) + scalar(G_00_ha
      scalar LATEh = scalar(H_11_hat)  -scalar(H_00_hat)
      scalar LATEg = -(- scalar(G_11_hat)  + scalar(G_00_hat))

      scalar DLATEh = scalar(H_11_hat)  -scalar(H_01_hat)
      scalar DLATEg =  -(- scalar(G_11_hat) + scalar(G_01_hat))
      scalar ILATEh = scalar(H_01_hat)  -scalar(H_00_hat)
      scalar ILATEg =  -(- scalar(G_01_hat)  + scalar(G_00_hat))


di LATE
di DLATE
di ILATE

di LATEh
di DLATEh
di ILATEh

di LATEg
di DLATEg
di ILATEg
```