Description   Remarks and examples   Acknowledgments   Also see

Description

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

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

See *Summary* 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)

`mi import`	import `mi` data
`mi export`	export `mi` data to non-Stata application

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

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

To perform estimation on mi data

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

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

mi fvset	fvset for mi data
mi svyset	svyset for mi data
mi xtset	xtset for mi data
mi tsset	tsset for mi data
mi stset	stset for mi data
mi streset	streset for mi data
mi st	st for mi data

To perform data management on mi data

mi rename	rename variable
mi append	append for mi data
mi merge	merge for mi data
mi expand	expand for mi data
mi reshape	reshape for mi data
mi stsplit	stsplit for mi data
mi stjoin	stjoin for mi data
mi add	add imputations from one mi dataset to another

To perform data management for which no mi prefix command exists

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

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

`mi xeq: ...`	execute commands on $m = 0, m = 1, m = 2, \ldots, m = M$
`mi xeq #: ...`	execute commands on $m = \#$
`mi xeq # # ...: ...`	execute commands on specified values of m

Useful utility commands

`mi convert`	convert `mi` data from one style to another
`mi extract #`	extract $m = \#$ from `mi` data
`mi select #`	programmer's command similar to `mi extract`
`mi copy`	copy `mi` data
`mi erase`	erase files containing `mi` data
`mi update`	verify/make `mi` data consistent
`mi reset`	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: $m = 0$, $m = 1$, ..., $m = M$ 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: $m = 0$, $m = 1$, ..., $m = M$ 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: $m = 0$, $m = 1$, ..., $m = M$ 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: $m = 0$, $m = 1$, ..., $m = M$ 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.

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 $m = 1, 2, \ldots, M$, 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

Remarks are presented under the following headings:

A simple example
Suggested reading order

A simple example

We are about to type six commands:

```
. use https://www.stata-press.com/data/r19/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

```
. webuse 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       =     0.0835
                                                Largest FMI       =     0.2642
DF adjustment:   Large sample                   DF:     min       =     139.75
                                                        avg       =  19,591.87
                                                        max       =  67,578.07
Model F test:       Equal FMI                   F(   5, 4836.6)   =       3.32
Within VCE type:          OIM                   Prob > F          =     0.0054

      attack   Coefficient  Std. err.      t    P>|t|     [95% conf. interval]

      smokes     1.187152   .3623514     3.28   0.001     .4768502    1.897453
         age     .0315179   .0163884     1.92   0.055    -.0006696    .0637055
         bmi     .1090419   .0516554     2.11   0.037     .0069434    .2111404
      hsgrad     .1712372   .4054594     0.42   0.673     -.623472    .9659464
      female     -.065744   .4156809    -0.16   0.874    -.8804781    .7489901
       _cons    -5.369962   1.863821    -2.88   0.005    -9.054895   -1.685029
```

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**.

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, who is coeditor 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 of the Murdoch Children's Research Institute and University of Melbourne, and John Galati of the Australian Public Service, and Patrick Royston and Ian White (yet again) for `mim`; John Galati for `inorm`; and Rodrigo Alfaro, while at 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**