How do I merge multiple files in Stata?

Authored by

stats writer

July 1, 2024

RECOMMENDED CITATION

stats writer (2024). *How do I merge multiple files in Stata?*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=163252

Merging multiple files in Stata refers to the process of combining two or more separate datasets into a single dataset. This can be done using the "merge" command in Stata, which allows for the merging of datasets based on a common variable. This command is useful for integrating different sources of data or for consolidating information from different time periods. In order to successfully merge files in Stata, it is important to ensure that the datasets have a shared variable and that the data is clean and consistent. By merging files, users can create a comprehensive dataset that can be used for further analysis and statistical procedures in Stata.

How can I merge multiple files in Stata? | Stata FAQ

NOTE: This page describes usage of an older version of the merge command (prior to Stata 11), which allowed multiple files to be merged in the same merge command. The current version of merge uses a different syntax (requiring a 1:1, m:1, or 1:m specification) and does not allow more than one file to be merged in a single merge command. However, the old syntax displayed on this page will still work in newer versions of Stata.

This FAQ is was based on a page developed by the Graduate Statistical Assistant Program at Boston College. We are grateful for their permission to reproduce this FAQ here.

It is not uncommon for data, especially survey data, to

come in multiple datasets (there are practical reasons for distributing datasets this way). When data is distributed in multiple files, the variables you want to use will often be scattered across several datasets. In order to work with information contained in two or more data files it is necessary to merge the segments into a new file that contains all of the variables you intend to work with.

First, you'll need to figure out which variables you need, and which datasets contain them, you can do this by consulting the codebook. In addition to finding the variables you want for your analysis, you need to know the name of the id variable. An id variable is a variable that is unique to a case (observation) in the dataset. For a given individual, the id should be the same across all datasets. This will allow you to match the data from different datasets to the right person. For

cross sectional data, this will typically be a single variable, in other cases, two or more variables are needed, this is commonly seen in panel data where subject id and date or wave are often needed to uniquely identify an observation. In order for Stata to merge the datasets, the id variable, or variables, will have to have the same name across all files. Additionally, if the variable is a string in one dataset, it must also be a string in all other datasets, and the same is true of numeric variables (the specific storage type is not important, as long as they are numerical). Once you have identified all the variables you need, and know what the id variable(s) are, you can begin to merge the datasets.

A simple example

A good first step is to describe our data. We can do this without actually opening file (this can be handy if the files are very

large), all we have to do is
open Stata and issue the command. The describe command gives us a
lot of useful information, for our purposes the most important things it
shows is that the variable id is numeric, and that the data are unsorted
(the data must be sorted by the id variable or variables in order to merge). We
also note that the variables we want from this dataset are in fact in the
dataset. We would want to do this for all three of our datasets, but to save
space we'll only show the output for one of the datasets. Lets assume that the datasets are
all unsorted and that the id variable has the same name (id) in all three datasets.

describe using http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data1

Contains data highschool and beyond (200 cases)

obs: 200 22 Jul 2008 13:47
vars: 4
size: 4,000
---
storage display value
variable name type format label variable label

---
id float %9.0g
female float %9.0g fl
race float %12.0g rl
ses float %9.0g sl

Sorted by:

(output ommitted)

describe using http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data2
describe using http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data3

**Since the datasets aren't sorted, we will need to open each dataset, sort it,
and then save the sorted dataset. Although we can use the data from a website
easily within Stata, we cannot save it there. So note that all of the use commands
pull datasets from our website, but save them to the directory "d:data" on the
users computer. The syntax below opens each dataset, sorts it by
id and then saves it in a new location with a new name. If the dataset
were already on our computer, we could save it in the same location, and,
possibly even under the same name (replacing the old dataset), this is the users
choice.**

```
use
http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data1, clear
sort id
save d:datadata1_a, replace
```

```
use
http://statistics.ats.ucla.edu/stat/data/stata_faq_multmer
ge/data2, clear
sort id
save d:datadata2_a, replace

use
http://statistics.ats.ucla.edu/stat/data/stata_faq_multmer
ge/data3, clear
sort id
save d:datadata3_a, replace
```

Next, we actually merge the datasets. The merge command merges corresponding
observations from the dataset currently in memory (called the master dataset)
with those from a different Stata-format dataset (called the using dataset) into
single observations. Assuming that we have data3 open from running the above syntax, that
will be our master dataset. The first line of syntax below merges the data. Directly after the merge command is the name of the
variable (or variables) that serve id variables, in this

**case id. Next is the
argument using this tells Stata that we are done listing the id variables, and
that what follows are the dataset(s) to be merged. The names are listed, with only spaces
(no commas, etc.) between them. (Note, if the names or paths of your
datasets include spaces, be sure to enclose them in quotation marks, i.e. " ".) The
next line of syntax saves our new merged dataset. Note that merge does not produce
output.**

```
merge id using d:datadata1_a d:datadata2_a
save d:datamerged_data
```

Now we can have a look at our newly merged dataset.

```
describe

Contains data from data3.dta
obs: 200 highschool and beyond (200 cases)
vars: 14 24 Jul 2008 15:54
size: 10,200 (99.0% of memory free)
-------------------------------------------------------------------------------
```

---

storage display value

variable name type format label variable label

---

id float %9.0g

schtyp float %9.0g scl type of school

prog float %9.0g sel type of program

female float %9.0g fl

race float %12.0g rl

ses float %9.0g sl

_merge1 byte %8.0g _merge representing data1

read float %9.0g reading score

write float %9.0g writing score

math float %9.0g math score

science float %9.0g science score

socst float %9.0g social studies score

_merge2 byte %8.0g _merge representing data2

_merge byte %8.0g

Sorted by:

In the above output we see the number of cases (200),

which is correct. This
is important since problems with the merge process often result in too few, or
more often too many, cases in the merged dataset. We also see a list of the variables,
which includes all the variables we want. The merged dataset contains three
extra variables.
These new variables are _merge, _merge1 and _merge2. The
command merge
will always generate at least one additional variable named _merge, when multiple
files are specified in using, the command will produce additional _merge* variables, one for each of
the datasets in the using list (in our case _merge1 and _merge2).
These variables tell us where each observation in the dataset came from, this is useful
as a check that your data merged properly. Sometimes an observation will
not be present in a given dataset, this does not necessarily mean that something went
wrong in the merge process, but this is another place

where one can often get
clues about what might have gone wrong in the merge process. Because in this example
all of the datasets include all of the cases, and because the merge went as it
should, the _merge* variables aren't very
interesting. We will discuss these variables in greater detail below, when we deal with
datasets where not all cases are present in all datasets.

Dropping unwanted variables

It is not uncommon to find that a large dataset contains many variables you
are not going to use in your analysis. You can just leave those variables in
your datasets when you merge them together, however, there are several reasons
you might not want to do this. First, there is a limit on the number of
variables Stata can handle. In Small Stata the limit is 99, in Stata/IC the
limit is 2,047 and in Stata/SE and Stata/MP the limit is 32,767. These limits
may see high, but if you merge multiple datasets, each

with a large number of
variables, you may exceed the limit for your type of Stata. The second reason
you might not want to leave unneeded variables in your dataset is
that each variable in memory uses additional system resources. A few extra
variables isn't going to hurt anything, but if you have a large number of
unwanted variables, you may be wasting system resources. Below we show several methods of eliminating extra variables. One option is that when you open
the datasets to sort them, you can also eliminate the variables you don't plan
to use. Depending on whether it is easier to list the variables you want you plan
to use in your analysis, or to list those variables you don't need, you can use
the commands keep or drop. There is at least one additional option,
you can open the datasets placing only those variables you need in memory. If I
have a dataset containing a number of variables, but the

only variables I need
from it are id and read, I can add variable names to my use
command as is shown in the first line of syntax below. This is particularly
useful with very large files which require a lot of memory to open. Once you
have opened the desired subset of variables, all you have to do is save the subset
of data under a new name.

```
use id read using http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data2
save d:datadata2_subset
```

In the above example, dataset2 contained the following variables: id,
read, write, math, science, and socst. Assume that my analysis only requires the variables read
and write, the only variables from dataset2 that are needed are those two and the variable id to merge the data with another dataset.
Below are examples of the same sort of data

preparation done above, using each of the techniques described. These techniques are equivalent, in that they produce the same end result. The efficiency of each technique varies depending on the situation.

Using keep to select variables:

```
use
http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data2, clear
keep id read write
sort id
save d:datadata2_b
```

Using drop to remove unwanted variables:

```
use
http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data2, clear
drop math science socst
sort id
save d:datadata2_b
```

Opening a subset of the data:

```
use id read write using
http://statistics.ats.ucla.edu/stat/data/stata_faq_multmer
ge/data2, clear
sort id
save d:datadata2_b
```

The _merge variables

The _merge variable(s) created by the merge command are easy to miss, but are very important. As discussed above, they tell us which dataset(s) each case came from. This is important because a lot of values that came from only one dataset may suggest a problem in the merge process. However, it is not uncommon for some cases to be in one dataset, but not another. In panel data this can occur when a given respondent did not participate in all the waves of the study. It can also occur for a number of other reasons. For example, a female respondent might appear in the subset of the data with demographic information, but be completely absent from the subset of data with information on female respondents'

children, because she does
not have children. Because cases that are not present in all datasets are not
necessarily a problem, in order for the information in _merge variables to be useful you need to know
what to expect if the datasets merged correctly. In the example above, where the same
200 cases appeared in three datasets I would expect to see 200 cases, all of which
came from all three of the datasets. If there are some cases missing from some of the
datasets, then I would expect to see a certain number of cases that did not come from
all the datasets, but I still need to make sure there aren't too many that come
from only some of the datasets. Having too many, or all, of the cases in your merged dataset come
from one, or only a few of the datasets you've merged is a sign that the id
variable does not match correctly across datasets. This is particularly common when
the id variable is a string. Below we examine a dataset after merging to see if all

went as expected.

The output below shows the file describe for a dataset data1m.dta, if we look
at the number of observations (obs) we see that the dataset contains only 197 cases,
but we know the study overall included 200 cases, so we know that there are three
cases missing entirely from data1m. This is important information if we are going to
correctly interpret the _merge variables later on. Finally we sort the data
and save it under a new name. To save space we won't show the output for the
other two datasets (the code does appear below in case you want
to run it). Assume that when we run describe on data2m and data3m we discover that they
are also missing cases. Dataset data2m contains 196 observations, and
dataset3m contains 197. It is possible that some of these cases are missing from
all three datasets (i.e. the missing observations overlap across datasets) but it is

also possible that all 200 observations occur in at least one of the datasets. We will find out once we merge the data.

```
use
http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data1m, clear
(highschool and beyond (200 cases))

describe
Contains data from data1m.dta
obs: 197 highschool and beyond (200 cases)
vars: 4 24 Jul 2008 16:31
size: 3,940 (99.6% of memory free)
-------------------------------------------------------------------------------
storage display value
variable name type format label variable label
-------------------------------------------------------------------------------
id float %9.0g
female float %9.0g fl
race float %12.0g rl
ses float %9.0g sl
```

Sorted by:

sort idsave d:datadata1m_a , replace

use
http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data2m, clear
describe
sort id
save d:datadata2m_a , replace

use
http://statistics.ats.ucla.edu/stat/data/stata_faq_multmerge/data3m, clear
describe
sort id
save d:datadata3m_a , replace

Once we have examined and sorted the datasets we can merge them. The syntax
below does this, note that the command is the same as in
the first example. By default, Stata will allow cases to

**come from
any of the three datasets. There are options that will allow you to control
which datasets the cases come from, you can find out about them by typing "help merge" (without
the quotes) in Stata.**

merge id using d:datadata1m_a d:datadata2m_a

**As before, the merge command created three new variables _merge,
_merge1, and _merge2. The variable _merge gives information about which cases
were present in the master dataset, it takes on one of three values:**

_merge = 1	The observation is present only in the master dataset
_merge = 2	The observation is present in one of the using datasets (but not the master datatset).
_merge = 3	The observation is present in at least two datasets, either master or using

**When more than one dataset appears in the using list, merge creates additional
_merge variables,
one for each dataset listed in using (e.g. _merge1, _merge2). These variables are equal to 1 if the**

observation was present in the dataset associated with that variable, and zero otherwise.

We will start our examination of these variables with _merge. Below we have
used tab to look at the variable _merge. The results show that we ended
up with a total of 200 observations, which is what we expected. Looking at the breakdown,
198 observations were present in at least two of the three datasets (_merge=3). This
includes both the cases that occur in all three datasets, and those that occur in only two
out of the three. There is one case that is only present in the master dataset, that is,
data3m (_merge=2). Finally there is one case that is only present in one of the using
datasets, that is, one case exists in data1m or data2m that does not exist in either
of the other two datasets. Because there are two of using variables, the information from
_merge do not give us complete information. This is where _merge1 and
_merge2 become useful.

tab _merge

_merge | Freq. Percent Cum.

------------+-----------------------------------

1 | 1 0.50 0.50

2 | 1 0.50 1.00

3 | 198 99.00 100.00

------------+-----------------------------------

Total | 200 100.00

We can use _merge1 and _merge2 to check our merge more closely.

When

there is more than one dataset in the using statement, there will be one of these variables for each

dataset in the using statement. The variables are assigned numbers based on the order of the datasets in the using statement. The variable label also indicates which dataset it is for. Here _merge1

indicates whether a case occurred in dataset data1m_a. _merge# is equal to 1 in the if the case in occurred in the corresponding dataset, and equal to 0 if it did not. Below we have used the

tab command to look at _merge1. Recall from above

that data1m_a had 197 cases, and
197 are equal to one for the variable _merge1. If the number of cases is correct (which we already know from
_merge) you should almost always see what you expect to see.

tab _merge1

_merge |
representin |
g data1m_a | Freq. Percent Cum.
------------+-----------------------------------
0 | 3 1.50 1.50
1 | 197 98.50 100.00
------------+-----------------------------------
Total | 200 100.00

Crosstabulating the _merge variables (again using the tab
command) is often a better check of what went of the merge process than oneway frequencies,
like the above output. Below we show _merge crossed with

_merge1. From this we can see that 196 cases are present in _merge1
and at least one of our other three datasets. Looking at the row for _merge=2
we see that one of the cases present in data1m_a was not present in the master dataset
(i.e. data3m_a). The row for _merge=1 shows that one case is present in the master
dataset and not present in data1m_a. Because _merge=1 indicates a case found only in the master dataset, this is logical. Although it can sometimes be difficult to figure
out exactly what they tables are telling you, crosstabulating the _merge variables
is probably the most effective way check that your data merged properly.

tab _merge _merge1

| _merge representing
| data1m_a
_merge | 0 1 | Total
-----------+----------------------+----------
1 | 1 0 | 1

2 | 0 1 | 1

3 | 2 196 | 198

----------+----------------------+----------

Total | 3 197 | 200

One final note on the _merge variables, they are temporary, that means they
will be discarded when you close Stata. If you wish to keep these variables you need to
rename them using the command rename, or by telling merge to create them as
permanent variables using the option _merge(varname) (this must be done at the time
you run merge).