17.871

How to Use the STATA **merge** and **reshape** commands

Most of the projects done in 17.871, and in fact most interesting research, require combining data sets. This handout reviews using the most valuable command for managing multiple data sets, the **merge** command. In addition, we are often interested in combining multiple observations from some unit of analysis (like countries or states or people) to create a panel data set. The **reshape** command helps to move between different organizations of the data.

The **merge** command

Let us say we are researching the effect of using different voting equipment on the tendency of voters to cast "blank" or "spoiled" ballots (that is, to record no vote, or to record multiple votes). One data set might record the number of votes cast and the number of blank ballots in each town. Here is an example:

Exhibit 1. ballots.dta

town	blank92	ballot92	blank96	ballot96
Barnstable	1163	22747	232	22467
Bourne	125	7885	61	8032
Brewster	42	5480	62	5498
Chatham	35	4558	103	4475
Dennis	294	8732	86	8493
Eastham	20	3013	26	3151
Falmouth	381	16377	169	16224
Harwich	184	6741	89	6737
Mashpee	22	4619	46	4962
Orleans	109	4251	113	4229
Provincetown	15	2488	6	2268
Sandwich	35	9151	62	9757
Truro	3	1161	7	1156
Wellfleet	14	1815	24	1768
Yarmouth	441	12862	104	12710

The variable town identifies a town (duh!), *blank92* and *blank96* record the number of blank ballots in these town for 1992 and 1996, and *ballot92* and *ballot96* record the total number of ballots cast in the town. Let us suppose we have saved this data set in the file ballots.dta

Another data set might record the ballot method used in the town, like this:

Exhibit 2. machines.dta

town	method92	method96
Barnstable	Paper	AccuVote
Bourne	AccuVote	AccuVote
Brewster	Optech	Optech
Chatham	Paper	Optech
Dennis	Optech	Optech
Eastham	Paper	Optech
Falmouth	Paper	AccuVote
Harwich	Optech	Optech
Mashpee	Paper	Optech
Orleans	Optech	Optech
Provincetown	Paper	Paper
Sandwich	Optech	Optech
Truro	Paper	Paper
Wellfleet	Paper	Paper
Yarmouth	Optech	Optech

As before, *town* is the town. The variables *method92* and *method96* record the type of voting device the town used in 1992 and 1996, respectively. The data are saved in machines.dta.

Notice the following important point: Both ballots.dta and machines.dta have a common variable, town, which uniquely identifies the cases. (In this example, the town name identifies the cases. For larger, more complex cases, it is often advisable to use a numerical variable as the joint identifier.)

To merge two data sets, follow these steps:

(1) Sort both data sets on the common identifying variable and save them to disk sorted.
(2) Use one of the data sets.
(3) Issue the merge command, using the following syntax:
merge 1:1 ***commonvariable*** **using** ***remotefilename***

This set of commands will augment the data set you had used (in step 2), by adding the variables from the remote file. In addition, a new variable will be created, called *_merge.* The variable *_merge* will be equal to 3 if the case was in both data sets, 1 if the case was in the "master" data set (i.e., the one used in step 2) but not in the "using" data set, and 2 if the case was in the "using" data set but not originally in the "master" data set.

(I will address what the "1:1" notation means in a bit.)

So, the following commands

use ballots
sort town
save ballots,replace
use machines
sort town
save machines,replace3
use ballots
merge 1:1 town using machines

would produce the following data set:

Exhibit 3. Merged data set

town	blank92	ballot92	blank96	ballot96	method92	method96	_merge
Barnstable	1163	22747	232	22467	Paper	AccuVote	3
Bourne	125	7885	61	8032	AccuVote	AccuVote	3
Brewster	42	5480	62	5498	Optech	Optech	3
Chatham	35	4558	103	4475	Paper	Optech	3
Dennis	294	8732	86	8493	Optech	Optech	3
Eastham	20	3013	26	3151	Paper	Optech	3
Falmouth	381	16377	169	16224	Paper	AccuVote	3
Harwich	184	6741	89	6737	Optech	Optech	3
Mashpee	22	4619	46	4962	Paper	Optech	3
Orleans	109	4251	113	4229	Optech	Optech	3
Provincetown	15	2488	6	2268	Paper	Paper	3
Sandwich	35	9151	62	9757	Optech	Optech	3
Truro	3	1161	7	1156	Paper	Paper	3
Wellfleet	14	1815	24	1768	Paper	Paper	3
Yarmouth	441	12862	104	12710	Optech	Optech	3

The *_merge* variable confirms that we brought in a balanced set of cases from both data sets. (In other words, it confirms that there aren't towns that we have election returns for that we don't have voting equipment data for, and vice versa.) However, it's a nuisance variable once the merge has been successfully completed, so it's a good idea to **drop** it once you've satisfied yourself that everything is OK.

Let me return to the merge syntax that I gave above:

merge 1:1 ***commonvariable*** **using** ***remotefilename***

What's this "1:1" thing? The thing that looks like a ratio in the command line tells Stata how many records Stata should expect to match up between the "master" dataset (i.e., the one in memory) and the "using" dataset (i.e., the one on disk). The example above tells Stata that there is a 1-to-1 correspondence between observations in the two files. In other words, there is a unique value of *town* in each dataset. This little bit of syntax does three things. First, it helps Stata do its computation more efficiently. Second, it allows Stata to alert you if there are extraneous lines of duplicated data in one or both of the datafiles. (For instance, if Stata finds two lines of data in **machines.dta**, it will issue an error message and tell you that there are multiple observations in the "using" dataset associated with at least one value of *town*, although you said there would be only one.) This alerts you to either errors in the dataset, or errors in your understanding of the dataset. Third, it allows Stata to drive you totally batty, if you have a really big dataset that has duplicate records.

There are four possibilities for this first bit of code in the **merge** commend, as follows:

- 1:1 specifies a one-to-one merge on the key variables in the two datasets
- m:1 specifies a many-to-one merge on the key variables in the two datasets
- 1:m specifies a one-to-many merge on the key variables in the two datasets
- m:m specifies a many-to-many merge on the key variables in the two datasets

By far, the most common choices are either m:1 or 1:m. For instance, say you have a file, called **countries.dta** that has economics data about countries, with a variable *country_name* that uniquely identifies each country. In addition, there is a variable recording the European country that had once colonized that country; the name of the variable is *colonizer*. You have another file, called **colonies.dta,** which records basic information about each European country that had held colonies — this might include data about legal principles encoded in the colonizer's traditions. This file also has a variable name called *colonizer* that records the relevant legal information.

In this situation, we assume that there will be a bunch of countries in **countries.dta**, but a smaller number of countries in **colonies.dta.** Therefore, if you wanted to merge **countries.dta** with **colonies.dta**, you would issue the following command:

merge m:1 *colonizer* using *colonies.dta*

The **reshape** command

The data set shown in Exhibit 3 could be used to analyze the rate of ballot blanking in 1992 and 1996, separately, as a function of voting technology used. And, it could be used to study the change in ballot blanking from 1992 to 1996, as a function of the *change* in voting technology. However, we might also want to treat the data set as consisting of 30 town-year observations, rather than 15 separate town observations. To do this, I would want to "stack up" the different town observations, having the 1992 variables for blanks, ballots, and method appearing first (and identified according to year), followed by the 1996 variables for blanks, ballots and method (also identified according to year). We can do this using the **reshape** command.

Notice first the naming convention I adopted in the above data sets: each substantive variable name consists of a stem (*blank* or *ballot*) followed by the year in question (92 or 96).

Then, the following command

reshape long blank ballot method, i(town) j(year)

will "reshape" the Exhibit 3 data set as follows:

Exhibit 4. Reshaped, merged data set

town	blank	ballot	method	year
Barnstable	1163	22747	Paper	92
Bourne	125	7885	AccuVote	92
Brewster	42	5480	Optech	92
Chatham	35	4558	Paper	92
Dennis	294	8732	Optech	92
Eastham	20	3013	Paper	92
Falmouth	381	16377	Paper	92
Harwich	184	6741	Optech	92
Mashpee	22	4619	Paper	92
Orleans	109	4251	Optech	92
Provincetown	15	2488	Paper	92
Sandwich	35	9151	Optech	92
Truro	3	1161	Paper	92
Wellfleet	14	1815	Paper	92
Yarmouth	441	12862	Optech	92
Barnstable	232	22467	AccuVote	96
Bourne	61	8032	AccuVote	96
Brewster	62	5498	Optech	96
Chatham	103	4475	Optech	96
Dennis	86	8493	Optech	96
Eastham	26	3151	Optech	96
Falmouth	169	16224	AccuVote	96
Harwich	89	6737	Optech	96
Mashpee	46	4962	Optech	96
Orleans	113	4229	Optech	96
Provincetown	6	2268	Paper	96
Sandwich	62	9757	Optech	96
Truro	7	1156	Paper	96
Wellfleet	24	1768	Paper	96
Yarmouth	104	12710	Optech	96

We now have a new data set with twice as many observations as before: each town has two observations, one for 1992 and the other for 1996. For each town-year observation, we have the number of blanks, total number of ballots, and the voting method used for that observation. The command

reshape wide blank ballot method, i(town) j(year)

will reverse the procedure, restoring the data set to Exhibit 3 (without *_merge*).