

**mi import wide** — Import wide-like data into mi

Description  
Remarks and examples

Menu  
Also see

Syntax

Options

## Description

`mi import wide` imports wide-like data, that is, data in which  $m = 0$ ,  $m = 1$ ,  $\dots$ ,  $m = M$  values of imputed and passive variables are recorded in separate variables.

`mi import wide` converts the data to `mi wide` style and `mi sets` the data.

## Menu

Statistics > Multiple imputation

## Syntax

```
mi import wide [ , options ]
```

<i>options</i>	Description
<code>imputed(<i>mvlist</i>)</code>	imputed variables
<code>passive(<i>mvlist</i>)</code>	passive variables
<code>dupsok</code>	allow variable to be posted repeatedly
<code>drop</code>	drop imputed and passive after posting
<code>clear</code>	okay to replace unsaved data in memory

See description of options below for definition of *mvlist*.

## Options

`imputed(mvlist)` and `passive(mvlist)` specify the imputed and passive variables.

For instance, if the data had two imputed variables, `x` and `y`; `x` and `y` contained the  $m = 0$  values; the corresponding  $m = 1$ ,  $m = 2$ , and  $m = 3$  values of `x` were in `x1`, `x2`, and `x3`; and the corresponding values of `y` were in `y1`, `y2`, and `y3`, then the `imputed()` option would be specified as

```
imputed(x=x1 x2 x3 y=y1 y2 y3)
```

If variable `y2` were missing from the data, you would specify

```
imputed(x=x1 x2 x3 y=y1 . y3)
```

The same number of imputations must be specified for each variable.

`dupsok` specifies that it is okay if you specify the same variable name for two different imputations. This would be an odd thing to do, but if you specify `dupsok`, then you can specify

```
imputed(x=x1 x1 x3 y=y1 y2 y3)
```

Without the `dupsok` option, the above would be treated as an error.

`drop` specifies that the original variables containing values for  $m = 1, m = 2, \dots, m = M$  are to be dropped from the data once `mi import wide` has recorded the values. This option is recommended.

`clear` specifies that it is okay to replace the data in memory even if they have changed since they were last saved to disk.

## Remarks and examples

[stata.com](http://www.stata.com)

The procedure to convert wide-like data to `mi wide` style is this:

1. use the unset data; see [\[D\] use](#).
2. Issue the `mi import wide` command.
3. Use `mi describe` (see [\[MI\] mi describe](#)) and `mi varying` (see [\[MI\] mi varying](#)) to verify that the result is as you anticipated.
4. Optionally, use `mi convert` (see [\[MI\] mi convert](#)) to convert the data to what you consider a more convenient style.

For instance, you have been given unset dataset `wi.dta` and have been told that it contains variables `a`, `b`, and `c`; that variable `b` is imputed and contains  $m = 0$  values; that variables `b1` and `b2` contain the  $m = 1$  and  $m = 2$  values; that variable `c` is passive (equal to `a + b`) and contains  $m = 0$  values; and that variables `c1` and `c2` contain the corresponding  $m = 1$  and  $m = 2$  values. Here are the data:

```
. use http://www.stata-press.com/data/r14/wi
(mi prototype)
. list
```

	a	b	c	b1	b2	c1	c2
1.	1	2	3	2	2	3	3
2.	4	.	.	4.5	5.5	8.5	9.5

These are the same data discussed in [\[MI\] styles](#). To import these data, type

```
. mi import wide, imputed(b=b1 b2 c=c1 c2) drop
```

These data are short enough that we can list the result:

```
. list
```

	a	b	c	_mi_miss	_1_b	_2_b	_1_c	_2_c
1.	1	2	3		0	2	2	3
2.	4	.	.		1	4.5	5.5	8.5

Returning to the procedure, we run `mi describe` and `mi varying` on the result:

```
. mi describe
Style: wide
      last mi update 14nov2014 14:44:00, 0 seconds ago

Obs.:  complete          1
      incomplete         1  (M = 2 imputations)
      -----
      total              2

Vars.: imputed:  2; b(1) c(1)
      passive:   0
      regular:   0
      system:    1; _mi_miss
      (there is one unregistered variable; a)

. mi varying
      Possible problem  variable names
-----
      imputed nonvarying:  (none)
      passive nonvarying:  (none)
-----
```

Perhaps you would prefer seeing these data in flong style:

```
. mi convert flong, clear
. list, separator(2)
```

	a	b	c	_mi_miss	_mi_m	_mi_id
1.	1	2	3	0	0	1
2.	4	.	.	1	0	2
3.	1	2	3	.	1	1
4.	4	4.5	8.5	.	1	2
5.	1	2	3	.	2	1
6.	4	5.5	9.5	.	2	2

## Also see

[\[MI\] intro](#) — Introduction to mi

[\[MI\] mi import](#) — Import data into mi