Processing dyad data in Stata

Brendan Halpin

April 2018

1 Processing dyads in Stata

Sometimes when you are working with nested data (such as household surveys, with data on all individuals in the household), analysis focuses on dyads (such as spouse pairs) rather than individual cases. This means you need to link data in one observation with that in another. As long as the data includes information in ego's record about where alter's record is (e.g., by holding alter's ID as a variable), the simplest way to do this is to create a separate data file, where the alter ID variable is renamed to ID, and the substantive variables are also renamed, and to match it back in to the original data. This is not terribly difficult, but it is messy, so I present here a more convenient method.

First, an example using the standard approach, and the wave 18 BHPS. The BHPS is a household survey where each record represents an individual, and in theory each adult member of the household is surveyed. Each individual has a unique ID, `pid`. For individuals whose spouse is in the survey (and therefore probably in the data set), their spouse's ID is stored in `osppid`.

```
use osppid osex ojbstat using /home/data/bhps/oindresp
tempfile spousedata
keep if osppid!=0 // Drop cases where no spouse reported
rename (osppid osex ojbstat) (pid spsex spjbstat)
save ‘spousedata’, replace
use pid osppid osex ojbstat using /home/data/bhps/oindresp
merge 1:1 pid using ‘spousedata’
keep if _merge!=2 // Drop people reported as alters who are not present as egos
```

This code first loads alter-ID and two substantive variables, renames them (renaming alter-ID to the same name as ego-ID), and saves to a temporary file. The file thus contains information about ego keyed to alter's ID: if we consider it from alter's point of view it consists of information about alter's alter keyed on ID (for spouse pairs the relationship is symmetric, but in general it reverses the relationship: if ego is the parent and alter the child, this file contains information about the individual's parent). It then loads ego-ID, alter-ID and the substantive variables again, and does a merge. It drops cases which are present only in the alter file (these are people whose ID is reported as spouses, who are not present in the file, due typically to non-response).

Here we see the result, crosstabulating ego and alter sex: nearly (but not quite) everyone is reporting heterosexual relationships:

```
. tab osex spsex
```

sex	sex: male	sex: female	Total
male	40	4,513	4,553
female	4,513	26	4,539
Total	4,553	4,539	9,092

My alternative involves using a custom program to find the row number of alter's record, and is more concise:

```
use pid osppid osex ojbstat using /home/data/bhps/oindresp, clear

dyadid pid osppid, gen(idx)
gen spsex2 = osex[idx]
gen spjbstat = ojbstat[idx]
```

The results are identical.

```
. tab osex spsex2
```

sex	spsex2 1	2	Total
male	40	4,513	4,553
female	4,513	26	4,539
Total	4,553	4,539	9,092

2 The program

In the example the main work is obscured, as it takes place in the `dyadid` command. This command uses Mata's associative arrays to create a new variable, which is the case number of the spouse record. Effectively, the Mata code passes through the data twice, first creating in an `asarray` a record of the case number for each observed ego-ID, and then plugging in each alter-ID into the same array to pull out the corresponding case number.

```
mata:

real matrix function dyadid (string idvar, string dyadidvar, string genvar) {
  st_view(id = ., ., (idvar))
  st_view(dyadid = ., ., (dyadidvar))
  st_view(gen = ., ., (genvar))

  nobs = length(dyadid)

  altindex = asarray_create("real")
  "Build AS-array"
  for (i=1; i<=nobs; i++) {
    asarray(altindex,id[i],i)
  }
  "Read AS-array"
  for (i=1; i<=nobs; i++) {
    if (asarray_contains(altindex,dyadid[i])) {
      gen[i] = asarray(altindex,dyadid[i])
    }
    else {
      gen[i] = .
    }
  }
  "Done"
}

end
```

```
program dyadid
syntax varlist(min=2 max=2), gen(string)
tokenize `varlist'

/* // Check that alter-ID is unique if not missing */
/* preserve */
/* keep if !missing(`2') */
/* isid `2' */
/* restore */

qui gen `gen' = .
mata dyadid("`1'", "`2'", "`gen'")
end

/*

With dyadic data, given ID (not necessarily unique) and alter-ID
(unique, but potentially missing), where alter-ID is the ID of the
partner, generate an index variable which is the row number of the
partner's record

. dyadid id spid, gen(idx)
. gen spempstat = empstat[idx]

*/
```

The syntax is

```
dyadid egoID alterID, gen(indexvar)
```

The ego-ID does not need to be unique, but the alter-ID should be (though it can be missing). However, if there are duplicates in alter-ID it won't provoke an error, but only the last occurrence will be recorded. Where there is no alter, or where alter's ID is not present in the data as an ego-record, the index variable will be missing.

To recap, the sort of data this is intended for includes records for both ego and alter, keyed on an ID variable, and linked by a variable that contains alter's ID. We link from ego to alter by finding the case number of the ego-record corresponding to the alter-ID variable.

3 Implications for SADI

I plan to extend some of my SADI sequence distance measures to use this mechanism to create dyadic distance variables, rather than square pairwise matrices. This means it is much more efficient with large data sets, if only dyadic distances are needed. Let me know if this interests you.

4 Installation

The code is currently available on my site, but I will upload to SSC in due course. For now:

```
. net from http://teaching.sociology.ul.ie/statacode
. net install dyadid
. net get dyadid
```