Combining Data in Stata

There are two common ways of combining data: *appending* and *merging*. Think of appending as "stacking" data or "adding rows" to an existing dataset. Think of merging as "adding columns."

Appending data

Imagine you are working with a dataset called *states69_89.dta* that includes state-level school finance data spanning the years 1969 to 1989. You have a second dataset called *states90_05.dta* that includes similarly-defined data for the years 1990 to 2005. You would like to append the latter dataset to the former one, to create one working file that spans the full 1969-2005 period.

states69_89.dta

state	year	totalrev	strev	locrev	enrollment	…
AL	1969	471135	283934	100717	777123	…
AL	1970	504383	279613	111170	754014	…
AL	1971	547500	306936	122873	745818	…
AL	1972	563080	321691	132367	736000	…
…	…	…	…	…	…	…

states90_05.dta

state	year	totalrev	strev	locrev	enrollment	…
AL	1990	2704515	1625517	777684	682524	…
AL	1991	2823340	1659018	841745	694078	…
AL	1992	2982753	1732506	902867	696071	…
AL	1993	3121320	1850898	924176	687047	…
…	…	…	…	…	…	…

With the first dataset in memory, the following syntax will append the second one to the first:

```
append using states90_05.dta
```

or:

```
append using states90_05.dta, keep(varlist)
```

The command itself is straightforward. However, it is important that the variables in the appending dataset be of the same variable type as in the master dataset. For example, if *year* is numeric in the first dataset but string in the second, the appended values will be read in as missing. (Stata will return a message letting you know, but will proceed anyway). Another common error is having variables of the same name that are defined differently. For example, if *totalrev* is in thousands of dollars in the first dataset but whole dollars in the second, the combined variable will be inconsistently measured. (Stata has no way of catching these kinds of errors—you must be attentive to this).

The set of variables in the two datasets do not have to be identical. For example, if *states90_05* has additional variables that are not in *states69_89*, these will be included in the combined dataset, but will have missing values in the years 1969-89. You can use the "keep" option in the `append` command to specify which variables you would like to retain from the appending dataset.

Last updated: August 30, 2021

Merging data

There are several types of merges. In my experience, the one-to-one merge and many-to-one merges are the most common. Details for these types of merges are provided below.

- One-to-one merge (1:1)
- Many-to-one merge (m:1)
- One-to-many merge (1:m)
- Many-to-many merge (m:m)
- One-to-one merge (by observation) (1:1 _n)

One-to-one merge (1:1)

In the 1:1 merge, there is a variable or combination of variables in your "master" dataset (the one you are working with) and "using" dataset (the one you are merging in) that uniquely match their observations. For example, suppose you are working with a dataset called *students.dta* that contains basic demographic information about a cross-section of students. You would like to merge this dataset with another that contains test scores for these students (*test.dta*).

students.dta

id	Year	sex	grade	…
1	2012	M	4	…
2	2012	F	4	…
3	2012	F	5	…
4	2012	F	4	…
…	…	…	…	…

→

test.dta

id	math	ELA	science
2	82	84	62
3	71	80	69
5	93	91	72
6	85	85	50
…	…	…	…

In this case, *id* uniquely identifies students in both the *students* and *test* data. Notice that some students in the former dataset (e.g. 1, 4) are not found in the latter. This is OK. With the master dataset in memory (*students*), the following syntax will perform the merge with the "using" dataset (*test*):

```
merge 1:1 id using test.dta
```

If the two datasets conform to the requirements for a 1:1 merge, the command will work correctly. If the observations are *not* uniquely identified by *id*, the command will not work. The combined dataset includes all of the variables (and all of the observations) found in both the master and using datasets. The command creates a new variable called *_merge* that takes one of three values for each observation:

_merge = 1 if the observation appeared only in the master dataset
_merge = 2 if the observation appeared only in the using dataset
_merge = 3 if the observation appeared in both

I am often uninterested in cases that are not in the master dataset, so I `drop` those with *_merge* ==2. There are many other options that can be used with the merge command (see the Stata help file for merge). If I only wish to keep certain variables from the using dataset, I can include the option:

```
merge 1:1 id using test.dta, keepusing(varlist)
```

One-to-one merges can also be performed using a combination of variables that uniquely identify the observations. For example, if the *students* data included observations by student and year, there might be multiple observations per student (*id*), but only one observation by student/year. If the using dataset was similarly structured, I could merge on student and year as follows:

```
merge 1:1 id year using test.dta
```

Note that the merge command will only match in variables from the using data that do not appear in the master data. If, for example, *test* contains some variables with the same names as those in *students*, they will not be matched in from *test,* or affect the existing variables with the same name. Be attentive to this, as sometime the redundant variable is more complete in the "using" data than in the "master." This is an option that can be adjusted, as described later.

Many-to-one merge (m:1)

In the m:1 merge, there is a variable or combination of variables in your "using" dataset that can be used to uniquely match its observations to those in the "master" dataset. In this case the master dataset can have many observations with the same identifying variables; they do not have to be unique. For example, suppose you are working with a dataset called *testsall.dta* that contains student test scores for multiple subjects and years. You would like to merge this with another that contains basic demographic information about the student that does not change over time (*demographics.dta*).

testsall.dta

id	year	score	subject	…
1	2010	62	Math	…
1	2010	75	ELA	…
1	2010	71	Sci	…
2	2010	85	Math	…
…	…	…	…	…

demographics.dta

id	sex	race	…
1	M	W	…
3	F	H	…
4	M	W	…
5	F	B	…
…	…	…	…

In this case, *id* uniquely identifies students in the *demographics* data only. Notice that some students in the *testsall* data (e.g. 2) are not found in the *demographics* file. This is OK. With the master dataset in memory (*testsall*), the following syntax will perform the merge with the "using" dataset:

```
merge m:1 id using demographics.dta
```

If the two datasets conform to the requirements for a m:1 merge, the command will work correctly. The combined dataset will include all of the variables (and observations) found in both the master and using datasets. The m:1 merge also creates the *_merge* variable as described above. It is common in examples like this to drop observations where *_merge==2*. If you would like to do this automatically, add the option `keep(match master)` to the `merge` command.

One-to-one merge by observation (1:1 _n)

In the 1:1 merge by observation, the observations (rows) in the master and using dataset line up exactly. This type of merge is rare. As an example, imagine you are working with a dataset called *population.dta* that includes U.S. states and their populations. You would like to merge this with a dataset called *stategdp.dta* that contains gross domestic product by state. Both datasets are sorted in ascending order by state. (This is important, since Stata literally matches the datasets observation-by-observation in this case).

population.dta

state	population	…
AK	687455	…
AL	4718206	…
AR	2874554	…
AZ	6280362	…
…	…	…

stategdp.dta

state	GDP
AK	30809112
AL	160254948
AR	94460843
AZ	226564946
…	…

With the master dataset in memory (*population.dta*), the following syntax performs the 1:1 merge by observation. Note there is no key identifying variable specified in this case.

```
merge 1:1 _n using stategdp.dta
```

Updating and replacing

In some applications you will use the `merge` command to fill in missing values or update old data with values from the "using" dataset. For example, suppose you are performing a 1:1 merge between the *students* and *test* data as above. In this case, there are variables in the *test* data that share the same name (and are defined in the same way) as variables in the *students* data. In the illustration below, the *math* variable is in both datasets, but some values are missing in *students*.

students.dta

id	year	sex	grade	math	…
2	2012	M	4	75	…
3	2012	F	4	.	…
5	2012	F	5	.	…
6	2012	F	4	82	…
…	…	…	…		…

test.dta

id	math	ELA
2	82	84
3	71	80
5	93	91
6	85	85
…	…	…

If your intent is only to fill in missing values for math using *test.dta* (here, id numbers 3 and 5), you would add the `update` option to the `merge` command:

```
merge 1:1 id using test.dta, update keepusing(math)
```

The resulting merged dataset will fill in the missing values for *math* from the *test* dataset. The *_merge* variable will equal 4 in this case for observations with updated values. If you instead wish to replace all values of *math* in the *students* data with those in *test*, you would add both the `replace` and `update` options to the `merge` command:

```
merge 1:1 id using test.dta, replace update keepusing(math)
```

This command over-writes the values of *math* in *students* with those in *test*. The *_merge* variable will equal 4 for observations that were missing in the original but updated, and 5 for observations not missing in the original that were replaced with updated values from the using data.