

mi import ice — Import ice-format data into mi

Description

Remarks and examples

Menu

References

Syntax

Also see

Options

Description

`mi import ice` converts the data in memory to `mi` data, assuming the data in memory are in `ice` format. See [Royston \(2004, 2005a, 2005b, 2007, 2009\)](#) for a description of `ice`.

`mi import ice` converts the data to `mi` style flong. The data are `mi set`.

Menu

Statistics > Multiple imputation

Syntax

```
mi import ice [ , options ]
```

<i>options</i>	Description
<code>automatic</code>	register variables automatically
<code>imputed(<i>varlist</i>)</code>	imputed variables to be registered
<code>passive(<i>varlist</i>)</code>	passive variables to be registered
<code>clear</code>	okay to replace unsaved data

Options

`automatic` determines the identity of the imputed variables automatically. Use of this option is recommended.

`imputed(varlist)` specifies the names of the imputed variables. This option may be used with `automatic`, in which case `automatic` is taken to mean automatically determine the identity of imputed variables in addition to the `imputed()` variables specified. It is difficult to imagine why one would want to do this.

`passive(varlist)` specifies the names of the passive variables. This option may be used with `automatic` and usefully so. `automatic` cannot distinguish imputed variables from passive variables, so it assumes all variables that vary are imputed. `passive()` allows you to specify the subset of varying variables that are passive.

Concerning the above options: If none are specified, all variables are left unregistered in the result. You can then use `mi varying` to determine the varying variables and use `mi register` to register them appropriately; see [\[MI\] `mi varying`](#) and [\[MI\] `mi set`](#). If you follow this approach, remember to register imputed variables before registering passive variables.

`clear` specifies that it is okay to replace the data in memory even if they have changed since they were last saved to disk. Remember, `mi import ice` starts with `ice` data in memory and ends with `mi` data in memory.

Remarks and examples

The procedure to convert ice data to mi flong is

1. use the ice data.
2. Issue the `mi import ice` command, preferably with the `automatic` option and perhaps with the `passive()` option, too, although it really does not matter if passive variables are registered as imputed, so long as they are registered.
3. Perform the checks outlined in *Using mi import nhanes1, ice, flong, and flongsep* of **[MI] mi import**.
4. Use `mi convert` (see **[MI] mi convert**) to convert the data to a more convenient style such as wide or mlong.

For instance, you have the following ice data:

```
. use https://www.stata-press.com/data/r17/icedata
. list, separator(2)
```

	<code>_mj</code>	<code>_mi</code>	<code>a</code>	<code>b</code>	<code>c</code>
1.	0	1	1	2	3
2.	0	2	4	.	.
3.	1	1	1	2	3
4.	1	2	4	4.5	8.5
5.	2	1	1	2	3
6.	2	2	4	5.5	9.5

`_mj` and `_mi` are ice system variables. These data contain the original data and two imputations. Variable `b` is imputed, and variable `c` is passive and in fact equal to `a + b`. These are the same data discussed in **[MI] Styles** but in ice format.

The fact that these data are nicely sorted is irrelevant. To import these data, you type

```
. mi import ice, automatic
(1 m=0 obs now marked as incomplete)
```

although it would be even better if you typed

```
. use https://www.stata-press.com/data/r17/icedata
. mi import ice, automatic passive(c)
(1 m=0 obs now marked as incomplete)
```

With the first command, both `b` and `c` will be registered as imputed. With the second, `c` will instead be registered as passive. Whether `c` is registered as imputed or passive makes no difference statistically.

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

```
. list, separator(2)
```

	a	b	c	_mi_m	_mi_id	_mi_miss
1.	1	2	3	0	1	0
2.	4	.	.	0	2	1
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	.

We will now perform the checks outlined in [Using mi import nhanes1, ice, flong, and flongsep](#) of [MI] **mi import**, which are to run `mi describe` and `mi varying` to verify that variables are registered correctly:

```
. mi describe
Style: flong
      last mi update 30apr2021 22:49:44, 0 seconds ago
Observations:
  Complete           1
  Incomplete         1 (M = 2 imputations)
-----
  Total              2
Variables:
  Imputed: 1; b(1)
  Passive: 1; c(1)
  Regular: 0
  System: 3; _mi_m _mi_id _mi_miss
  (there is one unregistered variable; a)
. mi varying
```

	Possible problem	Variable names
	imputed nonvarying:	(none)
	passive nonvarying:	(none)
	unregistered varying:	(none)
	*unregistered super/varying:	(none)
	unregistered super varying:	(none)

* super/varying means super varying but would be varying if registered as imputed; variables vary only where equal to soft missing in $m=0$.

We find that there are no remaining problems, so we convert our data to our preferred wide style:

```
. mi convert wide, clear
. list
```

	a	b	c	_mi_miss	_1_b	_1_c	_2_b	_2_c
1.	1	2	3	0	2	3	2	3
2.	4	.	.	1	4.5	8.5	5.5	9.5

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** — Import data into mi