SOC 561: Programing for the Social Sciences
Community Resource – Append, Merge, and Collapse
Steven Lizzol

A QUICK STATA GUIDE:
APPEND, MERGE, AND COLLAPSE

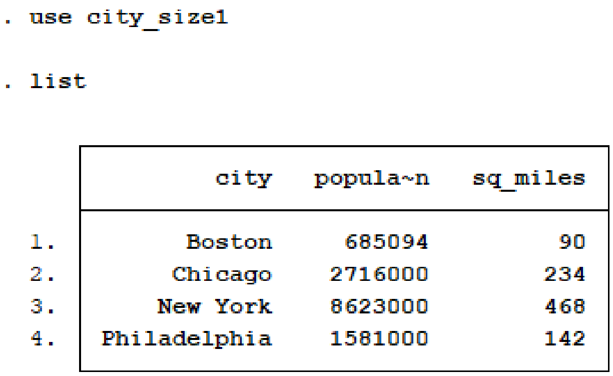
Before we begin, it is highly recommended that when performing append, merge, or collapse procedures the user performs the `preserve` / `restore` command beforehand to ensure their data are maintained. If you need a quick review, see the appendix.

1. APPEND

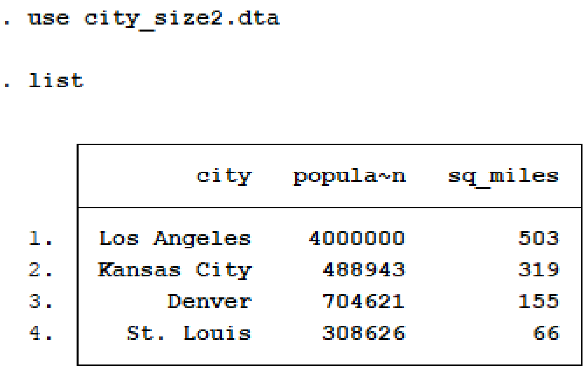
Adding cases / observations

Command:
```
append using dataset.dta
```

The append command combines the dataset in memory, known as the *master* dataset, with a dataset on disk, known as the *using* dataset. Typically, a user would implement the append command when they would like to add observations to an existing dataset with the same or similar variables. Let's assume we are interested in combining the following datasets:

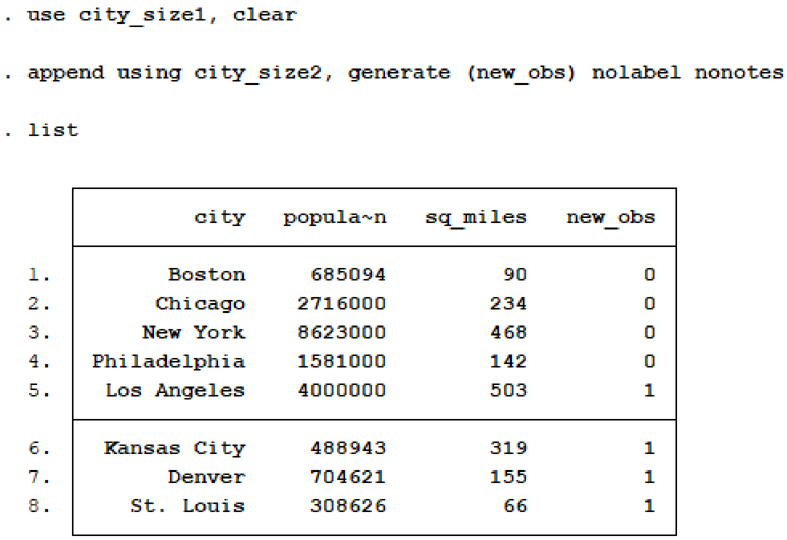


```
. use city_size1

. list

     +-------------------------------------+
     |        city   popula~n   sq_miles |
     |-------------------------------------|
  1. |      Boston     685094         90 |
  2. |     Chicago    2716000        234 |
  3. |    New York    8623000        468 |
  4. | Philadelphia   1581000        142 |
     +-------------------------------------+
```



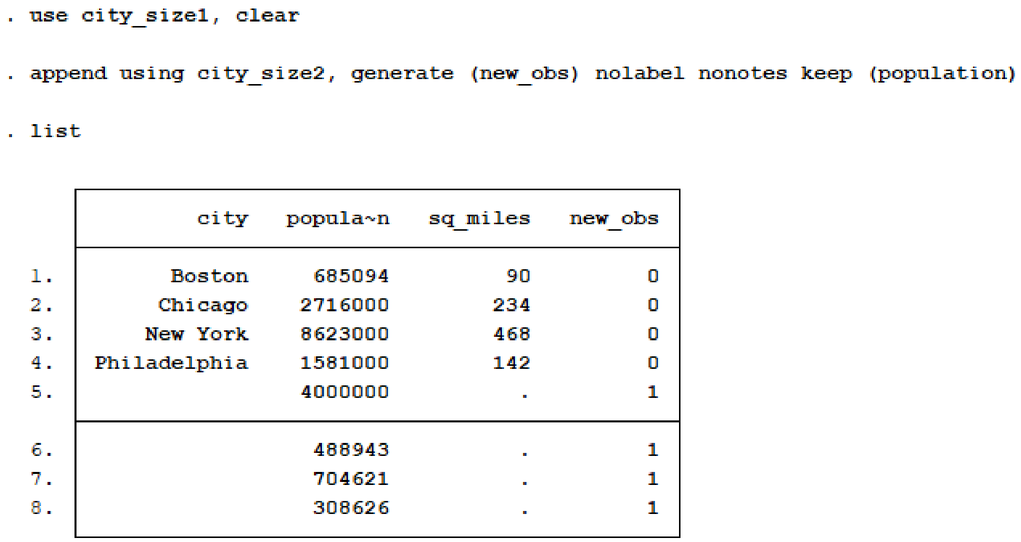
```
. use city_size2.dta

. list

     +-------------------------------------+
     |        city   popula~n   sq_miles |
     |-------------------------------------|
  1. | Los Angeles    4000000        503 |
  2. | Kansas City     488943        319 |
  3. |      Denver     704621        155 |
  4. |   St. Louis     308626         66 |
     +-------------------------------------+
```

Each dataset provides three variables: city, population, and square miles. All variables are named and formatted the same for each dataset. To append "city_size2.dta" to "city_size1.dta" we use the following:



```
. use city_size1, clear

. append using city_size2, generate (new_obs) nolabel nonotes

. list

     +-----------------------------------------------+
     |        city   popula~n   sq_miles   new_obs |
     |-----------------------------------------------|
  1. |      Boston     685094         90         0 |
  2. |     Chicago    2716000        234         0 |
  3. |    New York    8623000        468         0 |
  4. | Philadelphia   1581000        142         0 |
  5. | Los Angeles    4000000        503         1 |
     |-----------------------------------------------|
  6. | Kansas City     488943        319         1 |
  7. |      Denver     704621        155         1 |
  8. |   St. Louis     308626         66         1 |
     +-----------------------------------------------+
```

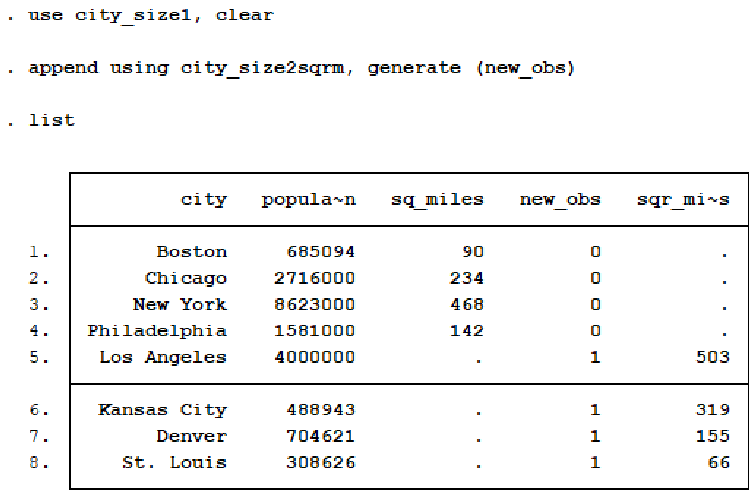
The command produced a new dataset that combined the observations from the master dataset, (city_size1.dta) with the using dataset (city_size2.dta). Notice no new variables with the exception of "new_obs" were created in the process; only additional observations. The additional commands after the coma are optional and are not required to execute the command. The "`, generate (new_obs)`" option above provided a "new_obs" variable that identifies which observations were appended to the master dataset. Additionally, the options "`nolabel nonotes`" prevented any labels or notes from the using dataset copying over to the appended master dataset. A different option allows the user to omit variables from the using dataset. For example:



```
. use city_size1, clear

. append using city_size2, generate (new_obs) nolabel nonotes keep (population)

. list

     +----------------------------------------------+
     |         city   popula~n   sq_miles   new_obs |
     |----------------------------------------------|
  1. |       Boston     685094         90         0 |
  2. |      Chicago    2716000        234         0 |
  3. |     New York    8623000        468         0 |
  4. | Philadelphia    1581000        142         0 |
  5. |                 4000000          .         1 |
     |----------------------------------------------|
  6. |                  488943          .         1 |
  7. |                  704621          .         1 |
  8. |                  308626          .         1 |
     +----------------------------------------------+
```

The "`keep (population)`" identified only one variable to copy over while leaving the other fields ("city" and "sq_miles for this example) as missing.

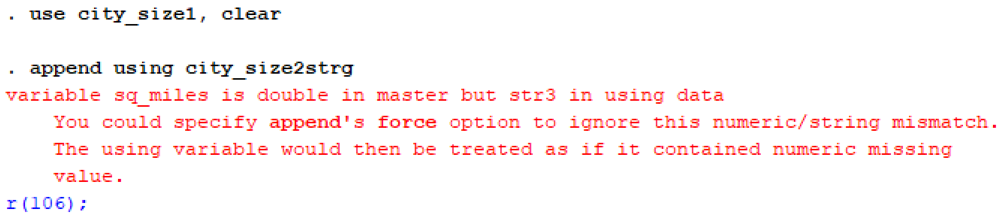
If you attempt to append a using dataset with variables that do not match with the master dataset, they will be added to the appended dataset as additional variables. For example:



```
. use city_size1, clear

. append using city_size2sqrm, generate (new_obs)

. list

     +---------------------------------------------------------+
     |         city   popula~n   sq_miles   new_obs   sqr_mi~s |
     |---------------------------------------------------------|
  1. |       Boston     685094         90         0          . |
  2. |      Chicago    2716000        234         0          . |
  3. |     New York    8623000        468         0          . |
  4. | Philadelphia    1581000        142         0          . |
  5. |  Los Angeles    4000000          .         1        503 |
     |---------------------------------------------------------|
  6. |  Kansas City     488943          .         1        319 |
  7. |       Denver     704621          .         1        155 |
  8. |    St. Louis     308626          .         1         66 |
     +---------------------------------------------------------+
```

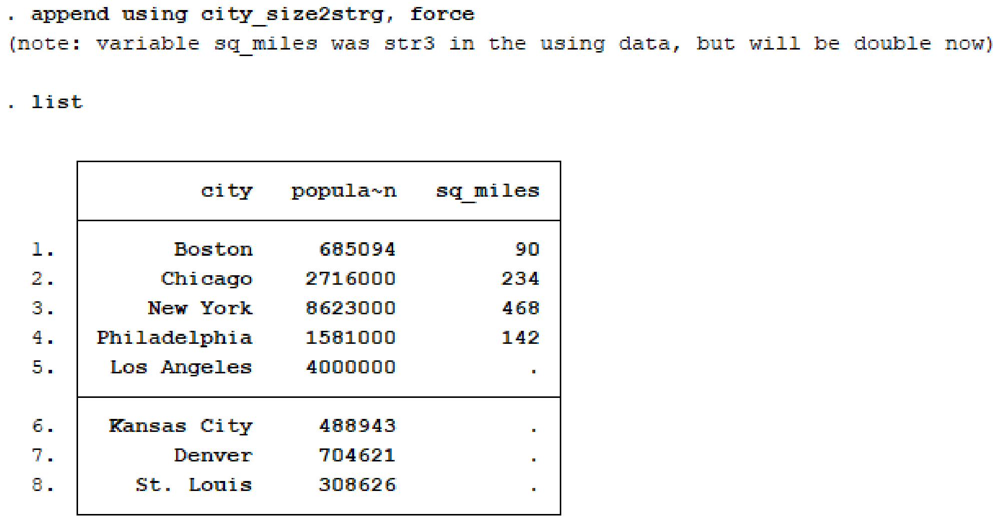
Even though the variables "sq_miles" and "sqr_miles" provide the same measure, they remained separate and provided missing data in observations where the variable name did not match.

Another consideration is if you are attempting to append a using dataset with a variable by the same name as the master dataset, but in a different format. For example, if you attempt to append the using dataset with a string variable (sq_miles) to a master dataset with a numeric variable by the same name (sq_miles), you will receive the following error message:



```
. use city_size1, clear

. append using city_size2strg
variable sq_miles is double in master but str3 in using data
    You could specify append's force option to ignore this numeric/string mismatch.
    The using variable would then be treated as if it contained numeric missing
    value.
r(106);
```

If we use the "`force`" option in this situation, Stata will append the using dataset and inform you that the variable from the using dataset will assume the format of the master dataset. Additionally, the values form the using datasets will change to missing.



```
. append using city_size2strg, force
(note: variable sq_miles was str3 in the using data, but will be double now)

. list

     +------------------------------------+
     |         city   popula~n   sq_miles |
     |------------------------------------|
  1. |       Boston     685094         90 |
  2. |      Chicago    2716000        234 |
  3. |     New York    8623000        468 |
  4. | Philadelphia    1581000        142 |
  5. |  Los Angeles    4000000          . |
     |------------------------------------|
  6. |  Kansas City     488943          . |
  7. |       Denver     704621          . |
  8. |    St. Louis     308626          . |
     +------------------------------------+
```

Major Options:

`generate(newvar)`	newvar marks source of resulting observations
`keep(varlist)`	keep specified variables from appending dataset(s)
`nolabel`	do not copy value-label definitions from dataset(s) on disk
`nonotes`	do not copy notes from dataset(s) on disk
`force`	append string to numeric or numeric to string without error

For additional information and examples, you can view the online `append` manual here.

2. MERGE

Adding variables

Command:

```
merge using dataset.dta
```

The merge command combines the dataset in memory, known as the *master* dataset, with a dataset on disk, known as the *using* dataset. While `append` added observations to a master dataset, the general purpose of `merge` is to add variables to existing observations. In its simplest form from past Stata versions (the command above), datasets are merged based on their observation (or row) order (e.g., the first observation is paired with the first outcomes for each variable). This older syntax is not recommended as it can be potentially dangerous if the two datasets are sorted differently or possess more or less id variables (see merge warning in the appendix). You can produce the same results using the following command recognized by newer Stata versions:

```
merge 1:1 _n using filename
```

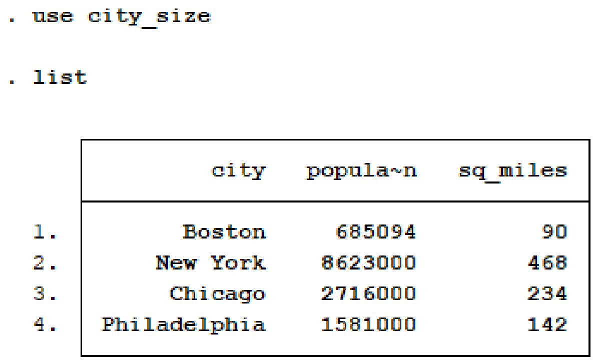
However, the new Stata commands in versions 10 or later mitigate chances for mismatched variables and observations. We will now focus on the two primary types: 'one-to-one' and a 'one-to-many' (or 'many-to-one').

One-to-one merging:

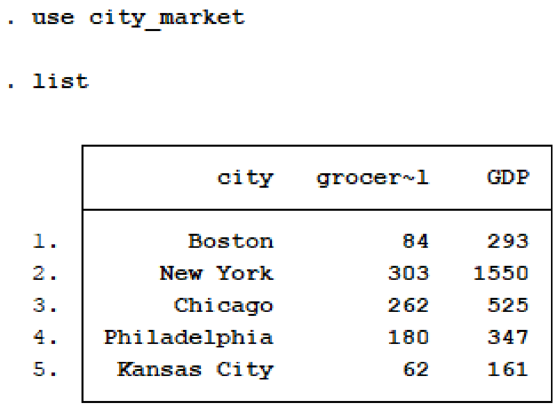
Command:

```
merge 1:1 varlist using filename
```

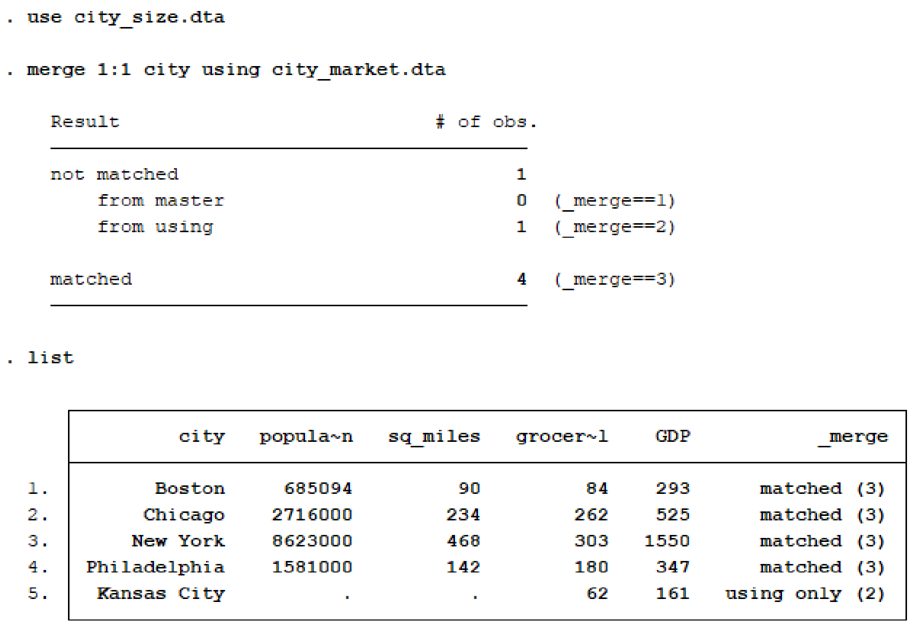
For this command "1:1" specifies that there is one id variable in each dataset that needs to be merged. For example, imagine you had a master dataset, "city_size.dta" that possessed the id variable "city" with multiple size variables (e.g., population, total square miles), and a separate using dataset, "city_market.dta" that possessed the same id variables with corresponding market variables (e.g., number of grocery retailers and total GDP). Given that you wanted to merge these two datasets and the presence of one identifier, "city" in this case, you would perform a one-to-one merge.



```
. use city_size

. list

     +----------------------------------+
     |         city   popula~n   sq_miles |
     |----------------------------------|
  1. |       Boston     685094         90 |
  2. |     New York    8623000        468 |
  3. |      Chicago    2716000        234 |
  4. | Philadelphia    1581000        142 |
     +----------------------------------+
```



```
. use city_market

. list

     +-----------------------------------+
     |         city   grocer~l      GDP |
     |-----------------------------------|
  1. |       Boston         84      293 |
  2. |     New York        303     1550 |
  3. |      Chicago        262      525 |
  4. | Philadelphia        180      347 |
  5. | Kansas City          62      161 |
     +-----------------------------------+
```



```
. use city_size.dta

. merge 1:1 city using city_market.dta

    Result                           # of obs.
    -----------------------------------------
    not matched                             1
        from master                         0  (_merge==1)
        from using                          1  (_merge==2)

    matched                                 4  (_merge==3)
    -----------------------------------------

. list

     +--------------------------------------------------------------------------+
     |         city   popula~n   sq_miles   grocer~l    GDP            _merge |
     |--------------------------------------------------------------------------|
  1. |       Boston     685094         90         84    293       matched (3) |
  2. |      Chicago    2716000        234        262    525       matched (3) |
  3. |     New York    8623000        468        303   1550       matched (3) |
  4. | Philadelphia    1581000        142        180    347       matched (3) |
  5. |  Kansas City          .          .         62    161    using only (2) |
     +--------------------------------------------------------------------------+
```

Stata merged the using dataset, “city_market.dta” variables to the corresponding observations in the “city” variable within the master dataset, “city_size.dta.” Also, the merge occurs based on the id variable regardless of sort order. The command will also create an additional variable “merge” that identifies if an observation was matched in the merge. It provides three indicators:

1 = observation found only in the master dataset
2 = observation found only in the using dataset
3 = observation found in both master and using dataset (complete match)

The “Kansas City” observation received a “2” identifier because this observation was only provided in the using dataset. Additionally, it resulted in missing values for the variables in the master dataset. The merge can occur based on other id variables if desired.

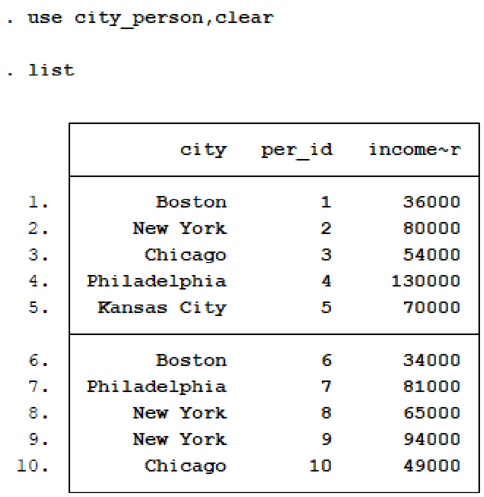
Many-to-one & one-to-many merge:

Command:

`merge 1:m` *`varlist`* `using` *`filename`*

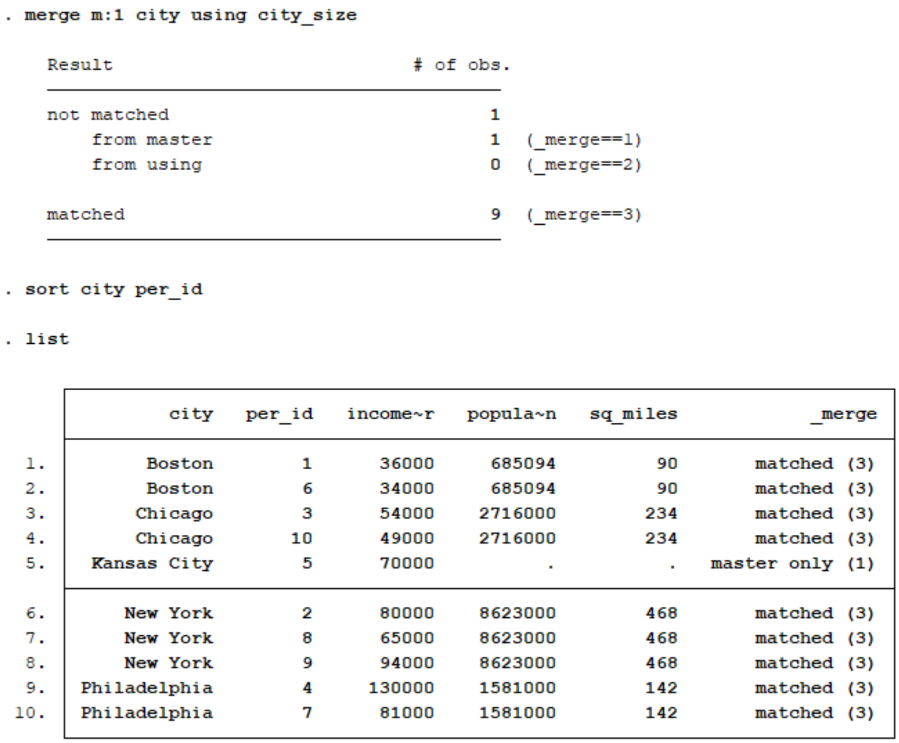
`merge m:1` *`varlist`* `using` *`filename`*

You can also merge datasets that have similar id variables with observations at different levels of analysis. For example, let’s suppose in addition to the “city_size.dta” and “city_market.dta” files, you have a “city_person.dta” dataset with variables that capture a person’s city of residence and yearly income.



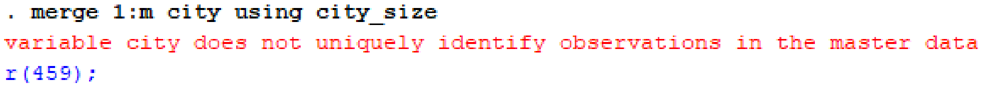
```
. use city_person,clear

. list

     +-------------------------------+
     |         city   per_id   income~r |
     |-------------------------------|
  1. |       Boston        1      36000 |
  2. |     New York        2      80000 |
  3. |      Chicago        3      54000 |
  4. | Philadelphia        4     130000 |
  5. |  Kansas City        5      70000 |
     |-------------------------------|
  6. |       Boston        6      34000 |
  7. | Philadelphia        7      81000 |
  8. |     New York        8      65000 |
  9. |     New York        9      94000 |
 10. |      Chicago       10      49000 |
     +-------------------------------+
```

This dataset includes a personal id variable. Using “city” as the id variable, you can merge “city_size.dta” as the using dataset with “city_person.dta” as the master dataset. Performing a ***many-to-one merge*** produces the following output:



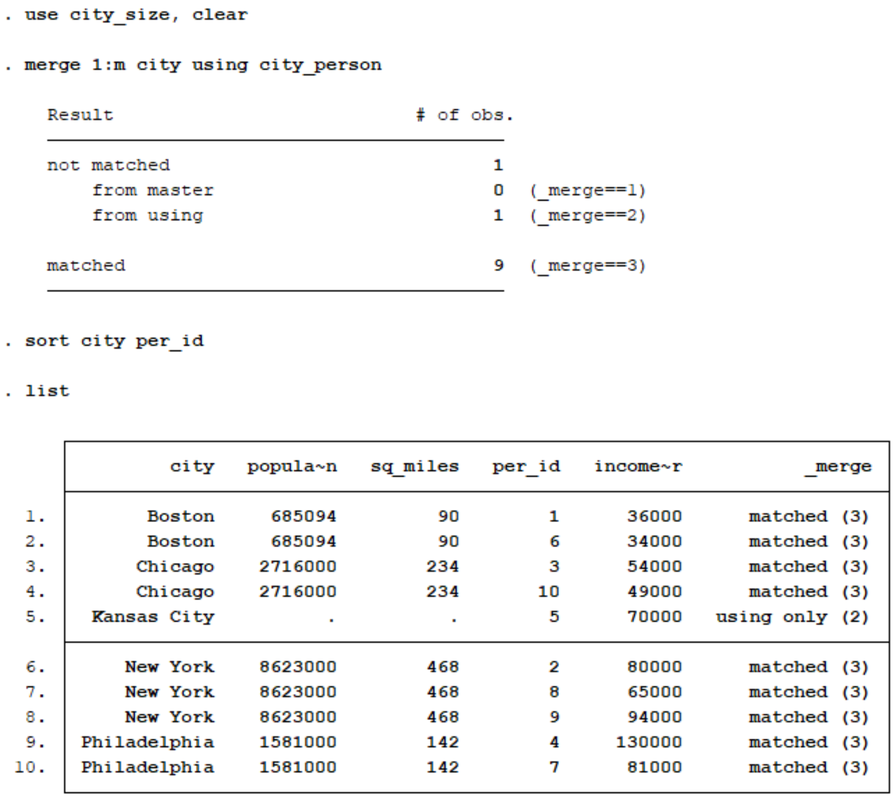
```
. merge m:1 city using city_size

    Result                           # of obs.
    -----------------------------------------
    not matched                             1
        from master                         1  (_merge==1)
        from using                          0  (_merge==2)

    matched                                 9  (_merge==3)
    -----------------------------------------

. sort city per_id

. list

     +-----------------------------------------------------------------------+
     |         city   per_id   income~r   popula~n   sq_miles          _merge |
     |-----------------------------------------------------------------------|
  1. |       Boston        1      36000     685094         90     matched (3) |
  2. |       Boston        6      34000     685094         90     matched (3) |
  3. |      Chicago        3      54000    2716000        234     matched (3) |
  4. |      Chicago       10      49000    2716000        234     matched (3) |
  5. |  Kansas City        5      70000          .          . master only (1) |
     |-----------------------------------------------------------------------|
  6. |     New York        2      80000    8623000        468     matched (3) |
  7. |     New York        8      65000    8623000        468     matched (3) |
  8. |     New York        9      94000    8623000        468     matched (3) |
  9. | Philadelphia        4     130000    1581000        142     matched (3) |
 10. | Philadelphia        7      81000    1581000        142     matched (3) |
     +-----------------------------------------------------------------------+
```

Now, all city level measures are assigned to each person depending on which city they reside in. Notice that values for "population" and "sq_miles" are missing since "Kansas City" is an identifier only provided in the "city_person.dta" dataset, thus the `_merge==1` result. If we attempt a one-to-many command with the same using and master dataset arrangement, Stata will present an error:



```
. merge 1:m city using city_size
variable city does not uniquely identify observations in the master data
r(459);
```

The "city_person.dta" dataset fails to provide an id variable that Stata can recognize as a unique identifier (e.g., "Boston" is assigned to more than one observation). If we switch the datasets where "city_size.dta" is the master dataset and "city_person.dta" is the using dataset, a ***one-to-many merge*** is possible.



```
. use city_size, clear

. merge 1:m city using city_person

    Result                           # of obs.
    -----------------------------------------
    not matched                             1
        from master                         0  (_merge==1)
        from using                          1  (_merge==2)

    matched                                 9  (_merge==3)
    -----------------------------------------

. sort city per_id

. list

     +-------------------------------------------------------------------+
     |         city   popula~n   sq_miles   per_id   income~r      _merge |
     |-------------------------------------------------------------------|
  1. |       Boston     685094         90        1      36000   matched (3) |
  2. |       Boston     685094         90        6      34000   matched (3) |
  3. |      Chicago    2716000        234        3      54000   matched (3) |
  4. |      Chicago    2716000        234       10      49000   matched (3) |
  5. |  Kansas City          .          .        5      70000 using only (2) |
     |-------------------------------------------------------------------|
  6. |     New York    8623000        468        2      80000   matched (3) |
  7. |     New York    8623000        468        8      65000   matched (3) |
  8. |     New York    8623000        468        9      94000   matched (3) |
  9. | Philadelphia    1581000        142        4     130000   matched (3) |
 10. | Philadelphia    1581000        142        7      81000   matched (3) |
     +-------------------------------------------------------------------+
```

Notice the one-to-many merged dataset sorted on “city” and “per_id” produces the same output as the many-to-one merged dataset.

A ***many-to-many merge*** can occur when you are unaware of how many of the same identifiers exist between two datasets, but believe there is at least one pair. A many-to-many command is not recommended. As stated in the Stata Data Management Reference Manual (Release 15):

> Because m:m merges are such a bad idea, we are not going to show you an example. If you think that you need an m:m merge, then you probably need to work with your data so that you can use a 1:m or m:1 merge.

It is recommended that the user is familiar enough with the datasets they desire to merge that a many-to-one or one-to-many is used for the desired outcomes.

Major options:

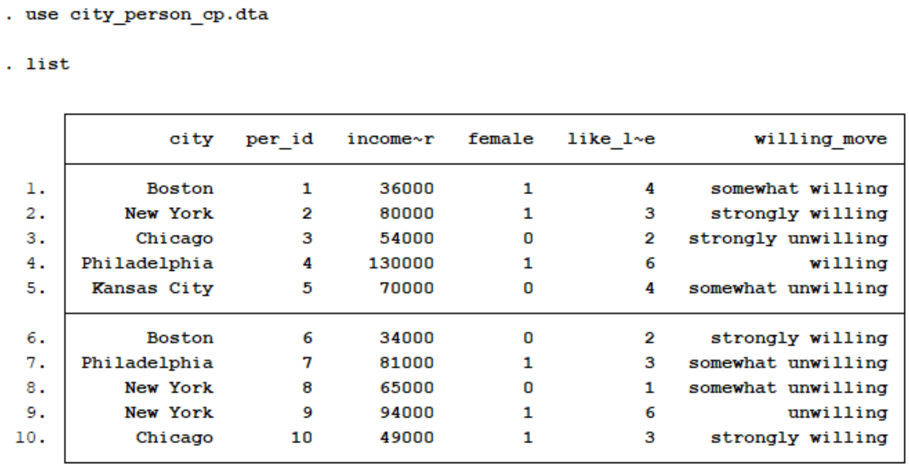
`keepusing (varlist)`	allows you to merge only select variables from the using dataset.
`generate (newvar)`	changes “_merge” variable name to one of your choosing
`nogenerate`	“_merge” variable not created after a merge
`nolabel`	prevents value/label definitions copying over from the using dataset
`nonotes`	prevents notes copying over form the using dataset
`noreport`	prevents the match results from showing after the merge

For additional information and examples, you can view the online `merge` manual here.

3. Collapse

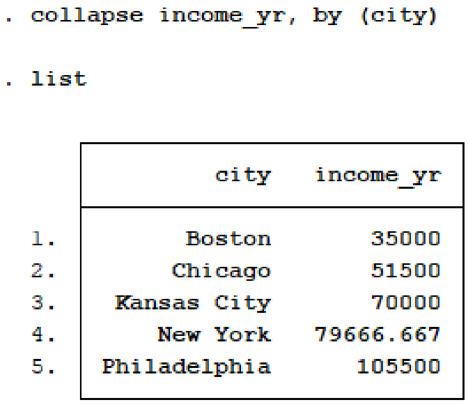
Command:
`collapse (`*`statistic`*`) var1, by (var2)`

This command takes an open (or master) dataset and creates a new dataset by summarizing statistics on a selected variable. Let's use the "city_person_cp.dta" dataset to go through some examples. For these examples we add three new variables: "female" (1 = female, 0 = male), "like_live" (rating of how much a person likes the city they live in (1 = completely dislike to 6 = completely like), and "willing_move" (rating of how willing a person is to move to a different city (1 = strongly unwilling to 6 = strongly willing).



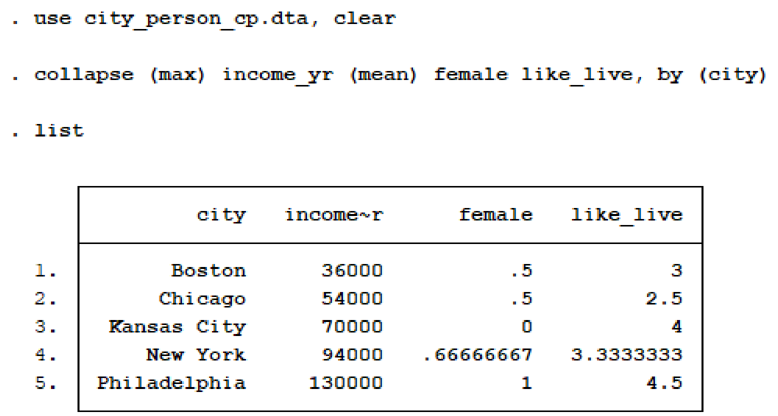
```
. use city_person_cp.dta

. list

     +--------------------------------------------------------------------------+
     |         city   per_id   income~r   female   like_l~e        willing_move |
     |--------------------------------------------------------------------------|
  1. |       Boston        1      36000        1          4     somewhat willing |
  2. |     New York        2      80000        1          3     strongly willing |
  3. |      Chicago        3      54000        0          2   strongly unwilling |
  4. | Philadelphia        4     130000        1          6              willing |
  5. |  Kansas City        5      70000        0          4   somewhat unwilling |
     |--------------------------------------------------------------------------|
  6. |       Boston        6      34000        0          2     strongly willing |
  7. | Philadelphia        7      81000        1          3   somewhat unwilling |
  8. |     New York        8      65000        0          1   somewhat unwilling |
  9. |     New York        9      94000        1          6            unwilling |
 10. |      Chicago       10      49000        1          3     strongly willing |
     +--------------------------------------------------------------------------+
```

The `collapse` command will allow us to find a statistic by a specific variable. For example, if we wanted to find the mean yearly income for each city based on the individual dataset, we would use the following:



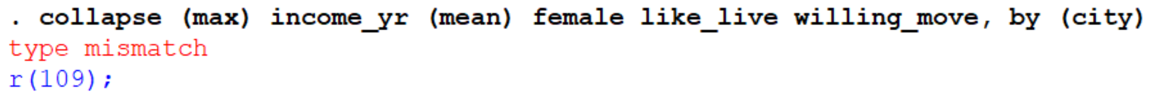
```
. collapse income_yr, by (city)

. list

     +--------------------------+
     |         city   income_yr |
     |--------------------------|
  1. |       Boston       35000 |
  2. |      Chicago       51500 |
  3. |  Kansas City       70000 |
  4. |     New York   79666.667 |
  5. | Philadelphia      105500 |
     +--------------------------+
```

The command 'collapsed' all individual yearly incomes in the dataset and produced a new dataset presenting the mean for each city. By default, `collapse` will provide the mean for each numeric variable listed. The collapse output can be changed to a variety of statistics. For example:



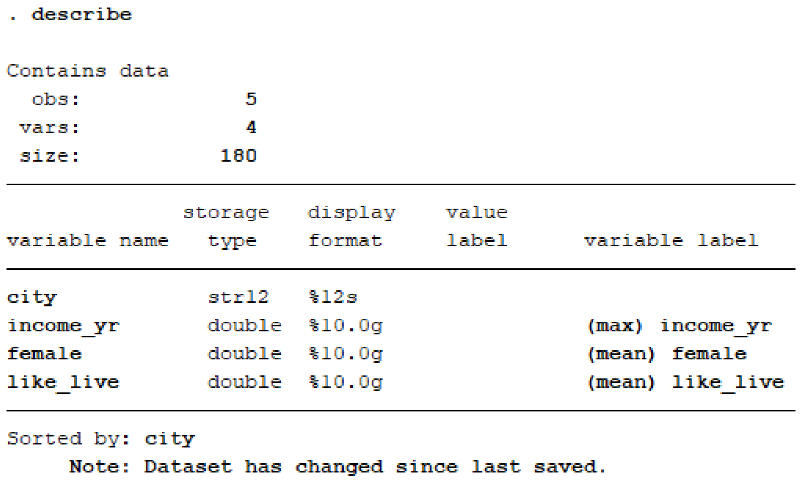
```
. use city_person_cp.dta, clear

. collapse (max) income_yr (mean) female like_live, by (city)

. list

     +---------------------------------------------------+
     |         city   income~r      female   like_live   |
     |---------------------------------------------------|
  1. |       Boston      36000          .5           3   |
  2. |      Chicago      54000          .5         2.5   |
  3. |  Kansas City      70000           0           4   |
  4. |     New York      94000   .66666667   3.3333333   |
  5. | Philadelphia     130000           1         4.5   |
     +---------------------------------------------------+
```

Here we ask Stata to find the maximum yearly income and the mean for female and like_live for each city. The output shows the highest yearly income for those in the sample who live in New York being $94,000 with roughly 67% of the New York respondents identifying as female and a mean score of how much they like living in New York being 3.33 (somewhat dislike). If we were to include the "willing_move" variable, Stata posts an error message identifying a "type mismatch" and require you to change this variable from a string to numeric format to perform the collapse.



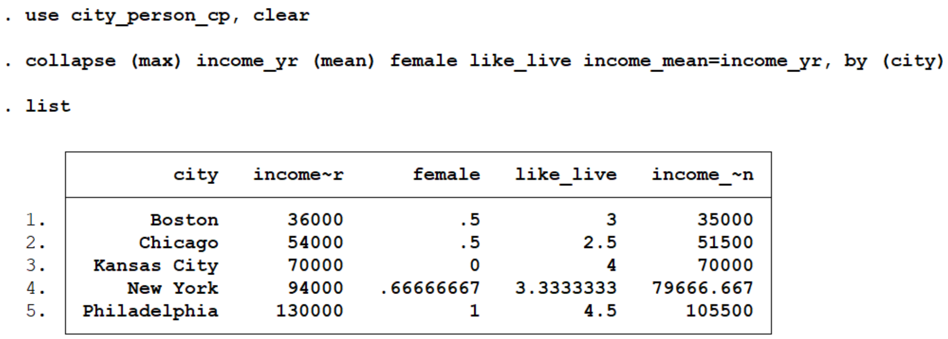
```
. collapse (max) income_yr (mean) female like_live willing_move, by (city)
type mismatch
r(109);
```

Using multiple statistical outcomes from one collapse can make keeping track of statistic output somewhat difficult by looking at the variable name alone. Fortunately, Stata develops labels for each variable providing details on which statistic occurred from a `collapse`.

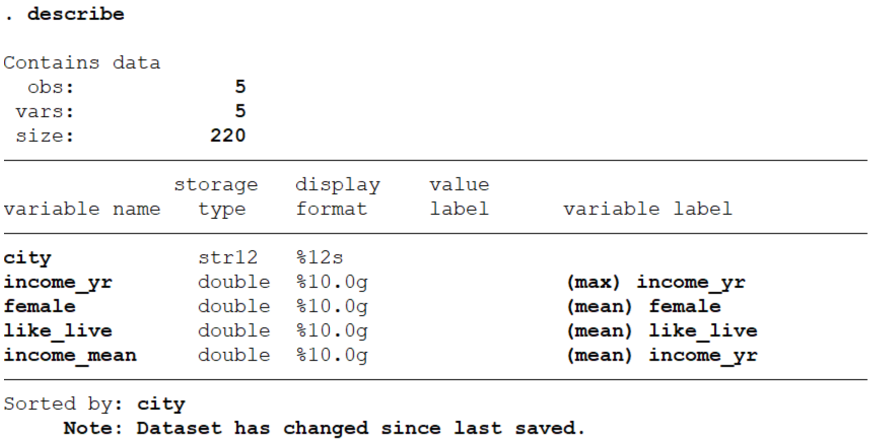


```
. describe

Contains data
  obs:             5
 vars:             4
 size:           180
--------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
--------------------------------------------------------------------
city            str12   %12s
income_yr       double  %10.0g                (max) income_yr
female          double  %10.0g                (mean) female
like_live       double  %10.0g                (mean) like_live
--------------------------------------------------------------------
Sorted by: city
     Note: Dataset has changed since last saved.
```

If you wanted to perform more than one statistic for the same variable, you will need to tell Stata the new variable name. For example, if we wanted the max and the mean for yearly income, we would perform the following command:

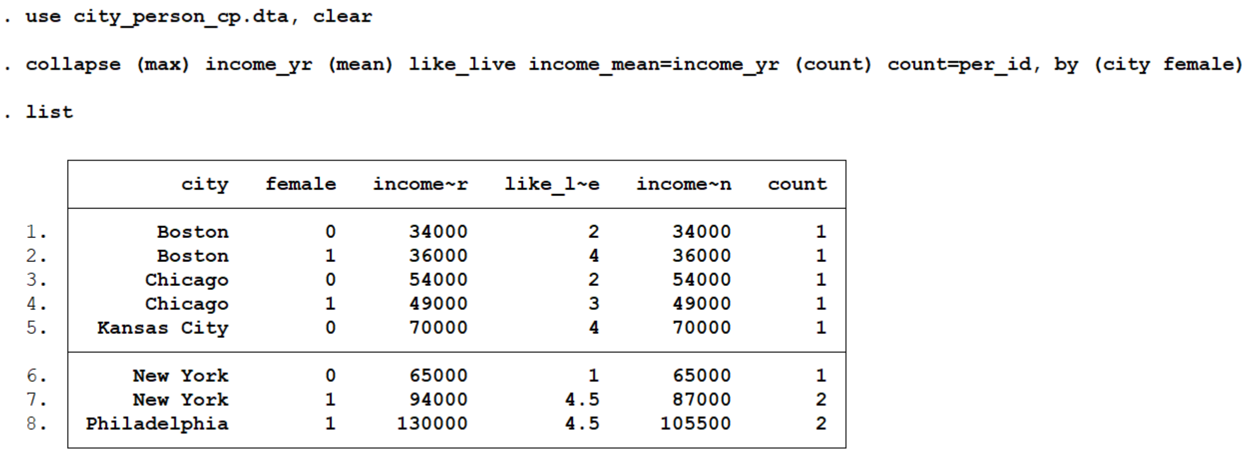


```
. use city_person_cp, clear

. collapse (max) income_yr (mean) female like_live income_mean=income_yr, by (city)

. list

     +--------------------------------------------------------------+
     |         city   income~r      female   like_live    income_~n |
     |--------------------------------------------------------------|
  1. |       Boston      36000          .5           3        35000 |
  2. |      Chicago      54000          .5         2.5        51500 |
  3. | Kansas City       70000           0           4        70000 |
  4. |     New York      94000   .66666667   3.3333333    79666.667 |
  5. | Philadelphia     130000           1         4.5       105500 |
     +--------------------------------------------------------------+
```



```
. describe

Contains data
  obs:             5
 vars:             5
 size:           220
------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------
city            str12   %12s
income_yr       double  %10.0g                (max) income_yr
female          double  %10.0g                (mean) female
like_live       double  %10.0g                (mean) like_live
income_mean     double  %10.0g                (mean) income_yr
------------------------------------------------------------------------
Sorted by: city
     Note: Dataset has changed since last saved.
```

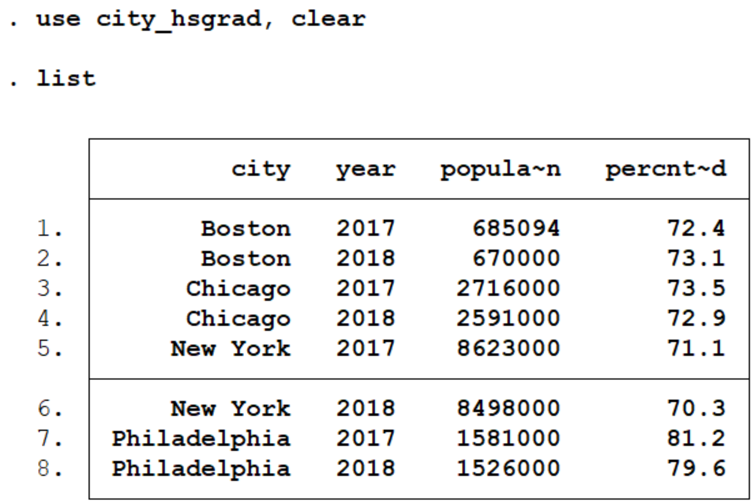
Stata will also allow a collapse to condition on combinations of variables. For example, we could perform the same collapse as the previous command, however instead of collapsing on just the city, we could collapse on the city and whether or not you identify as a female. For this next example, we will ask Stata to provide a count for the number of observations that occur for the specified collapse categories. For this to work, you must use a variable that has no missing values; “per_id” is a good fit for this case.



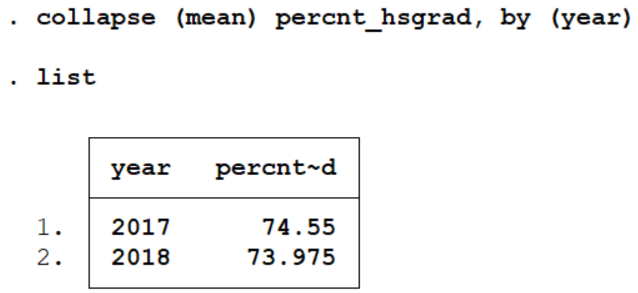
```
. use city_person_cp.dta, clear

. collapse (max) income_yr (mean) like_live income_mean=income_yr (count) count=per_id, by (city female)

. list

     +-----------------------------------------------------------------+
     |         city   female   income~r   like_l~e   income~n   count |
     |-----------------------------------------------------------------|
  1. |       Boston        0      34000          2      34000       1 |
  2. |       Boston        1      36000          4      36000       1 |
  3. |      Chicago        0      54000          2      54000       1 |
  4. |      Chicago        1      49000          3      49000       1 |
  5. |  Kansas City        0      70000          4      70000       1 |
     |-----------------------------------------------------------------|
  6. |     New York        0      65000          1      65000       1 |
  7. |     New York        1      94000        4.5      87000       2 |
  8. | Philadelphia        1     130000        4.5     105500       2 |
     +-----------------------------------------------------------------+
```

The output shows that the statistics based on the city and stratified on whether you identify as a female or not. For example, the max yearly income for those who live in New York and identify as female is $94,000, with the mean yearly income being $87,000 and an average “like_live” rating of 4.5 (like). This can be compared to those who live in New York who do not identify as a female (1 person in this simple dataset) with a yearly income of $65,000 and a “like_live” rating of 1 (strongly dislike). Additionally, Philadelphia does not have a “0” category because the dataset only captured two female observations.

Let’s now use a fictitious longitudinal dataset “city_hsgrad.dta” with variables that identify the city, city population, and percentage of high school graduates for the years 2017 and 2018 respectively. If we were interested in capturing the mean percentage of high school grads for the two different years, we could perform the following commands:

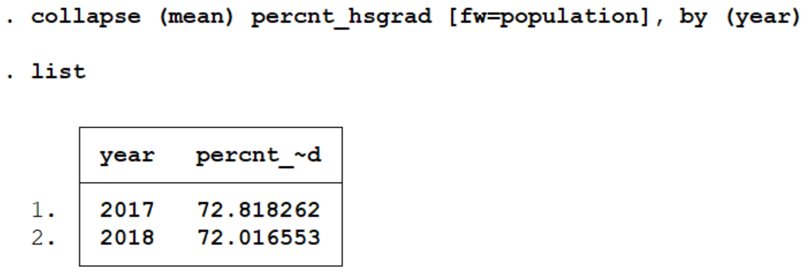


```
. use city_hsgrad, clear

. list

     +--------------------------------------------------+
     |         city   year   popula~n   percnt~d |
     |--------------------------------------------------|
  1. |       Boston   2017     685094       72.4 |
  2. |       Boston   2018     670000       73.1 |
  3. |      Chicago   2017    2716000       73.5 |
  4. |      Chicago   2018    2591000       72.9 |
  5. |     New York   2017    8623000       71.1 |
     |--------------------------------------------------|
  6. |     New York   2018    8498000       70.3 |
  7. | Philadelphia   2017    1581000       81.2 |
  8. | Philadelphia   2018    1526000       79.6 |
     +--------------------------------------------------+
```



```
. collapse (mean) percnt_hsgrad, by (year)

. list

     +-----------------+
     | year   percnt~d |
     |-----------------|
  1. | 2017      74.55 |
  2. | 2018     73.975 |
     +-----------------+
```

However, our results assume equal weighting of the graduate percentages regardless of city population and provide inaccurate outputs. A more precise approach involves using a weight command that accounts for each percentage on their respective population size.



```
. collapse (mean) percnt_hsgrad [fw=population], by (year)

. list

     +-------------------+
     | year    percnt_~d |
     |-------------------|
  1. | 2017    72.818262 |
  2. | 2018    72.016553 |
     +-------------------+
```

Major Options:

`by(varlist)`	groups over which stat is to be calculated
`cw`	casewise deletion instead of all possible observations

For additional information and examples, you can view the online `collapse` manual here.

Appendix

A quick word on preserve / restore:

If you are writing a Stata program that temporarily changes the dataset by the following append, merge, or collapse commands, it is highly recommended that you preserve your original dataset by using the `preserve` command. After preserve is entered, you can experiment with different dataset manipulation commands without worry of permanently changing your original dataset. Simply use the command `restore` to go back to the preserved dataset.

```
preserve

[commands that alter the dataset]

restore
```

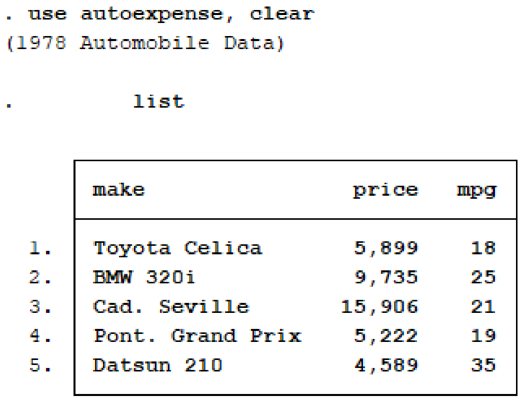
Additionally, if you are writing a Stata program that temporarily changes the order of the data and you want the data to be sorted in its original order at the end of execution, you can save a bit of programming by including `sortpreserve` on your program statement.

```
Program statdatasetfile, sortpreserve
```

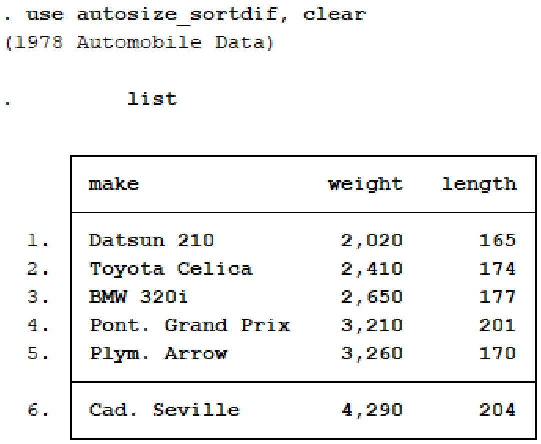
Stata will automatically sort into its original order at the end of execution.

Merge Warning

Suppose we wanted to merge the following two files: autoexpense.dta and autosize_sortdif.dta.

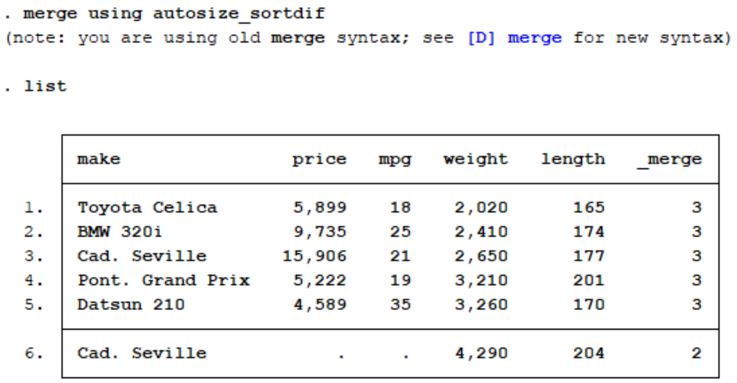


```
. use autoexpense, clear
(1978 Automobile Data)

.        list

     +----------------------------------+
     | make                price    mpg |
     |----------------------------------|
  1. | Toyota Celica       5,899     18 |
  2. | BMW 320i            9,735     25 |
  3. | Cad. Seville       15,906     21 |
  4. | Pont. Grand Prix    5,222     19 |
  5. | Datsun 210          4,589     35 |
     +----------------------------------+
```



```
. use autosize_sortdif, clear
(1978 Automobile Data)

.        list

     +-------------------------------------+
     | make               weight   length  |
     |-------------------------------------|
  1. | Datsun 210          2,020      165  |
  2. | Toyota Celica       2,410      174  |
  3. | BMW 320i            2,650      177  |
  4. | Pont. Grand Prix    3,210      201  |
  5. | Plym. Arrow         3,260      170  |
     |-------------------------------------|
  6. | Cad. Seville        4,290      204  |
     +-------------------------------------+
```

Notice how the autosize_sortdif.dta file has the same car makes with the exception of the "Cadillac Seville," but they are in a different order. If a simple merge command is used, Stata will do so based on the order within each dataset (e.g., the first observation is paired with the first outcomes for each variable). Here's the result:



```
. merge using autosize_sortdif
(note: you are using old merge syntax; see [D] merge for new syntax)

. list

     +------------------------------------------------------------------+
     | make                price   mpg   weight   length   _merge      |
     |------------------------------------------------------------------|
  1. | Toyota Celica       5,899    18    2,020      165        3      |
  2. | BMW 320i            9,735    25    2,410      174        3      |
  3. | Cad. Seville       15,906    21    2,650      177        3      |
  4. | Pont. Grand Prix    5,222    19    3,210      201        3      |
  5. | Datsun 210          4,589    35    3,260      170        3      |
     |------------------------------------------------------------------|
  6. | Cad. Seville            .     .    4,290      204        2      |
     +------------------------------------------------------------------+
```

The master dataset is now merged with the using dataset with a new "_merge" variable that indicates observations found in both master and using dataset (3) with the exception of the "Cadillac Seville" where observations were found only from the using dataset. There are two assumptions Stata was operating on: (1) the first variable was the id variable, and (2) the id variables were sorted in the same order. In this case Stata did not account for the different ordering of the "make" variable, and the datasets were merged incorrectly (e.g., the weight and length outcomes for the "Datsum 210" were assigned to the "Toyota Celica" in the merged dataset).