Using Structures and Pointers in Mata to Estimate Panel Data Models with Attrition

Pierre Hoonhout
18 Sep 2015, Stata Users Group Meeting
1. Introduction

- Most panel data suffer from attrition
- In practical work, not much is done to deal with this problem (MCAR assumption).
- If attrition is taken into account, usually MAR (Selection on observables) is assumed.
- The reason is that it is much more difficult (read: impossible) to correct for Non-ignorable attrition (selection on unobservables).
- If refreshment samples are available, something can be done.
- The Stata-command `attrition_model` aims to make correcting for non-ignorable attrition easier.
- Its implementation shows the usefulness of structures and pointers in Mata.
2a. Panel Data with Attrition

The attrition problem can be visualized as follows:

<table>
<thead>
<tr>
<th>sub-population</th>
<th>$Z_1$</th>
<th>$Z_2$</th>
<th>$Z_3$</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balanced Panel 3 (BP3)</td>
<td>observed</td>
<td>observed</td>
<td>observed</td>
<td>obs</td>
</tr>
<tr>
<td>Incomplete Panel 3 (IP3)</td>
<td>observed</td>
<td>observed</td>
<td>.</td>
<td>obs</td>
</tr>
<tr>
<td>Incomplete Panel 2 (IP2)</td>
<td>observed</td>
<td>.</td>
<td>.</td>
<td>obs</td>
</tr>
</tbody>
</table>

Just-identification of the population distribution (MAR):

\[
\Pr(D_2 = 1|Z^2) = G_2(k_{20}(x) + k_{21}(Z_1, x))
\]

\[
\Pr(D_3 = 1|Z^3) = G_2(k_{30}(x) + k_{31}(Z^2, x))
\]
2b. The Problem With Non-ignorable Attrition

- For a two-period panel, sometimes following attrition-model is used (HW):

\[ \Pr(D_2 = 1|Z^2) = G_2(k_{20}(x) + k_{20}(Z_2, x)) \]

- This model tries to allow for non-ignorable attrition. However, it is just-identified (like MAR).
- The HW and MAR models are therefore observationally equivalent in two-period panels.
- You can use HW, but an observationally equivalent solution can be derived from MAR.
- For this reason, there is no good reason to use non-ignorable
2c. Refreshment Samples

Additional information in the form of refreshment samples helps:

<table>
<thead>
<tr>
<th>sub-population</th>
<th>$Z_1$</th>
<th>$Z_2$</th>
<th>$Z_3$</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balanced Panel 3 (BP3)</td>
<td>observed</td>
<td>observed</td>
<td>observed</td>
<td>obs</td>
</tr>
<tr>
<td>Incomplete Panel 3 (IP3)</td>
<td>observed</td>
<td>observed</td>
<td></td>
<td>obs</td>
</tr>
<tr>
<td>Incomplete Panel 2 (IP2)</td>
<td>observed</td>
<td></td>
<td></td>
<td>obs</td>
</tr>
<tr>
<td>Refreshment Sample 2 (RS2)</td>
<td></td>
<td>observed</td>
<td></td>
<td>obs</td>
</tr>
<tr>
<td>Refreshment Sample 2 (RS2)</td>
<td></td>
<td></td>
<td>observed</td>
<td>obs</td>
</tr>
</tbody>
</table>
2d. Identification With Refreshment Samples (SAN)

- Hoonhout and Ridder (2016) show that the SAN model just-identifies the population distribution:

\[
Pr(D_2 = 1 | Z^2) = G_2(k_{20}(x) + k_{21}(Z_1, x) + k_{20}(Z_2, x))
\]

\[
Pr(D_3 = 1 | Z^3) = G_2(k_{30}(x) + k_{31}(Z^2, x) + k_{32}(Z_3, x))
\]

- SAN stands for Sequential Additively Non-ignorable.
- This generalizes the two-period panel result of Hirano, Imbens, Ridder and Rubin (Econometrica, 2001).
3a. Estimation of $\theta$ (if $k(\cdot)$'s known)

- Hoonhout (2016) proposes an estimator for SAN attrition models. This estimator estimates a vector of parameters $\theta$, defined by a set of moment-conditions $E[m(Z^3; \theta)] = 0$, free of attrition bias.
- It is a weighted GMM estimator, that solves (in the just-identified case):

$$\frac{1}{n} \sum_{i=1}^{n} \left[ \frac{\Pr(D^3 = 1)}{G_3(\cdot)G_2(\cdot)} m(Z_i^3; \theta)I_i(BP3) \right] = 0$$

- This estimator is consistent, because

$$f(Z^3) = \frac{\Pr(D^3 = 1)}{\Pr(D^3 = 1|Z^3)} f(Z^3|D^3 = 1).$$
3b. How to Estimate the k-functions?

- The only problem is estimation of the k-function.
- Under MAR is easy: probit $D_2 Z_1$ and probit $D_3 D_2 Z_1 Z_2$ (or GMM equivalent)
- SAN is difficult: probit $D_2 Z_1 Z_2$ is not possible, as $Z_2$ is only partially observed. probit $D_3 D_2 Z_1 Z_2 Z_3$ idem.
- Hoonhout and Ridder derive an information-theoretic interpretation for the SAN model. This implies that the k-functions satisfy a set of integral equations (infinitely many moment conditions).
- In order to get a finite number of moment equations, we discretize $Z$. We will denote the resulting set of dummy variables by $i.Z$, for notational convenience.
- We can use the resulting weights for weighted-GMM estimation. No discretization of $Z$ is required in the moment conditions for the
3c. Moment Equations for $k_{20}, k_{21}, k_{22}$

If we discretize the (continuous) $Z$ in two groups (one dummy $i.Z$), we obtain the following sample moment equations for the two-period panel:

\[
\Sigma_{z_1} \Sigma_{z_2} \frac{\Pr(D^3 = 1)}{\Pr(D_2 = 1|i.z^2; \alpha_2)} f(i.z^2|D^2 = 1) = 1
\]

\[
\Sigma_{z_2} \frac{\Pr(D^2 = 1)}{\Pr(D^2 = 1|i.z^2; \alpha_2)} f(i.z^2|D^2 = 1) = \bar{f}(i.z_1) \quad \forall i.z_1
\]

\[
\Sigma_{z_1} \frac{\Pr(D^2 = 1)}{\Pr(D^2 = 1|i.z^2; \alpha_2)} f(i.z^2|D^2 = 1) = \bar{f}(i.z_2) \quad \forall i.z_2
\]

where

\[
\Pr(D_2 = 1|i.z^2; \alpha_2) = G_2(\alpha'_2 i.z^2).
\]
This requires a lot of book-keeping...

3d. Moment Equations for $k_{30}, k_{31}, k_{32}$

A three-period panel also has the following sample moment equations (etcetera):

$$\sum_{z_1} \sum_{z_2} \sum_{z_3} \frac{\Pr(D^3 = 1)}{\Pr(D^3 = 1 | i. z^3; \alpha_3)} f(i. z^3 | D^3 = 1) = 1$$

$$\sum_{z_1} \frac{\Pr(D^3 = 1)}{\Pr(D^3 = 1 | i. z^3; \alpha_3)} f(i. z^3 | D^3 = 1) = \tilde{f}(i. z^2) \ \forall i. z^2$$

$$\sum_{z_1} \sum_{z_2} \frac{\Pr(D^3 = 1)}{\Pr(D^3 = 1 | i. z^3; \alpha_3)} f(i. z^3 | D^3 = 1) = \tilde{f}(i. z^3) \ \forall i. z_3$$

where

$$\Pr(D^3 = 1 | i. z^3; \alpha_3) = G_3(\alpha'_3 i. z^3) G_2(\alpha'_2 i. z^2).$$

This requires a lot of book-keeping...
3e. Obtaining Good Starting Values

1. initialization: get wave-2 MAR_ML estimates (using logit).
2. get wave-2 MAR_GMM estimates using MAR_ML as starting values.
3. get wave-2 SAN_GMM starting values using fixed $k(z^2)$ to estimate $\alpha_3$ (coordinate ascent method).
4. get wave-2 SAN_GMM estimates using SAN_GMM starting values
5. get wave-3 SAN_GMM estimates in the same way (using wave-2 estimates as starting values). do this until wave $T$.
6. get $\theta$ estimates using "known" weights.
7. get $\alpha_2, \alpha_3, ..., \alpha_T, \theta$ estimates using all earlier estimates as starting values.

This requires a lot of book-keeping...
4a. The attrition_model commands

- **attrition_model specify** allows the user to specify the attrition models for each wave with attrition.
- **attrition_model estimate** estimates the k-functions (piecewise constant) and $\theta$ (simultaneously). No starting values are required.
- **attrition_model graph** provides a graph of intermediate and final estimates.
4b. attrition_model specify

```plaintext
local k20 = ""
local k21 = i.z1
local k22 = i.z2

local k30 = ""
local k31 = i.z1##i.z2
local k32 = i.z3

attrition model specify ///
  (attr2: (\k20\) (\k21\) (\k22\), data=CP2RS2) ///
  (attr3: (\k30\) (\k31\) (\k32\), data=CP3RS2RS3) ///
  (pop: &pop_mef, options)
```
4c. Mata Structures and Pointers

- **Structures**: allows you to organize a collection of several scalars, vectors and matrices. The resulting structure object can be passed to other routines.

- **Pointers**: a pointer is an object that contains the address of another object:

```mata
.x = (1, 2, 3, 4)
p = &x
.p
   0x6000037ff6c8
.*p
   1  2  3  4
```

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
4d. The structure attrition_model (1)

mata:
struct attrition_model {
    // 1. basic information:
    real scalar T  //number of waves
    real matrix N  //rows: N_P, N_CP, N_BP, N_IP, N_RS
        //columns: t=2, t=3, ..., t=T

    // 2. information about the k-functions
    // k20, k21, k22, k30, k31, k32, ...
    pointer(transmorphic matrix) matrix k_info
    // assign: k_info[i,j]=&J(5,5,1) dereference: *(k_info[i,j])
    // columns (3*(T-1))+1: (k20,k21,k22), (k30,k31,k32), ..., (pop)
}
end
4e. The structure attrition_model (2)

- The structure includes a variable that is of type "matrix of pointers."
- Each **column** of this matrix describes a k-function. That is, if $T = 3$ the columns are $k_{20}, k_{21}, k_{22}, k_{30}, k_{31}, k_{32}$.
- Each **row** describes particular information for each k-function.
  - For instance, the first row contains the parameter-names of $\alpha_{20}, \ldots, \alpha_{32}$.
  - The second row stores the estimated values (for use in the estimation in later waves).
  - Another row contains the MAR estimates of the k-function parameters (estimated at the time of initialization of the structure, within attrition_specify).
- This structure facilitates the book-keeping enormously.
4f. Initializing the Structure

Once the user specifies the model, the number of waves $T$ is known. `attrition_model specify` calls the following function:

```plaintext
mata
    struct attrition_model scalar function
    init_attrition_model(real scalar T) {
        struct attrition_model scalar aM
        real scalar i, j

        aM.T = T
        aM.N = J(5, T-1, .) // rows: N_P, N_CP, N_BP, N_IP, N_RS
                          // columns: t=2, t=3, ..., t=T
        aM.k_info = J(4, 3*(T-1) + 1, NULL)
        // now, the dimensions are known.
        // assignment is now possible using aM.k_info[i,j] = &A
    }
```
```
4g. Accessing Values of a Structure

Instances of a structure cannot be accessed interactively:

```c
transmorphic get_aM(struct attrition_model scalar aM,
                     string scalar mata_obj, real scalar i, real scalar j){
    if (mata_obj == "T") return(aM.T)
    if (mata_obj == "N") return(aM.N[i,j])
    if (mata_obj == "k_info") {
       return(*((aM.k_info[i,j])))
    }
}
```

For changing the values you need a similar function (set_aM)
4h. attrition_model estimate

- Many estimations are done here, before arriving at the final estimates.
- All the estimations are similar but different.
- The idea is to write a single moment-evaluator-function. This moment-evaluator function morphs automatically into the moment-evaluator-function that is required for the current estimation.
- This can be achieved because the `gmm-command allows us to pass extra arguments to the moment-evaluator function`. We will simply pass the structure aM (of type attrition_model) to the evaluator function. With this information it can morph as required.
5. Conclusion

The attrition_model command provides a relatively straightforward way to obtain panel-data model estimates that are corrected for (potentially non-ignorable) attrition. The user can specify each of the k-functions separately, to keep the number of nuisance parameters within bounds.

1. All the book-keeping in attrition_model estimate is relegated to one or more instances of a structure. This structure is passed to the moment-evaluator function.

2. This structure contains a matrix of pointers. The columns of that matrix does the book-keeping for one k-function.