# Title

stata.com

**mi merge —** Merge mi data


Description Menu Syntax Options
Remarks and examples Stored results Also see


## Description

`mi merge` is `merge` for `mi` data; see [D] **merge** for a description of merging datasets.

It is recommended that the match variables (*varlist* in the syntax diagram) not include imputed or passive variables, or any varying or super-varying variables. If they do, the values of the match variables in $m = 0$ will be used to control the merge even in $m = 1$, $m = 2$, ..., $m = M$. Thus $m = 0$, $m = 1$, ..., $m = M$ will all be merged identically, and there will continue to be a one-to-one correspondence between the observations in $m = 0$ with the observations in each of $m > 0$.

## Menu

Statistics > Multiple imputation

## Syntax

```
mi merge 1:1 varlist using filename [ , options ]

mi merge m:1 varlist using filename [ , options ]

mi merge 1:m varlist using filename [ , options ]

mi merge m:m varlist using filename [ , options ]
```

| *options* | Description |
|---|---|
| Options | |
| `generate(`*newvar*`)` | create *newvar* recording how observations matched |
| `nolabel` | do not copy value-label definitions from using |
| `nonotes` | do not copy notes from using |
| `noreport` | do not display result summary table |
| `force` | allow string/numeric variable type mismatch without error |
| Results | |
| `assert(`*results*`)` | require observations to match as specified |
| `keep(`*results*`)` | results to keep |
| `noupdate` | see [MI] **noupdate option** |

`collect` is allowed; see [U] **11.1.10 Prefix commands**.

Notes:

1. Jargon:
   match variables = *varlist*, variables on which match performed
   master = data in memory
   using = data on disk (*filename*)
2. Master must be `mi set`; using may be `mi set`.
3. `mi merge` is syntactically and logically equivalent to `merge` (see [D] **merge**).
4. `mi merge` syntactically differs from `merge` in that the `nogenerate`, `sorted`, `keepusing()`, `update`, and `replace` options are not allowed. Also, no `_merge` variable is created unless the `generate()` option is specified.
5. *filename* must be enclosed in double quotes if *filename* contains blanks or other special characters.

# Options

Options

`generate(`*newvar*`)` creates new variable *newvar* containing the match status of each observation in the resulting data. The codes are `1`, `2`, and `3` from the table below.

`nolabel` prevents copying the value-label definitions from the using data to the master. Even if you do not specify this option, label definitions from the using never replace those of the master.

`nonotes` prevents any notes in the using from being incorporated into the master; see [D] **notes**.

`noreport` suppresses the report that `mi merge` ordinarily presents.

`force` allows string/numeric variable type mismatches, resulting in missing values from the using dataset. If omitted, `mi merge` issues an error message; if specified, `mi merge` issues a warning message.

Results

`assert(`*results*`)` specifies how observations should match. If results are not as you expect, an error message will be issued and the master data left unchanged.

| Code | Word | Description |
|---|---|---|
| 1 | master | observation appeared in master only |
| 2 | using | observation appeared in using only |
| 3 | match | observation appeared in both |

(Numeric codes and words are equivalent; you may use either.)

`assert(match)` specifies that all observations in both the master and the using are expected to match, and if that is not so, an error message is to be issued. `assert(match master)` means that all observations match or originally appeared only in the master. See [D] **merge** for more information.

`keep(`*results*`)` specifies which observations are to be kept from the merged dataset. `keep(match)` would specify that only matches are to be kept.

`noupdate` in some cases suppresses the automatic `mi update` this command might perform; see [MI] **noupdate option**.

## Remarks and examples

stata.com

Use mi merge when you would use merge if the data were not mi.

Remarks are presented under the following headings:

*Merging with non-mi data*
*Merging with mi data*
*Merging with mi data containing overlapping variables*

### Merging with non-mi data

Assume that file ipats.dta contains data on the patients in the ICU of a local hospital. The data are mi set, $M = 5$, and missing values have been imputed. File nurses.dta contains information on nurses and is not mi data. You wish to add the relevant nurse information to each patient. Type

```
. use ipats, clear
. mi merge m:1 nurseid using nurses, keep(master)
```

The resulting data are still mi set with $M = 5$. The new variables are unregistered.

### Merging with mi data

Now assume the same situation as above except this time nurses.dta is mi data. Some of the nurse variables have missing values, and those values have been imputed. $M$ is 6. To combine the datasets, you type the same as you would have typed before:

```
. use ipats, clear
. mi merge m:1 nurseid using nurses, keep(master)
```

Remember, $M = 5$ in ipats.dta and $M = 6$ in nurses.dta. The resulting data have $M = 6$, the larger value. There are missing values in the patient variables in $m = 6$, so we need to either impute them or drop the extra imputation by typing mi set M = 5.

### Merging with mi data containing overlapping variables

Now assume the situation as directly above but this time nurses.dta contains variables other than nurseid that also appear in ipats.dta. Such variables—variables in common that are not used as matching variables—are called overlapping variables. Assume seniornurse is such a variable. Let's imagine that seniornurse has no missing values and is unregistered in ipats.dta, but does have missing values and is registered as imputed in nurses.dta.

You will want seniornurse registered as imputed if merging nurses.dta adds new observations that have seniornurse equal to missing. On the other hand, if none of the added observations has seniornurse equal to missing, then you will want the variable left unregistered. And that is exactly what mi merge does. That is,

- Variables unique to the master will be registered according to how they were registered in the master.
- Variables unique to the using will be registered according to how they were registered in the using.
- Variables that overlap will be registered according to how they were in the master if there are no unmatched using observations in the final result.

- If there are such unmatched using observations in the final result, then the unique variables that do not contain missing in the unmatched-and-kept observations will be registered according to how they were registered in the master. So will all variables registered as imputed in the master.
- Variables that do contain missing in the unmatched-and-kept observations will be registered as imputed if they were registered as imputed in the using data or as passive if they were registered as passive in the using data.

Thus variables might be registered differently if we typed

```
. mi merge m:1 nurseid using nurses, keep(master)
```

rather than

```
. mi merge m:1 nurseid using nurses, gen(howmatch)
. keep if howmatch==3
```

If you want to keep the matched observations, it is better to specify `merge`'s `keep()` option.

## Stored results

`mi merge` stores the following in `r()`:

Scalars

| | |
|---|---|
| `r(N_master)` | number of observations in $m$=0 in master |
| `r(N_using)` | number of observations in $m$=0 in using |
| `r(N_result)` | number of observations in $m$=0 in result |
| `r(M_master)` | number of imputations ($M$) in master |
| `r(M_using)` | number of imputations ($M$) in using |
| `r(M_result)` | number of imputations ($M$) in result |

Macros

| | |
|---|---|
| `r(newvars)` | new variables added |

Thus values in the resulting data are

$N$ = # of observations in $m = 0$
= `r(N_result)`

$k$ = # of variables
= *k_master* + `':word count `r(newvars)''`

$M$ = # of imputations
= `max(r(M_master), r(M_using))`
= `r(M_result)`

## Also see

[MI] **Intro** — Introduction to mi

[D] **merge** — Merge datasets

[MI] **mi append** — Append mi data