

intro — Introduction to mi

[Syntax](#)

[Description](#)

[Remarks and examples](#)

[Acknowledgments](#)

[Also see](#)

Syntax

To become familiar with `mi` as quickly as possible, do the following:

1. See [A simple example](#) under *Remarks and examples* below.
2. If you have data that require imputing, see [\[MI\] mi set](#) and [\[MI\] mi impute](#).
3. Alternatively, if you have already imputed data, see [\[MI\] mi import](#).
4. To fit your model, see [\[MI\] mi estimate](#).

To create `mi` data from original data

<code>mi set</code>	declare data to be <code>mi</code> data
<code>mi register</code>	register imputed, passive, or regular variables
<code>mi unregister</code>	unregister previously registered variables
<code>mi unset</code>	return data to unset status (rarely used)

See [Description](#) below for a summary of `mi` data and these commands.

See [\[MI\] Glossary](#) for a definition of terms.

To import data that already have imputations for the missing values (do not `mi set` the data)

<code>mi import</code>	import <code>mi</code> data
<code>mi export</code>	export <code>mi</code> data to non-Stata application

Once data are `mi set` or `mi imported`

<code>mi query</code>	query whether and how <code>mi set</code>
<code>mi describe</code>	describe <code>mi</code> data
<code>mi varying</code>	identify variables that vary over <i>m</i>
<code>mi misstable</code>	tabulate missing values
<code>mi passive</code>	create passive variable and register it

To perform estimation on mi data

<code>mi impute</code>	impute missing values
<code>mi estimate</code>	perform and combine estimation on $m > 0$
<code>mi ptrace</code>	check stability of MCMC
<code>mi test</code>	perform tests on coefficients
<code>mi testtransform</code>	perform tests on transformed coefficients
<code>mi predict</code>	obtain linear predictions
<code>mi predictnl</code>	obtain nonlinear predictions

To `stset`, `svyset`, `tsset`, or `xtset` any mi data that were not set at the time they were mi set

<code>mi fvset</code>	<code>fvset</code> for mi data
<code>mi svyset</code>	<code>svyset</code> for mi data
<code>mi xtset</code>	<code>xtset</code> for mi data
<code>mi tsset</code>	<code>tsset</code> for mi data
<code>mi stset</code>	<code>stset</code> for mi data
<code>mi streset</code>	<code>streset</code> for mi data
<code>mi st</code>	<code>st</code> for mi data

To perform data management on mi data

<code>mi rename</code>	rename variable
<code>mi append</code>	append for mi data
<code>mi merge</code>	merge for mi data
<code>mi expand</code>	expand for mi data
<code>mi reshape</code>	reshape for mi data
<code>mi stsplitt</code>	<code>stsplitt</code> for mi data
<code>mi stjoin</code>	<code>stjoin</code> for mi data
<code>mi add</code>	add imputations from one mi dataset to another

To perform data management for which no mi prefix command exists

<code>mi extract</code>	extract $m = 0$ data
<code>...</code>	perform data management the usual way
<code>mi replace0</code>	replace $m = 0$ data in mi data

To perform the same data management or data-reporting command(s) on $m = 0, m = 1, \dots$

<code>mi xeq: ...</code>	execute commands on $m = 0, m = 1, m = 2, \dots, m = M$
<code>mi xeq #: ...</code>	execute commands on $m = \#$
<code>mi xeq # # ...: ...</code>	execute commands on specified values of m

Useful utility commands

<code>mi convert</code>	convert mi data from one style to another
<code>mi extract #</code>	extract $m = \#$ from mi data
<code>mi select #</code>	programmer's command similar to <code>mi extract</code>
<code>mi copy</code>	copy mi data
<code>mi erase</code>	erase files containing mi data
<code>mi update</code>	verify/make mi data consistent
<code>mi reset</code>	reset imputed or passive variable

For programmers interested in extending mi

[MI] [technical](#) Detail for programmers

Summary of styles

There are four styles or formats in which mi data are stored: flongsep, flong, mlong, and wide.

1. Flongsep: $m = 0, m = 1, \dots, m = M$ are each separate `.dta` datasets. If $m = 0$ data are stored in `pat.dta`, then $m = 1$ data are stored in `_1_pat.dta`, $m = 2$ in `_2_pat.dta`, and so on. Flongsep stands for *full long and separate*.
2. Flong: $m = 0, m = 1, \dots, m = M$ are stored in one dataset with $_N = N + M \times N$ observations, where N is the number of observations in $m = 0$. Flong stands for *full long*.
3. Mlong: $m = 0, m = 1, \dots, m = M$ are stored in one dataset with $_N = N + M \times n$ observations, where n is the number of incomplete observations in $m = 0$. Mlong stands for *marginal long*.
4. Wide: $m = 0, m = 1, \dots, m = M$ are stored in one dataset with $_N = N$ observations. Each imputed and passive variable has M additional variables associated with it. If variable `bp` contains the values in $m = 0$, then values for $m = 1$ are contained in variable `_1_bp`, values for $m = 2$ in `_2_bp`, and so on. Wide stands for *wide*.

See *style* in [MI] [Glossary](#) and see [MI] [styles](#) for examples. See [MI] [technical](#) for programmer's details.

Description

The `mi` suite of commands deals with multiple-imputation data, abbreviated as `mi` data.

In summary,

1. `mi` data may be stored in one of four formats—`flongsep`, `flong`, `mlong`, and `wide`—known as styles. Descriptions are provided in [Summary of styles](#) directly above.
2. `mi` data contain M imputations numbered $m = 1, 2, \dots, M$, and contain $m = 0$, the original data with missing values.
3. Each variable in `mi` data is registered as imputed, passive, or regular, or it is unregistered.
 - a. Unregistered variables are mostly treated like regular variables.
 - b. Regular variables usually do not contain missing, or if they do, the missing values are not imputed in $m > 0$.
 - c. Imputed variables contain missing in $m = 0$, and those values are imputed, or are to be imputed, in $m > 0$.
 - d. Passive variables are algebraic combinations of imputed, regular, or other passive variables.
4. If an imputed variable contains a value greater than `.` in $m = 0$ —it contains `.a`, `.b`, `...`, `.z`—then that value is considered a hard missing and the missing value persists in $m > 0$.

See [\[MI\] Glossary](#) for a more thorough description of terms used throughout this manual.

Remarks and examples

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

Remarks are presented under the following headings:

[A simple example](#)

[Suggested reading order](#)

A simple example

We are about to type six commands:

```
. use http://www.stata-press.com/data/r13/mheart5      (1)
. mi set mlong                                       (2)
. mi register imputed age bmi                       (3)
. set seed 29390                                     (4)
. mi impute mvn age bmi = attack smokes hsgrad female, add(10) (5)
. mi estimate: logistic attack smokes age bmi hsgrad female (6)
```

The story is that we want to fit

```
. logistic attack smokes age bmi hsgrad female
```

but the `age` and `bmi` variables contain missing values. Fitting the model by typing `logistic ...` would ignore some of the information in our data. Multiple imputation (MI) attempts to recover that information. The method imputes M values to fill in each of the missing values. After that, statistics are performed on the M imputed datasets separately and the results combined. The goal is to obtain better estimates of parameters and their standard errors.

In the solution shown above,

1. We load the data.
2. We set our data for use with mi.
3. We inform mi which variables contain missing values for which we want to impute values.
4. We impute values in command 5; we prefer that our results be reproducible, so we set the random-number seed in command 4. This step is optional.
5. We create $M = 10$ imputations for each missing value in the variables we registered in command 3.
6. We fit the desired model separately on each of the 10 imputed datasets and combine the results.

The results of running the six-command solution are

```
. use http://www.stata-press.com/data/r13/mheart5
(Fictional heart attack data)

. mi set mlong

. mi register imputed age bmi
(28 m=0 obs. now marked as incomplete)

. set seed 29390

. mi impute mvn age bmi = attack smokes hsgrad female, add(10)

Performing EM optimization:
note: 12 observations omitted from EM estimation because of all imputation
      variables missing
      observed log likelihood = -651.75868 at iteration 7

Performing MCMC data augmentation ...

Multivariate imputation                Imputations =      10
Multivariate normal regression         added =          10
Imputed: m=1 through m=10              updated =          0
Prior: uniform                          Iterations =     1000
                                          burn-in =       100
                                          between =       100
```

Variable	Observations per m			
	Complete	Incomplete	Imputed	Total
age	142	12	12	154
bmi	126	28	28	154

(complete + incomplete = total; imputed is the minimum across m of the number of filled-in observations.)

```

. mi estimate: logistic attack smokes age bmi hsgrad female
Multiple-imputation estimates      Imputations      =      10
Logistic regression                Number of obs    =     154
                                   Average RVI       =     0.1031
                                   Largest FMI       =     0.3256
DF adjustment: Large sample        DF: min         =     92.90
                                   avg              =    25990.98
                                   max              =     77778.66
Model F test: Equal FMI           F( 5, 3279.8)   =     3.27
Within VCE type: OIM              Prob > F        =     0.0060

```

attack	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
smokes	1.18324	.3605462	3.28	0.001	.4765251	1.889954
age	.0321028	.016145	1.99	0.047	.0004071	.0637984
bmi	.1100667	.0546424	2.01	0.047	.0015561	.2185772
hsgrad	.1413171	.4043884	0.35	0.727	-.6512819	.933916
female	-.0759589	.416927	-0.18	0.855	-.8931367	.7412189
_cons	-5.38815	1.85184	-2.91	0.004	-9.047656	-1.728644

Note that the output from the last command,

```
. mi estimate: logistic attack smokes age bmi hsgrad female
```

reported coefficients rather than odds ratios, which `logistic` would usually report. That is because the estimation command is not `logistic`, it is `mi estimate`, and `mi estimate` happened to use `logistic` to obtain results that `mi estimate` combined into its own estimation results.

`mi estimate` by default displays coefficients. If we now wanted to see odds ratios, we could type

```
. mi estimate, or
(output showing odds ratios would appear)
```

Note carefully: We replay results by typing `mi estimate`, not by typing `logistic`. If we had wanted to see the odds ratios from the outset, we would have typed

```
. mi estimate, or: logistic attack smokes age bmi hsgrad female
```

Suggested reading order

The order of suggested reading of this manual is

- [MI] [intro substantive](#)
- [MI] [intro](#)
- [MI] [Glossary](#)
- [MI] [workflow](#)
- [MI] [mi set](#)
- [MI] [mi import](#)
- [MI] [mi describe](#)
- [MI] [mi misstable](#)
- [MI] [mi impute](#)
- [MI] [mi estimate](#)
- [MI] [mi estimate postestimation](#)
- [MI] [styles](#)
- [MI] [mi convert](#)
- [MI] [mi update](#)

[MI] [mi rename](#)

[MI] [mi copy](#)

[MI] [mi erase](#)

[MI] [mi XXXset](#)

[MI] [mi extract](#)

[MI] [mi replace0](#)

[MI] [mi append](#)

[MI] [mi add](#)

[MI] [mi merge](#)

[MI] [mi reshape](#)

[MI] [mi stsplit](#)

[MI] [mi varying](#)

Programmers will want to see [MI] [technical](#).

What's new

For a complete list of all the new features in Stata 13, see [U] [1.3 What's new](#).

Acknowledgments

We thank Jerry (Jerome) Reiter of the Department of Statistical Science at Duke University; Patrick Royston of the MRC Clinical Trials Unit, London, and coauthor of the Stata Press book *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model*; and Ian White of the MRC Biostatistics Unit, London, for their comments and assistance in the development of `mi`. We also thank for their comments James Carpenter of the London School of Hygiene and Tropical Medicine and Jonathan Sterne of the School of Social and Community Medicine at the University of Bristol, UK, and editor of the Stata Press book *Meta-Analysis in Stata: An Updated Collection from the Stata Journal*.

Previous and still ongoing work on multiple imputation in Stata influenced the design of `mi`. For their past and current contributions, we thank Patrick Royston and Ian White again for `ice`; John Carlin and John Galati, both of the Murdoch Children's Research Institute and University of Melbourne, and Patrick Royston and Ian White (yet again) for `mim`; John Galati for `inorm`; and Rodrigo Alfaro of the Banco Central de Chile for `mira`.

Also see

[MI] [intro substantive](#) — Introduction to multiple-imputation analysis

[MI] [Glossary](#)

[MI] [styles](#) — Dataset styles

[MI] [workflow](#) — Suggested workflow

[U] [1.3 What's new](#)