Title

stata.com

mi import — Import data into mi

Description Syntax Remarks and examples References Also see

Description

`mi import` imports into `mi` data that contain original data and imputed values.

Syntax

```
mi import nhanes1 ...

mi import ice ...

mi import flong ...

mi import flongsep ...

mi import wide ...
```

See [MI] **mi import nhanes1**, [MI] **mi import ice**, [MI] **mi import flong**, [MI] **mi import flongsep**, and [MI] **mi import wide**.

Remarks and examples

stata.com

Remarks are presented under the following headings:

When to use which mi import command

`mi import nhanes1` imports data recorded in the format used by the National Health and Nutrition Examination Survey (NHANES) produced by the National Center for Health Statistics of the U.S. Centers for Disease Control and Prevention (CDC); see http://www.cdc.gov/nchs/nhanes.htm.

`mi import ice` imports data recorded in the format used by `ice` (Royston 2004, 2005a, 2005b, 2007, 2009).

`mi import flong` and `mi import flongsep` import data that are in flong- and flongsep-like format, which is to say, the data are repeated for $m = 0$, $m = 1$, ..., and $m = M$. `mi import flong` imports data in which the information is contained in one file. `mi import flongsep` imports data in which the information is recorded in a collection of files.

`mi import wide` imports data that are in wide-like format, where additional variables are used to record the imputed values.

Import data into Stata before importing into mi

With the exception of `mi import ice`, you must import the data into Stata before you can use `mi import` to import the data into `mi`. `mi import ice` is the exception only because the data are already in Stata format. That is, `mi import` requires that the data be stored in Stata-format `.dta` datasets. You perform the initial import into Stata by using any method described in [D] **import**.

Using mi import nhanes1, ice, flong, and flongsep

Import commands `mi import nhanes1` and `mi import flongsep` produce an flongsep result; `mi import ice` and `mi import flong` produce an flong result. You can use `mi convert` (see [MI] **mi convert**) afterward to convert the result to another style, and we usually recommend that. Before doing that, however, you need to examine the freshly imported data and verify that all imputed and passive variables are registered correctly. If they are not registered correctly, you risk losing imputed values.

To perform this verification, use the `mi describe` (see [MI] **mi describe**) and `mi varying` (see [MI] **mi varying**) commands immediately after `mi import`:

```
. mi import ...
. mi describe
. mi varying
```

`mi describe` will list the registration status of the variables. `mi varying` will report the varying and super-varying variables. Verify that all varying variables are registered as imputed or passive. If one or more is not, register them now:

```
. mi register imputed forgottenvar
. mi register passive another_forgottenvar
```

There is no statistical distinction between imputed and passive variables, so you may register variables about which you are unsure either way. If an unregistered variable is found to be varying and you are convinced that is an error, register the variable as regular:

```
. mi register regular variable_in_error
```

Next, if `mi varying` reports that your data contain any super-varying variables, determine whether the variables are due to errors in the source data or really are intended to be super varying. If they are errors, register the variables as imputed, passive, or regular, as appropriate. Leave any intended super-varying variables unregistered, however, and make a note to yourself: never convert these data to the wide or mlong styles. Data with super-varying variables can be stored only in the flong and flongsep styles.

Now run `mi describe` and `mi varying` again:

```
. mi describe
. mi varying
```

Ensure that you have registered variables correctly, and, if necessary, repeat the steps above to fix any remaining problems.

After that, you may use `mi convert` to switch the data to a more convenient style. We generally start with style wide:

```
. mi convert wide
```

Do not switch to wide, however, if you have any super-varying variables. Try flong instead:

```
. mi convert flong
```

Whichever style you choose, if you get an insufficient-memory error, you will have to either increase the amount of memory dedicated to Stata or use these data in the more inconvenient, but perfectly workable, flongsep style. Concerning increasing memory, see *Converting from flongsep* in [MI] **mi convert**. Concerning the workability of flongsep, see *Advice for using flongsep* in [MI] **styles**.

We said to perform the checks above before using `mi convert`. It is, however, safe to convert the just-imported flongsep data to flong, perform the checks, and then convert to the desired form. The checks will run more quickly if you convert to flong first.

You can vary how you perform the checks. The logic underlying our recommendations is as follows:

- It is possible that you did not specify all the imputed and passive variables when you imported the data, perhaps due to errors in the data's documentation. It is also possible that there are errors in the data that you imported. It is worth checking.
- As long as the imported data are recorded in the flongsep or flong style, unregistered variables will appear exactly as they appeared in the original source. It is only when the data are converted to the wide or mlong style that assumptions about the structure of the data are exploited to save memory. Thus you need to perform checks before converting the data to the more convenient wide or mlong style.
- If you find errors, you could go back and reimport the data correctly, but it is easier to use `mi register` after the fact. When you type `mi register` you are not only informing `mi` about how to deal with the variable but also asking `mi register` to examine the variable and fix any problems given its new registration status.

References

Royston, P. 2004. Multiple imputation of missing values. *Stata Journal* 4: 227–241.

——. 2005a. Multiple imputation of missing values: Update. *Stata Journal* 5: 188–201.

——. 2005b. Multiple imputation of missing values: Update of ice. *Stata Journal* 5: 527–536.

——. 2007. Multiple imputation of missing values: Further update of ice, with an emphasis on interval censoring. *Stata Journal* 7: 445–464.

——. 2009. Multiple imputation of missing values: Further update of ice, with an emphasis on categorical variables. *Stata Journal* 9: 466–477.

Also see

[MI] **intro** — Introduction to mi

[MI] **mi import flong** — Import flong-like data into mi

[MI] **mi import flongsep** — Import flongsep-like data into mi

[MI] **mi import ice** — Import ice-format data into mi

[MI] **mi import nhanes1** — Import NHANES-format data into mi

[MI] **mi import wide** — Import wide-like data into mi

[MI] **styles** — Dataset styles