GMM and Maximum Likelihood estimators with Mata and moptimize

Alfonso Miranda
Centro de Investigación y Docencia Económicas (CIDE)
(alfonso.miranda@cide.edu)

Mexican Stata Users Group meeting

May 18th 2016

Introduction

- `moptimize()` is Mata's and Stata's premier optimization routine. This is the routine used by most of the official optimization-based estimators implemented in Stata.
- From Stata 11 Stata's ML is a wrapper for `moptimize()`, that does life easier for the user.
  - Always use ML instead of `moptimize()` when fitting a model by maximum likelihood
  - `moptimize()` is intended for use when you want to work directly on Mata, or when you working with a problem that does not fit into the ML environment.

Mathematical statement of the `moptimize()` problem

$$\max_{((\mathbf{b}_1,c_1),(\mathbf{b}_2,c_2)\ldots,(\mathbf{b}_m,c_m))} f(p_1, p_2, \ldots, p_m; y_1, y_2, \ldots, y_D)$$

where,

$$p_1 = \mathbf{X}_1\mathbf{b}_1' : +c_1$$

$$\vdots$$

$$p_m = \mathbf{X}_1\mathbf{b}_m' : +c_m$$

$f()$ is the objective function. $\mathbf{X}_j$ is a $N_j \times k_j$ matrix, $\mathbf{b}_j$ is a $1 \times k_j$ row vector, and c_j is a 1×1 scalar (the constant), for $j = 1, \ldots, m$. The response variables $y_1, y_2, \ldots, y_D$ may have arbitrary dimension.

- Usually, $N_1 = N_2 = \ldots = N_m$, and the model is said to be fit on data of N observations.
- Also $y_1, y_2, \ldots, y_D$ are usually of N observations each.
- But this does not need to be the case!!

Linear model example

$$\max_{(\mathbf{b}_1,c_1),(c_2)} \sum ln(normalden(y - p_1, 0, p_2))$$

with,

$$p_1 = \mathbf{X}_1\mathbf{b}_1' : +c_1$$
$$p_2 = c_2$$

and y is $N \times 1$.

- This is an example of a two equation (two parameter) model.
- The objective function is maximised in terms of p_1 and p_2, $\mathbf{X}_1$ and c_1 play a secondary role (just determining p_1).
- The evaluator program will be written in terms of p_1 and p_2.

Note that the variance s^2 is given by p_2, and currently, we have $p_2 = c_2$, that is, a constant. We could easily write $p_2 = \mathbf{X}_2\mathbf{b}_2' : +c_2$ and that would allow the variance to depend on a set of explanatory variables. From the point of view of `moptimize()` this modified problem is the same as the original one.

ML with Moptimize

linear regression with `moptimize()`

Ok, now to the code. We start defining our mata function, which will call `linregeval()`.

```
sysuse auto, clear
mata:
function linregeval(transmorphic M, real rowvector b,
 real colvector lnf)
 {
```

The first argument defines a *handle M* (a pointer!!). This handle contains all the information of our optimisation problem and once it is set Mata will know what we are talking about everytime we invoke the handle *M*. The second argument contains the current value of the coefficients *b* (which is updated at each iteration step). And the third argument *lnf* is current log-likelihood value

Next we need to parse the dependent variable y and evaluate the current value of $\mathbf{x}_i\beta$ and σ. To archive that just make use of the `moptimize_util_depvar()` and `moptimize_util_xb()` functions

```
y = moptimize_util_depvar(M, 1)
xb = moptimize_util_xb(M, b, 1)
lns = moptimize_util_xb(M, b, 2)
s = exp(lns)
lnf = ln(normalden(y:-xb, 0, s))
```

σ is a standard deviation and cannot be negative. To avoid problems with the optimisation we must explicitly ensure that this requirement is met whatever number is the current value of s.

```
s = exp(lns)
```

Finally we fill in the log-likelihood value and close the curly bracket. Notice that in *lf* evaluators *lnf* is a *N times*1 vector.

```
lnf = ln(normalden(y:-xb, 0, s))
}
```

that concludes the writing of our `moptimize()` evaluator function. Now, we need to initialise `moptimize()` and parse all the information it needs to solve the problem.

```
M = moptimize_init()
```

`moptimize_init()` initialises `moptimize()`, allocates memory to the problem we will work on, and parses that memory address to handle *M*. So, `moptimize_init()` creates a pointer but not only that.

```
moptimize_init_evaluator(M, &linregeval())
moptimize_init_evaluatortype(M, "lf")
```

Next, we parse to `moptimize()` the name of our evaluator. Here you'll see another application of pointers, as we point towards `linregeval()` using `&linregeval()`. Declare the "evaluator type" using `moptimize_init_evaluatortype(.)`. In this case we will use a "lf" evaluator (more of this later).

```
moptimize_init_depvar(M, 1, "mpg")
moptimize_init_eq_indepvars(M, 1, "weight foreign")
moptimize_init_eq_indepvars(M, 2, "")
```

Now we declare the dependent variable using `moptimize_init_depvar(.)` and the independent variables for each equation using `moptimize_init_eq_indepvars()`. Notice that for linear regression we have one dependent variable, and two equations (one for $\mathbf{x}_i\boldsymbol{\beta}$ and one for $\ln\sigma$). Finally, we perform the maximisation and display results.

```
moptimize(M)
moptimize_result_display(M)
```

This is the whole code together

```
sysuse auto, clear
mata:
function linregeval(transmorphic M, real rowvector b,
 real colvector lnf)
 {
  xb = moptimize_util_xb(M, b, 1)
  s = moptimize_util_xb(M, b, 2)
  y = moptimize_util_depvar(M, 1)
  s = exp(s)
  lnf = ln(normalden(y:-xb, 0, s))
}
M = moptimize_init()
moptimize_init_evaluator(M, &linregeval())
 moptimize_init_evaluatortype(M, "lf")
moptimize_init_depvar(M, 1, "mpg")
moptimize_init_eq_indepvars(M, 1, "weight foreign")
moptimize_init_eq_indepvars(M, 2, "")
moptimize(M)
moptimize_result_display(M)
```

```
moptimize(M)
initial:       f(p) =     -<inf>  (could not be evaluated)
feasible:      f(p) = -12949.708
  (output omitted)
Iteration 7:   f(p) = -194.18306
moptimize_result_display(M)

                                                Number of obs   =         74

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
      weight |  -.0065879   .0006241   -10.56   0.000     -.007811   -.0053647
     foreign |  -1.650027   1.053958    -1.57   0.117    -3.715746    .4156915
       _cons |    41.6797   2.121196    19.65   0.000     37.52223    45.83717
-------------+----------------------------------------------------------------
eq2          |
       _cons |   1.205157   .0821995    14.66   0.000     1.044049    1.366265
------------------------------------------------------------------------------
```

Same thing with `regress`

```
. regress mpg weight foreign

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  2,    71) =   69.75
       Model |   1619.2877     2  809.643849            Prob > F      =  0.0000
    Residual |  824.171761    71   11.608053            R-squared     =  0.6627
-------------+------------------------------           Adj R-squared =  0.6532
       Total |  2443.45946    73  33.4720474            Root MSE      =  3.4071

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0065879   .0006371   -10.34   0.000    -.0078583   -.0053175
     foreign |  -1.650029   1.075994    -1.53   0.130      -3.7955    .4954422
       _cons |   41.6797   2.165547    19.25   0.000     37.36172    45.99768
------------------------------------------------------------------------------
```

Various flavours

- **lf evaluators** Requires observation-by-observation calculation of the log-likelihood function (i.e., no good for panel data estimators because the likelihood is only defined at the individual/panel level!)
- **d evaluators** Relaxes the requirement that the log-likelihood function be summable over the observations and thus suitable for all types of estimators. Robust estimates of variance, adjustment for clustering or survey design is not automatically done and dealing with this requires substantial effort

- **gf evaluators** Relaxes the requirement that the log-likelihood function be summable over the observations and thus suitable for all types of estimators. Type `gf` evaluators can work with panel data models and 'resurrect' robust standard errors, clustering, and survey-data adjustments. However, type `gf` evaluators are harder to write than `lf` or `d` evaluators
- **q evaluators** are used to deal with quadratic optimisation problems such as GMM or NLS. These evaluators are suitable for cross-section and panel data models. The optimiser does not return a covariance matrix for the estimated parameters, only point estimates are reported. Estimation of the covariance matrix should be performed by had using Mata's matrix algebra facilities.

moptimize() sub-flavours

evaluatortype	Description
"lf"	*function*() returns *N* x 1 colvector value
"d0"	*function*() returns scalar value
"d1"	same as "d0" and returns gradient rowvector
"d2"	same as "d1" and returns Hessian matrix
"d1debug"	same as "d1" but checks gradient
"d2debug"	same as "d2" but checks gradient and Hessian
"lf0"	*function*() returns *N* x 1 colvector value
"lf1"	same as "lf0" and returns equation-level score matrix
"lf2"	same as "lf1" and returns Hessian matrix
"lf1debug"	same as "lf1" but checks gradient
"lf2debug"	same as "lf2" but checks gradient and Hessian
"gf0"	*function*() returns *N* x 1 colvector value
"gf1"	same as "gf0" and returns score matrix
"gf2"	same as "gf1" and returns Hessian matrix
"gf1debug"	same as "gf1" but checks gradient
"gf2debug"	same as "gf2" but checks gradient and Hessian
"q0"	*function*() returns colvector value
"q1"	same as "q0" and returns score matrix
"q1debug"	same as "q1" but checks gradient

Probit example

- Dependent variable y_i is 1×1 binary variable.
- Set of control variables $\mathbf{x}_i$ is a $1 \times K$ vector of explanatory variables (including the constant).
- $\boldsymbol{\beta}$ is a $K \times 1$ vector of coefficients (parameters to be estimated).
- We have a sample of size N. That is, we got data $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, N$.

The response probability is given by

$$\pi_i = P(y_i = 1|\mathbf{x}_i) = \Phi(\mathbf{x}_i\boldsymbol{\beta})$$

The contribution of the i-th individual to the likelihood is

$$\begin{aligned}\ell_i = \ln L_i &= y_i \ln \Phi(\mathbf{x}_i\boldsymbol{\beta}) + (1 - y_i)\Phi(-\mathbf{x}_i\boldsymbol{\beta}) \\ &= \Phi(q_i\mathbf{x}_i\boldsymbol{\beta}); \;\; q_i = 2y_i - 1.\end{aligned}$$

The first derivative of ℓ_i with respect to the linear index $\mathbf{x}_i\boldsymbol{\beta}$ gives the the score (at equation level) or gradient

$$g_i = \frac{\partial \ell_i}{\partial \mathbf{x}_i\boldsymbol{\beta}} = \frac{q_i \phi(\mathbf{x}_i\boldsymbol{\beta})}{\Phi(q_i \mathbf{x}_i\boldsymbol{\beta})}$$

The second derivative of ℓ_i with respect to the linear index $\mathbf{x}_i\boldsymbol{\beta}$ gives the Hessian (at equation level)

$$H_i = \frac{\partial^2 \ell_i}{\partial^2 \mathbf{x}_i\boldsymbol{\beta}} = -g_i(g_i + \mathbf{x}_i\boldsymbol{\beta}).$$

```
/* Probit by ML d0 with moptimize*/
set more off
sysuse cancer, clear
gen drug2=drug==2
gen drug3=drug==3
global xvars "drug2 drug3 age"
mata:
mata clear
function myprobit_d0(transmorphic M, real scalar todo,
 real rowvector b, fv, g, H)
{
 y = moptimize_util_depvar(M, 1)
 xb = moptimize_util_xb(M, b, 1)
 N = rows(y)
 q = 2*y - J(N,1,1)
 Sigma = ln(normal(q:*xb))
 fv =  moptimize_util_sum(M, Sigma)
}
M = moptimize_init()
moptimize_init_evaluator(M, &myprobit_d0())
moptimize_init_evaluatortype(M, "d0")
moptimize_init_depvar(M, 1, "died")
moptimize_init_eq_indepvars(M, 1, "drug2 drug3 age")
moptimize(M)
moptimize_result_display(M)
end
```

```
: moptimize(M)
initial:       f(p) = -33.271065
alternative:   f(p) = -31.427839
  (output omitted)
: moptimize_result_display(M)
```

Number of obs = 48

died	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
drug2	-1.941275	.597057	-3.25	0.001	-3.111485	-.7710645
drug3	-1.792924	.5845829	-3.07	0.002	-2.938686	-.6471627
age	.0666733	.0410666	1.62	0.104	-.0138158	.1471623
_cons	-2.044876	2.28428	-0.90	0.371	-6.521983	2.432231

```
 /* probit by ML d1 with moptimize */
set more off
sysuse cancer, clear
gen drug2=drug==2
gen drug3=drug==3
global xvars "drug2 drug3 age"
mata:
mata clear
function myprobit_d1(transmorphic M, real scalar todo,
 real rowvector b, fv, g, H)

 y = moptimize_util_depvar(M, 1)
 xb = moptimize_util_xb(M, b, 1)
 N = rows(y)
 q = 2*y - J(N,1,1)
 Sigma = ln(normal(q:*xb))
 fv =  moptimize_util_sum(M, Sigma)
 if (todo>0)
  d = (q:*normalden(xb)):/normal(q:*xb)
  g =  moptimize_util_vecsum(M, 1, d, fv)


M = moptimize_init()
moptimize_init_evaluator(M, &myprobit_d1())
moptimize_init_evaluatortype(M, "d1")
moptimize_init_depvar(M, 1, "died")
moptimize_init_eq_indepvars(M, 1, "drug2 drug3 age")
moptimize(M)
moptimize_result_display(M)
end
```

```
/* Probit by ML d3 with moptimize */
set more off
sysuse cancer, clear
gen drug2=drug==2
gen drug3=drug==3
global xvars "drug2 drug3 age"
mata:
mata clear
function myprobit_d2(transmorphic M, real scalar todo,
 real rowvector b, fv, g, H)
{
 y = moptimize_util_depvar(M, 1)
 xb = moptimize_util_xb(M, b, 1)
 N = rows(y)
 q = 2*y - J(N,1,1)
 Sigma = ln(normal(q:*xb))
 fv =  moptimize_util_sum(M, Sigma)
 if (todo>0) {
  d = (q:*normalden(xb)):/normal(q:*xb)
  g =  moptimize_util_vecsum(M, 1, d, fv)
 }
 if (todo>1) {
  Dd = J(N,1,-1):*d:*(d+xb)
  H = moptimize_util_matsum(M, 1, 1, Dd, fv)
 }
}
M = moptimize_init()
moptimize_init_evaluator(M, &myprobit_d2())
moptimize_init_evaluatortype(M, "d2")
moptimize_init_depvar(M, 1, "died")
moptimize_init_eq_indepvars(M, 1, "drug2 drug3 age")
moptimize(M)
moptimize_result_display(M)
end
```

```
/* probit by ML gf1 with moptimize */
set more off
use http://www.stata-press.com/data/r12/union
global xvars "age grade not_smsa year south"
mata:
mata clear
function myprobit_gf1(transmorphic M, real scalar todo,
 real rowvector b, fv, g, H)
{
 y = moptimize_util_depvar(M, 1)
 xb = moptimize_util_xb(M, b, 1)
 N = rows(y)
 q = 2*y - J(N,1,1)
 fv = ln(normal(q:*xb))
 if (todo>0) {
  st_view(X,.,tokens(st_global("xvars")))
  one = J(rows(X),1,1)
  X = (X,one)
  g = (q:*normalden(xb)):/normal(q:*xb)
  g = g:*X
 }
}
M = moptimize_init()
moptimize_init_evaluator(M, &myprobit_gf1())
moptimize_init_evaluatortype(M, "gf1")
moptimize_init_vcetype(M, "robust")
moptimize_init_cluster(M, "idcode")
moptimize_init_depvar(M, 1, "union")
moptimize_init_eq_indepvars(M, 1, "$xvars")
moptimize(M)
moptimize_result_display(M)
end
```

Writing a Mata evaluator for Stata 's ML

```
/* Probit by ML with ml d2 */
set more off
sysuse cancer, clear
gen drug2=drug==2
gen drug3=drug==3
global xvars "drug2 drug3 age"
mata:
mata clear
function myprobit_d2(transmorphic M, real scalar todo,
 real rowvector b, fv, g, H)
{
 y = moptimize_util_depvar(M, 1)
 xb = moptimize_util_xb(M, b, 1)
 N = rows(y)
 q = 2*y - J(N,1,1)
 Sigma = ln(normal(q:*xb))
 fv =  moptimize_util_sum(M, Sigma)
 if (todo>0) {
  d = (q:*normalden(xb)):/normal(q:*xb)
  g =  moptimize_util_vecsum(M, 1, d, fv)
 }
 if (todo>1) {
  Dd = J(N,1,-1):*d:*(d+xb)
  H = moptimize_util_matsum(M, 1, 1, Dd, fv)
 }
}
end
/* Run the regression */
ml model d2 myprobit_d2() (died = $xvars), maximize
ml display
```

GMM with Moptimize

Introduction

- When $E(u_{ig}|\mathbf{x}_{ig}) \neq 0$, $g = 1, \ldots, G$, an IV approach is needed for performing system estimation (SIV).
- Estimation based on the principle of **generised method of moments** is the the modern approach to SIV.
  - White (1982) and Hansen (1982) derive the asymptotic properties of GMM.
- The most familiar application of SIV estimation is the **simultaneous equations model** (SEM).

Example

Labour supply and wage offer

$$h^s(w) = \gamma_1 w + \mathbf{z}_1\boldsymbol{\delta}_1 + u_1 \quad (1)$$

$$w^o(h) = \gamma_2 h + \mathbf{z}_2\boldsymbol{\delta}_2 + u_2 \quad (2)$$

Rarely can assume that an individual gets an exogenous wage offer and then, at that wage, decide how many hours to work. A reasonable assumption is that we observe equilibrium quantities.

$$h_i = \gamma_1 w_i + \mathbf{z}_{i1}\boldsymbol{\delta}_1 + u_{i1} \quad (3)$$

$$w_i = \gamma_2 h_i + \mathbf{z}_{i2}\boldsymbol{\delta}_2 + u_{2i} \quad (4)$$

We assume that $\mathbf{z}_1$ and $\mathbf{z}_2$ are exogenous. That is $E(u_1|\mathbf{z}_1, \mathbf{z}_2) = E(u_2|\mathbf{z}_1, \mathbf{z}_2) = 0$. In general u_{i1} will be correlated with w_i and u_{i2} will be correlated with h_i. That is, w_i is endogenous in equation (3) and h_i is endogenous in equation (4).

System of equations

The population model is a set of G linear equations

$$y_{ig} = \mathbf{x}_{ig}\boldsymbol{\beta}_g + u_{ig} \tag{5}$$

where $\mathbf{x}_{ig}$ is $1 \times K_g$, $i = 1, \ldots, N$ and $g = 1, \ldots, G$. We can stack the G equations to write

$$\begin{bmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{ig} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{i1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{x}_{i2} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{x}_{iG} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_G \end{bmatrix} + \begin{bmatrix} u_{i1} \\ u_{i2} \\ \vdots \\ u_{iG} \end{bmatrix}$$

which can be written as

$$\mathbf{y}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{u}_i \tag{6}$$

where $\mathbf{y}_i$ is $(G \times 1)$, $\mathbf{X}_i$ is $(G \times K)$, $\boldsymbol{\beta}$ is $(K \times 1)$, $\mathbf{u}_i$ is $(G \times 1)$ and $K = K_1 + \ldots + K_G$. Crucially, some $\mathbf{x}_g$ are correlated with u_g.

System of equations

For each equation we have a set of instrumental variables, a $1 \times L_g$ vector $\mathbf{z}_g$ such that

$$E(\mathbf{z}_g' u_g) = 0, \;\; g = 1, \ldots, G. \tag{7}$$

in much applications the constant is part of $\mathbf{z}_g$ such that $E(u_g) = 0$ for all g. Define

$$\mathbf{Z}_i = \begin{bmatrix} \mathbf{z}_{i1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{z}_{i2} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{z}_{iG} \end{bmatrix}$$

which has dimensions $G \times L$, where $L = L_1 + \ldots + L_G$.

Assumption (SIV.1)

$$E\left(\mathbf{Z}_i'\mathbf{u}_i\right) = \mathbf{0}$$

where $\mathbf{Z}_i$ *is a* $G \times L$ *matrix of observable instrumental variables.*

Assumption (SIV.2)

$$rank\left(\mathbf{Z}_i'\mathbf{X}_i\right) = K$$

This is the **rank condition** *and is a generalisation of the condition that is imposed in 2SLS estimation. This requires the columns of of* $\mathbf{Z}_i'\mathbf{X}_i$ *to be linearly independent. Necessary for the rank condition is the* **order condition***:* $L \geq K$*.*

Notice that

$$E\left(\mathbf{Z}_i'\mathbf{X}_i\right) = \begin{bmatrix} E\left(\mathbf{z}_{i1}'\mathbf{x}_{i1}\right) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & E\left(\mathbf{z}_{i2}'\mathbf{x}_{i2}\right) & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & E\left(\mathbf{z}_{iG}'\mathbf{x}_{iG}\right) \end{bmatrix}$$

where $E(\mathbf{z}_i'\mathbf{x}_i)$ is $L_g \times K_g$. Assumption SIV.2 requires this matrix to have full column rank. A well known result from linear algebra says that a block diagonal matrix has full rank if and only if each block in the matrix has full column rank. In other words the rank condition requires

$$\text{rank}(\mathbf{z}_{ig}'\mathbf{x}_{ig}) = K_g \quad g = 1, \ldots, G. \tag{8}$$

This is the condition to estimate each equation by 2SLS. So, identification of systems by IV is equivalent to identification equation by equation. This assumes that no restrictions are set on $\boldsymbol{\beta}$.

Generalised method of moments (GMM)

We use the **analogy** principle, which says we can estimate a parameter by replacing a population moment condition with its sample analogue. The orthogonality condition SIV.1 suggest that $\boldsymbol{\beta}$ is the *unique* vector solving the set of population moment conditions

$$E\left(\mathbf{Z}_i'\left(\mathbf{y}_i - \mathbf{X}_i\boldsymbol{\beta}\right)\right) = \mathbf{0} \tag{9}$$

the uniqueness follows from the SIV.2 assumption. Because sample averages are consistent estimators of population moments we can use the analogy principle a choose

$$N^{-1}\sum_{i=1}^{N}\left(\mathbf{Z}_i'\left(\mathbf{y}_i - \mathbf{X}_i\widehat{\boldsymbol{\beta}}\right)\right) = \mathbf{0} \tag{10}$$

This is a set of L linear equations on the K unknowns $\widehat{\boldsymbol{\beta}}$.

Just identified case

Consider the case where we have exactly the same number of instruments as the number of parameters to be estimated $L = K$. Then if the $\sum_{i=1}^{N} \mathbf{Z}_i'\mathbf{X}_i$ matrix is nonsingngular we can solve for $\widehat{\boldsymbol{\beta}}$.

$$\widehat{\boldsymbol{\beta}}_{SIV} = \left(N^{-1} \sum_{i=1}^{N} \mathbf{Z}_i'\mathbf{X}_i \right)^{-1} \left(N^{-1} \sum_{i=1}^{N} \mathbf{Z}_i'\mathbf{y}_i \right) \tag{11}$$

which can be written in full matrix notation as

$$\widehat{\boldsymbol{\beta}}_{SIV} = \left(\mathbf{Z}'\mathbf{X} \right)^{-1} \mathbf{Z}'\mathbf{Y} \tag{12}$$

where $\mathbf{Z}$ is the $NG \times L$ matrix obtained by stacking $\mathbf{Z}_i$ from $i = 1, \ldots, N$, $\mathbf{X}$ is the $NG \times K$ matrix obtained by stacking $\mathbf{X}_i$ from $i = 1, \ldots, N$, and $\mathbf{Y}$ is the $NG \times 1$ matrix obtained by stacking $\mathbf{y}_i$ from $i = 1, \ldots, N$. We can use the LLN to show that $\widehat{\boldsymbol{\beta}}_{SIV}$ is a consistent estimator of $\boldsymbol{\beta}$ and the CLT to show that $\widehat{\boldsymbol{\beta}}_{SIV}$ is asymptotically normal.

Overidentified case

Now we have more instruments than parameters to be estimated $L > K$. In this case choosing an estimator $\widehat{\beta}$ is more complicated because, except for special cases, equation (10) will not have a solution. **Instead we choose $\widehat{\beta}$ to make the vector in equation (10) as "small" as possible**. One possibility is to minimise the squared Euclidian length of the $L \times 1$ vector in (10). This approach suggest choosing $\widehat{\beta}$ to make

$$\left[\sum_{i=1}^{N} \mathbf{Z}_i' \left(\mathbf{y}_i - \mathbf{X}_i \widehat{\beta}\right)\right]' \left[\sum_{i=1}^{N} \mathbf{Z}_i' \left(\mathbf{y}_i - \mathbf{X}_i \widehat{\beta}\right)\right]$$

as small as possible. This produces a consistent estimator. However, it rarely gives an efficient estimator.

GMM estimator

A general class of estimators is obtained by using a **weighting matrix** in the quadratic form. Let $\widehat{\mathbf{W}}_N$ be an $L \times L$, symmetric, positive definite matrix where the hat is written to stress the fact that $\widehat{W}_N$ is generally estimated beforehand. The generalised method of moments (GMM) estimator minimises

$$Q_N(\beta) = \left[\sum_{i=1}^{N} \mathbf{Z}_i' \left(\mathbf{y}_i - \mathbf{X}_i\widehat{\beta}\right)\right]' \widehat{\mathbf{W}}_N \left[\sum_{i=1}^{N} \mathbf{Z}_i' \left(\mathbf{y}_i - \mathbf{X}_i\widehat{\beta}\right)\right] \quad (13)$$

Because $Q_N(\beta)$ is a quadratic form, equation (13) has a unique solution. In particular

$$\widehat{\beta}_{GMM} = \left(\mathbf{X}'\mathbf{Z}\widehat{\mathbf{W}}\mathbf{Z}'\mathbf{X}\right)^{-1}\left(\mathbf{X}'\mathbf{Z}\widehat{\mathbf{W}}\mathbf{Z}'\mathbf{Y}\right) \quad (14)$$

assuming that $\mathbf{X}'\mathbf{Z}\widehat{\mathbf{W}}\mathbf{Z}'\mathbf{X}$ is not singular.

To show that this is a consistent estimator we need

Assumption (SIV.3)

$$\widehat{\mathbf{W}}_N \xrightarrow{p} \mathbf{W} \text{ as } N \to \infty$$

where $\mathbf{W}$ *is a nonrandom, symetric,* $L \times L$ *positive definite matrix, which does not depend on* β*. Notice that* $\widehat{\mathbf{W}}$ *is a stochastic matrix that depends on the sample size.*

In applications, the convergence in SIV.3 will follow from a LLN because $\widehat{\mathbf{W}}$ will be a function of sample averages.

- Different choices of $\widehat{\mathbf{W}}$ will give different estimators that, although consistent, have different variances for $L > K$.
- This approach is quite general and can be used for nonlinear models.

Is easy to show that the GMM estimator $\hat{\beta}_{GMM}$ is

- A consistent estimator for β.
- Asymptotically normal.

System 2SLS

We need an estimator for $\mathbf{W}$. A choice that leads to a useful and familiar-looking estimator is

$$\widehat{\mathbf{W}} = \left(N^{-1} \sum_{i=1}^{N} \mathbf{Z}_i' \mathbf{Z}_i \right)^{-1} = (N^{-1} \mathbf{Z}'\mathbf{Z}) \tag{15}$$

which is a consistent estimator of $E(\mathbf{Z}_i'\mathbf{Z}_i)$. SIV.3 requires that $E(\mathbf{Z}_i'\mathbf{Z}_i)$. exists, which is not very restrictive. Plugging this into (14) we get

$$\widehat{\boldsymbol{\beta}}_{S2SLS} = \left[\mathbf{X}'\mathbf{Z} \left(\mathbf{Z}'\mathbf{Z} \right)^{-1} \mathbf{Z}'\mathbf{X} \right]^{-1} \mathbf{X}'\mathbf{Z} \left(\mathbf{Z}'\mathbf{Z} \right)^{-1} \mathbf{Z}'\mathbf{Y} \tag{16}$$

This system 2SLS is equivalent to 2SLS equation by equation. This, however, is not the asymptotically efficient estimator.

Optimal weighting matrix

We would like to choose a $\widehat{\mathbf{W}}$ that produces the the smallest asymptotic variance. Notice that,

$$\text{Avar}\sqrt{N}\left(\widehat{\boldsymbol{\beta}}_{GMM}-\boldsymbol{\beta}\right)=\left(\mathbf{C}'\mathbf{W}\mathbf{C}\right)^{-1}\mathbf{C}'\mathbf{W}\boldsymbol{\Lambda}\mathbf{W}\mathbf{C}\left(\mathbf{C}'\mathbf{W}\mathbf{C}\right)^{-1} \quad (17)$$

with $\mathbf{C}\equiv E(\mathbf{Z}_i'\mathbf{X}_i)$, simplifies to

$$\text{Avar}\sqrt{N}\left(\widehat{\boldsymbol{\beta}}_{GMM}-\boldsymbol{\beta}\right)=\left(\mathbf{C}'\boldsymbol{\Lambda}^{-1}\mathbf{C}\right)^{-1} \quad (18)$$

if we set $\mathbf{W}=\boldsymbol{\Lambda}^{-1}$. Further, it is possible to show that

$$\left(\mathbf{C}'\mathbf{W}\mathbf{C}\right)^{-1}\mathbf{C}'\mathbf{W}\boldsymbol{\Lambda}\mathbf{W}\mathbf{C}\left(\mathbf{C}'\mathbf{W}\mathbf{C}\right)^{-1}-\left(\mathbf{C}'\boldsymbol{\Lambda}^{-1}\mathbf{C}\right)^{-1}$$

is positive definite for any $L\times L$ positive definite matrix $\mathbf{W}$. Hence $\mathbf{W}=\boldsymbol{\Lambda}^{-1}$ is the optimal weighting matrix!

Assumption (SIV.4)

$$\mathbf{W} = \boldsymbol{\Lambda}^{-1}$$

where $\boldsymbol{\Lambda} \equiv E\left(\mathbf{Z}_i'\mathbf{u}_i\mathbf{u}_i'\mathbf{Z}_i\right) = \mathrm{Var}(\mathbf{Z}_i'\mathbf{u}_i)$.

Under assumptions SIV.1-SIV.4 the GMM estimator is efficient among all GMM estimators. So, provided that we can find a consistent estimator for $\boldsymbol{\Lambda}$ we can get the asymptotically efficient GMM estimator.

GMM with optimal weighting matrix

- Let $\widehat{\widehat{\boldsymbol{\beta}}}$ Get a consistent estimator of $\boldsymbol{\beta}$. In most cases this is the system 2SLS estimator or GMM with $\widehat{\mathbf{W}} = \mathbf{I}_L$.
- obtain the $G \times 1$ residual vectors.

$$\hat{\mathbf{u}}_i = \mathbf{y}_i - \widehat{\widehat{\boldsymbol{\beta}}}, \quad i = 1, \ldots, N$$

- A consistent estimator of $\boldsymbol{\Lambda}$ is $\widehat{\boldsymbol{\Lambda}} = N^{-1} \sum_{i=1}^{N} \mathbf{Z}_i' \hat{\mathbf{u}}_i \hat{\mathbf{u}}_i' \mathbf{Z}_i$.
- Choose

$$\widehat{\mathbf{W}} \equiv \widehat{\boldsymbol{\Lambda}} = \left(N^{-1} \sum_{i=1}^{N} \mathbf{Z}_i' \hat{\mathbf{u}}_i \hat{\mathbf{u}}_i' \mathbf{Z}_i \right)^{-1}$$

- Use $\widehat{\mathbf{W}}$ to obtain the asymptotically optimal GMM estimator.
- This puts no restrictions on the form of $\boldsymbol{\Lambda}$.

The assynptotic variance of the optimal GMM estimator is

$$\text{Avar}\left(\widehat{\beta}_{GMM}\right) = \left[(\mathbf{X}'\mathbf{Z})\left(\sum_{i=1}^{N}\mathbf{Z}_i'\hat{\mathbf{u}}_i\hat{\mathbf{u}}_i'\mathbf{Z}\right)^{-1}(\mathbf{Z}'\mathbf{X})\right]^{-1}$$

where $\hat{\mathbf{u}}_i = \mathbf{y}_i - \mathbf{X}_i\widehat{\beta}_{GMM}$. Asymptotically it makes no difference if the first-stage residuals $\hat{\mathbf{u}}_i$ are used. This is called the **minimum chi-square estimator**.

Remarks

- When the system is just identified ($L = K$), all GMM estimators reduce to IV equation-by-equation.
- When the system is just identified ($L = K$), SIV reduces to IV equation-by-equation.

Hypothesis testing

Under SIV.1-SIV.4 we can proceed as usual

- Confidence intervals.
- t statistics and test are valid (and robust to heteroskedasticity).
- Wald tests and F-tests.

Testing overidentification restrictions

When $L > K$ we can test whether overidentifying restrictions are valid. It can be shown that

$$\left(N^{-1/2} \sum_{i=1}^{N} \mathbf{Z}_i' \hat{\mathbf{u}}_i \right)' \widehat{W} \left(N^{-1/2} \sum_{i=1}^{N} \mathbf{Z}_i' \hat{\mathbf{u}}_i \right) \overset{a}{\sim} \chi^2_{(L-K)} \tag{19}$$

under the null H_0: $E(\mathbf{Z}_i' \mathbf{u}_i) = \mathbf{0}$. Because we have used K orthogonality conditions to estimate $\hat{\boldsymbol{\beta}}_{GMM}$ we loss K degrees of freedom. Notice that here we use the optimal weighting matrix $\widehat{W}$, otherwise the statistics does not have a limiting chi-square distribution. When $L = K$ the left hand side is exactly zero and there are no overidentifying restrictions to be tested.

Testing overidentification restrictions

Other (equivalent) names for the over identification test are

- Hansen's test
- Sargan's test
- Hansen-Sargan test.

To the code

```
/* GMM WITH MATA */

/* Simulate data */
set more off
clear
set obs 100
set seed 6789
gen x2 = rnormal()
gen x3 = rnormal()
gen z1 = rnormal()
gen z2 = rnormal()
gen z3 = rnormal()
gen IQ = rnormal()
gen x1 = z1 + z2 + z3 + IQ + rnormal()
gen y1 = 0.5 + 2*x1 + 2*x2 + 2*x3 + IQ + rnormal() // x1 is correlated with composite
gen y2 = 0.2 + 4*x1 + 4*x3 + IQ + rnormal()       // error through IQ.
gen one = 1
global depvar1 "y1"
global depvar2 "y2"
global xvars1 "x1 x2 x3 one"
global xvars2 "x1 x3 one"
global zvars1 "x2 x3 z1 z2 z3 one"
global zvars2 "x3 z1 z2 z3 one"
```

```
/* Bring Data to Mata */
mata
mata clear
y1 = NULL
y2 = NULL
X1 = NULL
X2 = NULL
Z1 = NULL
Z2 = NULL
st_view(y1,.,"$depvar1")
st_view(y2,.,"$depvar2")
st_view(X1,.,"$xvars1")
st_view(X2,.,"$xvars2")
st_view(Z1,.,"$zvars1")
st_view(Z2,.,"$zvars2")
/* obtain number of individuals */
N = rows(y1)
/* Define y */
for (i=1;i<=N;i++) {
 if (i==1) y=(y1[i]\y2[i])
 else y = (y\(y1[i]\y2[i]))
}
```

```
/* Define X */
for (i=1;i<=N;i++) {
 if (i==1) X=blockdiag(X1[i,],X2[i,])
 else {
  X = (X\blockdiag(X1[i,],X2[i,]))
 }
}
/* Define Z */
for (i=1;i<=N;i++) {
 if (i==1) Z=blockdiag(Z1[i,],Z2[i,])
 else {
  Z = (Z\blockdiag(Z1[i,],Z2[i,]))
 }
}
/* Start with an identity matrix weighting matrix */
Wopt = I(cols(Z))
/* perform GMM optimisation */
***Define evaluator function
function oiv_gmm0(transmorphic M, real scalar todo,
 real rowvector b, q,S) {
 y1 = moptimize_util_depvar(M,1)
 y2 = moptimize_util_depvar(M,2)
 xb1 = moptimize_util_xb(M,b,1)
 xb2 = moptimize_util_xb(M,b,2)
 st_view(Z1,.,moptimize_util_userinfo(M, 2))
 st_view(Z2,.,moptimize_util_userinfo(M, 4))
 N = rows(y1)
```

```
 /* Define y */
 for (i=1;i<=N;i++) {
  if (i==1) y=(y1[i]\y2[i])
  else y = (y\(y1[i]\y2[i]))
 }
/* Define Z */
for (i=1;i<=N;i++) {
 if (i==1) Z=blockdiag(Z1[i,],Z2[i,])
 else {
  Z = (Z\blockdiag(Z1[i,],Z2[i,]))
 }
}
 e = y - xb
 q = Z'*e
}
** initialise moptimize
M = moptimize_init()
moptimize_init_evaluator(M, & oiv_gmm0())
moptimize_init_evaluatortype(M, "q0")
moptimize_init_ndepvars(M, 2)
moptimize_init_depvar(M, 1, "y1")
moptimize_init_depvar(M, 2, "y2")
moptimize_init_eq_indepvars(M, 1, "x1 x2 x3")
moptimize_init_eq_indepvars(M, 2, "x1 x3")
moptimize_init_userinfo(M, 1, ("x1", "x2", "x3","one"))
moptimize_init_userinfo(M, 2, ("x2", "x3","z1","z2","z3","one"))
moptimize_init_userinfo(M, 3, ("x1", "x3","one"))
moptimize_init_userinfo(M, 4, ("x3","z1","z2","z3","one"))
moptimize_init_gnweightmatrix(M,Wopt)
moptimize_init_which(M, "min")
```

```
moptimize_init_technique(M, "gn")
moptimize(M)
***get estimates
b_gmm = moptimize_result_coefs(M)'
u_gmm = y - X*b_gmm
***get estimate of optimal W
sel11 = J(N,1,1)
sel12 = J(N,1,0)
for (i=1;i<=N;i++) {
 if (i==1) sel1=(sel11[i] \sel12[i])
 else sel1 =(sel1\(sel11[i]\sel12[i]))
}
u1_gmm=NULL
st_select(u1_gmm,u_gmm,sel1)              // get residual in eq 1
sel21 = J(N,1,0)
sel22 = J(N,1,1)
for (i=1;i<=N;i++) {
 if (i==1) sel2=(sel21[i]\sel22[i])
 else sel2 =(sel2\(sel21[i]\sel22[i]))
}
u2_gmm=NULL
st_select(u2_gmm,u_gmm,sel2)              // get residual in eq 1
```

```
for (i=1;i<=N;i++) {              // get N^(-1) Sum(Zi'uiui'Zi)
 if (i==1) {
  Zi = blockdiag(Z1[i,],Z2[i,])
  ui = u1_gmm[i]\u2_gmm[i]
  D = Zi'*ui*ui'*Zi
 }
 else {
  Zi = blockdiag(Z1[i,],Z2[i,])
  ui = u1_gmm[i]\u2_gmm[i]
   D = D + Zi'*ui*ui'*Zi
 }
}
D = (1/N)*D
Wopt = invsym(D)
*** re-optimize
M = moptimize_init()
moptimize_init_evaluator(M, & oiv_gmm0())
moptimize_init_evaluatortype(M, "q0")
moptimize_init_ndepvars(M, 2)
moptimize_init_depvar(M, 1, "y1")
moptimize_init_depvar(M, 2, "y2")
moptimize_init_eq_indepvars(M, 1, "x1 x2 x3")
moptimize_init_eq_indepvars(M, 2, "x1 x3")
moptimize_init_userinfo(M, 1, ("x1", "x2", "x3","one"))
moptimize_init_userinfo(M, 2, ("x2", "x3","z1","z2","z3","one"))
moptimize_init_userinfo(M, 3, ("x1", "x3","one"))
moptimize_init_userinfo(M, 4, ("x3","z1","z2","z3","one"))
moptimize_init_gnweightmatrix(M,Wopt)
moptimize_init_which(M, "min")
```

```
moptimize_init_technique(M, "gn")
moptimize(M)
*** get GMM estimes
b_gmm = moptimize_result_coefs(M)'
u_gmm = y - X*b_gmm
/* Calculate Covariance Matrix */
*** get estimate of optimal W at GMM estimates
sel11 = J(N,1,1)
sel12 = J(N,1,0)
for (i=1;i<=N;i++) {
 if (i==1) sel1=(sel11[i]\sel12[i])
 else sel1 =(sel1\(sel11[i]\sel12[i]))
}
u1_gmm=NULL
st_select(u1_gmm, u_gmm,sel1)                // get residual in eq 1
sel21 = J(N,1,0)
sel22 = J(N,1,1)
for (i=1;i<=N;i++) {
 if (i==1) sel2=(sel21[i]\sel22[i])
 else sel2 =(sel2\(sel21[i]\sel22[i]))
}
u2_gmm=NULL
st_select(u2_gmm,u_gmm,sel2)                // get residual in eq 1
```

```
for (i=1;i<=N;i++) {    // get N^(-1) Sum(Zi'uiui'Zi)
 if (i==1) {
  Zi = blockdiag(Z1[i,],Z2[i,])
  ui = u1_gmm[i]\u2_gmm[i]
  D = Zi'*ui*ui'*Zi
 }
 else {
  Zi = blockdiag(Z1[i,],Z2[i,])
  ui = u1_gmm[i]\u2_gmm[i]
   D = D + Zi'*ui*ui'*Zi
 }
}
Wopt = invsym(D)
/* Calculate covariance matrix */
V_gmm = invsym((X'*Z)*Wopt*Z'X)
/* parse results to stata */
moptimize_result_post(M, "robust")
st_matrix("b_gmm", b_gmm')
st_matrix("V_gmm", V_gmm)
end
/* post estimates to stata */
program drop _all
program mypost, eclass
ereturn repost b=b_gmm V=V_gmm
end
mypost
```

```
/* display results */
eret di
```

	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
eq1						
x1	1.771437	.1170327	15.14	0.000	1.542057	2.000817
x2	2.181796	.1573859	13.86	0.000	1.873325	2.490267
x3	1.855917	.1460442	12.71	0.000	1.569676	2.142158
_cons	.7202702	.1573938	4.58	0.000	.411784	1.028756
eq2						
x1	3.966332	.08466	46.85	0.000	3.800402	4.132263
x3	3.914237	.1213175	32.26	0.000	3.676459	4.152015
_cons	.2945812	.1254364	2.35	0.019	.0487304	.540432

Same thing with Stata's gmm

```
#delimit ;
gmm (y1 - {b1}*x1 - {b2}*x2 - {b3}*x3 - {b0})
 (y2 - {g1}*x1 -  {g2}*x3 - {g0}),
 instruments(1: x2 x3 z1 z2 z3)
 instruments(2: x3 z1 z2 z3)
 wmatrix(robust) winitial(identity);
#delimit cr
  (output omitted)
GMM estimation

Number of parameters =   7
Number of moments    =  11
Initial weight matrix: Identity                     Number of obs   =       100
GMM weight matrix:     Robust

------------------------------------------------------------------------------
             |               Robust
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         /b1 |   1.771437   .1170393    15.14   0.000     1.542044     2.00083
         /b2 |   2.181796    .157495    13.85   0.000     1.873112    2.490481
         /b3 |   1.855917   .1460756    12.71   0.000     1.569614     2.14222
         /b0 |   .7202702   .1574271     4.58   0.000     .4117188    1.028822
         /g1 |   3.966332   .0846927    46.83   0.000     3.800338    4.132327
         /g2 |   3.914237   .1213739    32.25   0.000     3.676348    4.152125
         /g0 |   .2945812   .1254592     2.35   0.019     .0486857    .5404767
------------------------------------------------------------------------------
Instruments for equation 1: x2 x3 z1 z2 z3 _cons
Instruments for equation 2: x3 z1 z2 z3 _cons
```

References

- Cameron, C.A.; Trivedi, P.K. (2005). Microeconometrics: Methods and Applications. Cambridge University Press.
- Cameron, C.A.; Trivedi, P.K (2010). Microeconometrics Using Stata, Revised Edition. Stata Press.
- Hansen, LP. (1982). Large sample properties of generalised method of moments estimators. *Econometrica* 50: 1029-1054.
- White, H. (1982). Instrumental variable estimation with independent observations. *Econometrica* 50: 483-499.
- Wooldridge, J.M. (2010). Econometric Analysis of Cross Section and Panel Data (2nd edition). The MIT Press.