Multiple-imputation analysis using Stata's `mi` command

Yulia Marchenko

Senior Statistician
StataCorp LP

2009 UK Stata Users Group Meeting

Outline

What is multiple imputation?

- Multiple imputation (MI) is a simulation-based approach for analyzing incomplete data.
- MI replaces missing values with multiple sets of simulated values to complete the data, applies standard analyses to each completed dataset, and adjusts the obtained parameter estimates for missing-data uncertainty.
- The objective of MI is not to predict missing values as close as possible to the true ones but to handle missing data in a way resulting in valid statistical inference (Rubin 1996).

Why use multiple imputation?

- It is more flexible than fully-parametric methods, e.g. maximum likelihood, purely Bayesian analysis.
- It can be more efficient than listwise deletion (complete-cases analysis) and can correct for potential bias.
- It accounts for missing-data uncertainty and, thus, does not underestimate the variance of estimates unlike single imputation methods.

MI technique

The MI technique consists of three steps:

1. *Imputation*. Replace missing values with M sets of plausible values according to an imputation model (e.g., Rubin 1987; Schafer 1997) to create M completed datasets.
2. *Completed-data analysis*. Perform primary analysis on each imputed (completed) dataset to obtain a set of completed-data estimates $\hat{\mathbf{q}}_i$ and their respective VCEs $\hat{\mathbf{U}}_i$, $i = 1, \ldots, M$.
3. *Pooling*. Consolidate results from the completed-data analyses $\{\hat{\mathbf{q}}_i, \hat{\mathbf{U}}_i\}_{i=1}^{M}$ into one MI inference using Rubin's combination rules (e.g. Rubin 1987, 76).

Statistical validity of MI

MI yields statistically valid inference if

- an imputation method used in step 1 is proper per Rubin (1987, 118–119).

  Rubin recommends drawing imputations from a Bayesian posterior predictive distribution of missing data to ensure that imputations are proper.

- the primary, completed-data analysis used in step 2 is statistically valid in the absence of missing data; see Rubin (1987, 116–118) for details.

Notation and some terminology

- *Original data* — data containing missing values.
- With a slight abuse of terminology, by an *imputation* we mean a copy of the original data in which missing values are imputed.
- M denotes the number of imputations.
- $m\,(=0,\ldots,M)$ refers to the original or imputed data: $m=0$ means original data and $m>0$ means imputed data. $m=1$ means the first imputation, $m=2$ means the second imputation, etc.

Main features

Stata 11's `mi` command provides full support for all three steps of the MI technique:

- `mi impute` performs imputation (step 1);
- `mi estimate` performs individual analyses, collects estimates of coefficients and their VCEs, applies Rubin's combination rules to the collected estimates, and reports final results (steps 2 and 3).

In addition, `mi` offers full data management of multiply-imputed data: you can create or drop variables, observations as if you were working with one dataset — `mi` will replicate the changes correctly across the imputed datasets.

Other unique features of `mi`:

- the ability to store multiply-imputed data in different formats — `mi` data styles;
- the ability to verify consistency of the data across multiple copies.

`mi` data system

The `mi` command tracks information about the data which allows full integration of statistical and data-management components. `mi` records:

- the format (style) in which MI data will be stored: `wide`, `mlong`, `flong`, or `flongsep`;
- the number of imputations, M;
- the types of registered variables: `imputed`, `passive`, `regular`;
- information about complete and incomplete observations;
- see **[MI] technical** for more detail.

`mi` views system missing values (`.`) as missing values to be imputed — *soft missing*. All other, extended, missing values are viewed as missing values not to be imputed — *hard missing values*.

The `mi set` and `mi register` commands are used to set up `mi` data.

`mi` data styles

To use `mi` you must declare the storage style.

`mi` supports 4 styles (formats) for storing MI data:

- `flongsep` — full long and separate — imputed data are in separate files, one per imputation;
- `flong` — full long — original and imputed data are in one file, imputations are saved as extra observations;
- `mlong` — marginal long — original and imputed data are in one file, only observations containing imputed values are saved as extra observations. `mlong` is a memory-efficient version of `flong`;
- `wide` — wide — original and imputed data are in one file, imputations are saved as extra variables.

Some tasks are easier in one style than another. You can switch from one style to another during your `mi` session by using `mi convert`.

The role of registered variables in `mi`

`mi` uses a variable's status to verify its consistency across imputations. You can register variables by using `mi register`. Registering variables is, in general, not required but highly recommended.

`mi` distinguishes 3 types of variables:

- imputation (`imputed`) — variables containing soft missings (system missing values), i.e. missing values to be filled in;
- passive (`passive`) — variables which are functions of imputation and or other passive variables;
- regular (`regular`) — variables which are the same across imputations;
- other variables are treated as unregistered.

Always register imputation variables!

Important: the status of each observation, complete or incomplete, is determined based on registered imputation variables. An observation in which at least one imputation variable contains a soft missing is marked as incomplete. If no variables are registered as `imputed`, all observations are treated as complete. You should always register imputation variables.

Example

Consider fictional data recording heart attacks. The objective is to examine a relationship between smoking and heart attacks adjusting for age, body mass index, gender, and educational status.

```
. webuse mheart0
(Fictional heart attack data; bmi missing)

. describe attack smokes age bmi female hsgrad

              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------
attack          byte   %9.0g                  Outcome (heart attack)
smokes          byte   %9.0g                  Current smoker
age             float  %9.0g                  Age, in years
bmi             float  %9.0g                  Body Mass Index, kg/m^2
female          byte   %9.0g                  Gender
hsgrad          byte   %9.0g                  High school graduate
```

We examine data for missing values using `misstable`.

```
. misstable summarize
                                                          Obs<.
                                                +------------------------------
               |                                | Unique
      Variable |     Obs=.     Obs>.     Obs<.  | values        Min         Max
  -------------+--------------------------------+------------------------------
           bmi |        22                 132  |    132   17.22643    38.24214
  -----------------------------------------------------------------------------
```

Variable `bmi` contains 22 missing values. We use multiple imputation to perform analysis of the heart-attack data.

Following the MI technique, we need to impute missing values of `bmi` and then analyze the resulting multiply-imputed data. We will use `mi impute` and `mi estimate` to do this but first we need to declare our data to be `mi` data.

```
. /* check if data are -mi set- */
. mi query
(data not mi set)

. /* declare MI data to be stored in the marginal long style */
. mi set mlong

. /* register bmi as imputation variable */
. mi register imputed bmi
(22 m=0 obs. now marked as incomplete)

. /* register other variables as regular */
. mi register regular attack smokes age female hsgrad
```

- We use the regression imputation method to fill in missing values of `bmi`.
- We arbitrarily create 5 imputations.
- We also set the random-number seed for reproducibility.

```
. mi impute regress bmi attack smokes age female hsgrad, add(5) rseed(123)

Univariate imputation                    Imputations =        5
Linear regression                              added =        5
Imputed: m=1 through m=5                     updated =        0

               |              Observations per m
               |----------------------------------------------
      Variable |   complete   incomplete   imputed |     total
---------------+-----------------------------------+----------
           bmi |        132           22        22 |       154
--------------------------------------------------------------
(complete + incomplete = total; imputed is the minimum across m
 of the number of filled in observations.)
```

We perform logistic analysis of the multiply-imputed data using `mi estimate: logit`.

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

Multiple-imputation estimates                     Imputations     =          5
Logistic regression                               Number of obs   =        154
                                                  Average RVI     =     0.0564
DF adjustment:   Large sample                     DF:      min    =      78.77
                                                           avg    =   14754.79
                                                           max    =   39201.13
Model F test:       Equal FMI                     F(   5, 3527.0) =       3.39
Within VCE type:          OIM                     Prob > F        =     0.0047

------------------------------------------------------------------------------
      attack |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      smokes |   1.193653   .3579481     3.33   0.001      .492038    1.895268
         age |   .0360079   .0155205     2.32   0.020     .0055845    .0664314
         bmi |   .0985092   .0516418     1.91   0.060     -.004286    .2013044
      female |   -.113328   .4165623    -0.27   0.786    -.9298195    .7031636
      hsgrad |   .1555202   .4034539     0.39   0.700    -.6352593    .9462997
       _cons |  -5.329907   1.800598    -2.96   0.004    -8.893172   -1.766643
------------------------------------------------------------------------------
```

Imputation methods

- Univariate imputation:
  - linear regression for continuous variables — `mi impute regress`
  - predictive mean matching for continuous variables — `mi impute pmm`
  - logistic regression for binary variables — `mi impute logit`
  - ordinal logistic regression for ordinal variables — `mi impute ologit`
  - multinomial logistic regression for nominal variables — `mi impute mlogit`
- Multivariate imputation:
  - monotone method for multiple variables of different types — `mi impute monotone`
  - multivariate normal regression for multiple continuous variables — `mi impute mvn`

Imputation methods — assumption

- `mi impute` assumes that missing data are missing at random; that is, missing values do not carry any extra information about why they are missing than what is already available in the observed data.
- `mi impute` creates imputations by simulating from a (approximate) Bayesian posterior predictive distribution of the missing data, following Rubin's recommendation.

`mi impute`'s general syntax

`mi impute` *method* *model_spec* [, *common_options* *method_options*]

The two main common options are `add()` and `replace`. These options allow you to perform the following actions:

1. Create imputations or add new imputations to the existing ones:
   `mi impute ..., add(#) ...`
2. Replace existing imputations with new ones:
   `mi impute ..., replace ...`
3. Replace existing imputations and add new ones:
   `mi impute ..., add(#) replace ...`

See **[MI] mi impute** for more details.

Multivariate imputation: heart-attack data

```
. webuse mheart5s0
(Fictional heart attack data; bmi and age missing)

. mi describe

  Style:  mlong
          last mi update 19jun2009 10:50:18, 30 days ago

  Obs.:   complete          126
          incomplete         28  (M = 0 imputations)
          _____________________
          total             154

  Vars.:  imputed:  2; bmi(28) age(12)

          passive: 0

          regular: 4; attack smokes female hsgrad

          system:  3; _mi_m _mi_id _mi_miss

         (there are no unregistered variables)
```

- Data are already set in the `mlong` style.
- Two variables, `bmi` and `age`, contain missing values and are registered as imputed.
- Other variables are registered as regular.
- Data contain no imputations.
- System variable `_mi_miss` records the status of observations (1 means incomplete) based on imputation variables `age` and `bmi`.
- `_mi_m` records imputation numbers and `_mi_id` records observation identifiers; see **[MI] technical** for details.

Multivariate imputation: monotone missing pattern

```
. mi misstable nested

     1.  age(12) -> bmi(28)

. mi impute monotone (pmm) bmi  (regress) age  = attack smokes hsgrad female, add(5)

Conditional models:
           age: regress age attack smokes hsgrad female
           bmi: pmm bmi age attack smokes hsgrad female

Multivariate imputation                      Imputations =        5
Monotone method                                    added =        5
Imputed: m=1 through m=5                         updated =        0

           age: linear regression
           bmi: predictive mean matching

                             Observations per m
              ----------------------------------------------
    Variable    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.)
```

- Note that because the data are already `mi set`, we used `mi misstable` rather than `misstable`.
- We used `mi misstable nested` to check if variables are nested with respect to missing values.
- From the output of `mi misstable`, missing values of `age` and `bmi` form a monotone-missing pattern: `age` is missing only in observations where `bmi` is missing. `bmi` does not have any observations with nonmissing values for which `age` is missing.
- Therefore, we used `mi impute monotone` to impute `bmi` and `age`.
- We can also use `mi impute mvn` to impute `bmi` and `age` (as shown on the next slide) but using `mi impute monotone` is faster because it does not require iteration.

Multivariate imputation: normal regression

```
. mi impute mvn age bmi = attack smokes hsgrad female, replace
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 =         5
Multivariate normal regression                    added =         0
Imputed: m=1 through m=5                        updated =         5
Prior: uniform                               Iterations =       500
                                                burn-in =       100
                                                between =       100

                              Observations per m
             ----------------------------------------------------------
    Variable |   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 impute mvn` uses data augmentation, an iterative MCMC method, to impute missing values under a multivariate normal model.
- `mi impute mvn` uses estimates from the EM algorithm as starting values for the MCMC procedure. You can supply your own initial values, if needed, using option `initmcmc()`.
- The default prior is uniform under which posterior mode estimates and maximum-likelihood estimates are equivalent. You can change the default prior specification using option `prior()`.
- The first imputation is drawn after an initial default burn-in period of 100 iterations. You can use option `burnin()` to choose a different burn-in period.
- The subsequent imputations are drawn every 100 (the default) iterations apart. You can change the number of iterations between imputations using option `burnbetween()`.

Importing existing imputations to `mi`

In the heart-attack example we created imputations using `mi impute`. What if you need to analyze multiply-imputed data created outside of Stata?

1. Read file(s) containing multiply-imputed data into Stata; see, for example, **[D] infile**.
2. Use `mi import` to set up the multiply-imputed data in `mi`.

`mi import` supports various styles in which multiply-imputed data can be recorded. In particular,

- `mi import nhanes1` imports MI data recorded in the format used by NHANES; see http://www.cdc.gov/nchs/nhanes.htm.
- `mi import ice` imports MI data recorded in the format used by the user-written command `ice` performing imputation via chained equations.
- see **[MI] mi import** for details.

Describing `mi` data

To describe the `mi` data use

- `mi query` to get a short summary of the `mi` settings;
- `mi describe` to get a more detailed report about `mi` data.

`mi varying` is useful to identify variables that vary over imputations. For example, you can use it to identify imputation and passive variables and then register them using `mi register`. This command also helps to detect potential problems.

Manipulating `mi` data

Manipulation of `mi` data can be done in one of two ways:

- repeating the same data-management command on each imputed dataset;
- using a data-management routine specialized for multiply-imputed data. For example, specialized routines are needed to append or merge multiply-imputed data.

Stata offers both:

- Use `mi xeq:`*command* to perform *command* on each imputed dataset.
- Use, e.g. `mi append`, `mi merge`, `mi reshape` to append, merge, and reshape `mi` data; see **[MI] intro** (or type `help mi`) for a list of all `mi`-specific data-management commands.

Verifying consistency of the `mi` data — `mi update`

`mi update` is a unique feature of `mi`. `mi update` verifies that:

- the number of observations is consistent in $m \geq 0$;
- the number and instances of variables are consistent in $m \geq 0$;
- complete/incomplete observations are correctly identified by the imputation variables;
- regular variables contain the same values in imputed data as in the original data;
- imputation variables contain the same nonmissing values in imputed data as in the original data;
- passive variables contain the same values in complete observations in imputed data as in the original data;
- . . . ; see **[MI] mi update** for more detail.

`mi update` is executed automatically each time an `mi` command is run. It can also be run manually.

Managing observations, variables — a quick example

Example

`mi` data contain 1 imputation and are saved in the `flongsep` style.

1. Replace a value:

```
. mi xeq: replace age = 20 in 30
 m=0 data:
-> replace age = 20 in 30
(1 real change made)
 m=1 data:
-> replace age = 20 in 30
(1 real change made)
```

2. Drop a variable:

```
. mi xeq: drop female
 m=0 data:
-> drop female
 m=1 data:
-> drop female
```

3. Alternatively to 1 and 2 above,

```
. replace age = 20 in 30
(1 real change made)
. drop female
. mi update
(regular variable female unregistered because not in m=0)
(imputed variables updated in 1 obs. in m>0 in order to match m=0 data)
```

Managing imputations

You can use

- `mi impute` to create or add new imputations;
- `mi set m` to delete selected imputations;
- `mi add` to add imputations from a separate file;
- `mi set M` to reset the number of imputations (or create empty imputations in which missing data are not filled in).

When performing data manipulation on `mi` data, remember

- to use the `mi` versions of the data-management routines, if they exist;
- to use `mi xeq` with routines for which there is no `mi` prefix;
- to run `mi update` periodically to ensure consistency of the `mi` data.

Estimation using multiple imputation

`mi estimate` is designed to perform analysis of the multiply-imputed data; the data must be `mi set` and must have at least 2 imputations.

Basic syntax:

`mi estimate:` *estimation_command*

The above runs *estimation_command* separately on each imputed dataset, and reports the MI estimates of coefficients and their standard errors based on results saved by *estimation_command*. *estimation_command* is one of the supported estimation commands as listed in **[MI] estimation**.

Extended syntax:

`mi estimate` [, *options*]: *estimation_command*

`mi estimate` also provides *options* allowing to select how many and which imputations to use in the computation, to report additional information about the MI estimates, and more.

Other features of `mi estimate`

Other features of `mi estimate` include

- the ability to obtain estimates of functions of coefficients;
- the ability to save individual estimates for later use with `mi estimate using` without the need of refitting the models on each imputed dataset.

Other features of mi estimate

For example, using our earlier heart-attack data example, we save individual estimates to a `myest.ster` estimation file:

```
. mi estimate, saving(myest): logit attack smokes age bmi female hsgrad
```

We then compute MI estimates of the ratio of coefficients for age and `bmi` using saved individual estimation results rather than refitting the completed-data models:

```
. mi estimate (ratio: _b[age]/_b[bmi]) using myest
```

See **[MI] mi estimate** and **[MI] mi estimate using** for more detail.

Analyzing complex multiply-imputed data

You can use `mi estimate` to analyze complex data such as, for example, survey data. In Stata, prior to analyzing complex data, it must be declared. For example, survey data must be `svyset`, survival data must be `stset`, and so on.

To declare the complex `mi` data, you should use the corresponding set command with the `mi` prefix. For example, to declare `mi` survey data, use

```
. mi svyset ...
```

Then, to fit a model on `mi` survey data, use

```
. mi estimate: svy: ...
```

Using `mi estimate` with user-written commands

`mi estimate` provides a way (option `cmdok`) of applying combination rules to results from estimation commands not officially supported by `mi estimate`, such as user-written estimation commands.

```
. mi estimate, cmdok : user_command
```

It is the user's responsibility to verify that the combination rules are applicable to the results reported by the used command. The *user_command* should also satisfy technical requirements from "Writing programs for use with `mi`" in **[P] program properties**.

Authors of user-written commands can also modify their estimation commands to be accepted by `mi estimate` without specifying `cmdok` as described in the section mentioned above.

Testing hypotheses

After `mi estimate`, you can test subsets of coefficients, linear or nonlinear hypotheses using `mi test` and `mi testtransform`. See **[MI] postestimation** for examples.

`mi test` and `mi testtransform` provide

- the conditional (equal fraction-missing-information, equal FMI) test of Li et al. (1991);
- the unconditional test of Rubin (1987, 77–78). This test may be preferable when the number of imputations is large and the equal FMI assumption is suspect.
- small-sample adjustments for the tests as described in Marchenko and Reiter (2009).

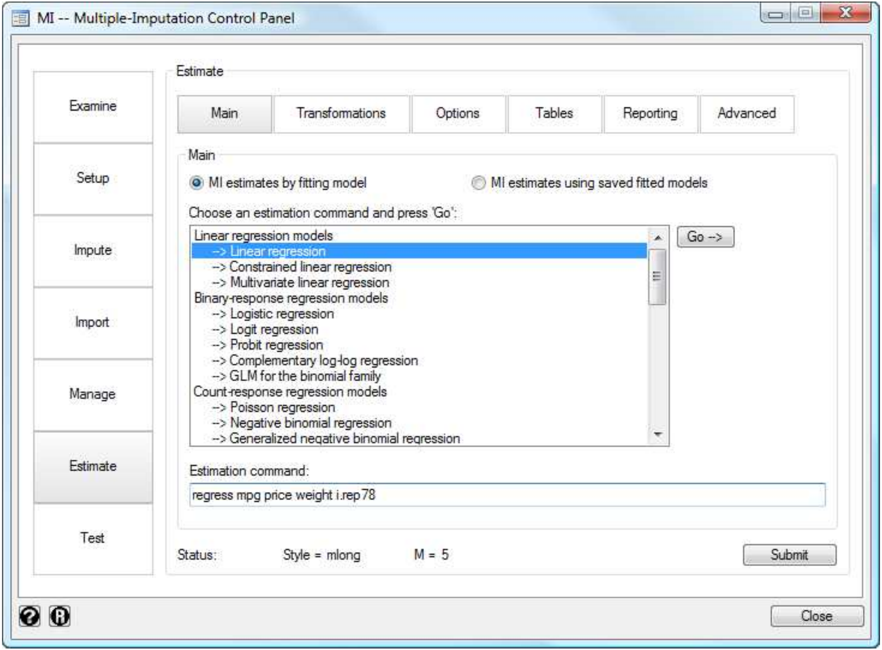
GUI — MI control panel

The MI control panel, which can be invoked from the **Statistics** > **Multiple imputation** menu or by typing

```
. db mi
```

guides you through all the phases of MI.

(NEXT SLIDE)


MI -- Multiple-Imputation Control Panel
Examine
Setup
Impute
Import
Manage
Estimate
Test
Estimate
Main
Transformations
Options
Tables
Reporting
Advanced
Main
MI estimates by fitting model
MI estimates using saved fitted models
Choose an estimation command and press 'Go':
Linear regression models
--> Linear regression
--> Constrained linear regression
--> Multivariate linear regression
Binary-response regression models
--> Logistic regression
--> Logit regression
--> Probit regression
--> Complementary log-log regression
--> GLM for the binomial family
Count-response regression models
--> Poisson regression
--> Negative binomial regression
--> Generalized negative binomial regression
Go -->
Estimation command:
regress mpg price weight i.rep78
Status:
Style = mlong
M = 5
Submit
Close

Relation to existing user-written commands

User-written commands `ice` and `mim` are widely used to perform multiple-imputation analysis in Stata. `ice` creates imputations using the chained-equation approach of van Buuren et al. (1999). `mim` analyzes multiply-imputed data.

- `mi estimate` and the `mi` data-management routines cover most estimation and all data-management capabilities of `mim`.
- `mi` does not provide imputation via chained equations and thus `ice` remains the only implementation of the chained-equation approach.
- See http://www.stata.com/statalist/archive/2009-08/msg00385.html for more detail.
- A more detailed FAQ is coming soon.

Summary

- `mi` accommodates all steps of the MI technique:
  - `mi impute` provides univariate and multivariate methods for filling in missing values;
  - `mi estimate` performs completed-data analysis and combines estimates using Rubin's pooling rules.
- `mi` provides full data-management support.
- `mi` provides 4 styles for storing MI data and can import from 5 styles.
- `mi` verifies consistency of your data at every opportunity.
- `mi` offers postestimation features: testing linear or nonlinear hypotheses.
- `mi` provides elaborate GUI support — MI control panel.
- `mi` offers extensive documentation, manual **[MI] Multiple imputation**.

References

Li, K.-H., T. E. Raghunathan, and D. B. Rubin. 1991. Large-sample significance levels from multiply imputed data using moment-based statistics and an F reference distribution. *Journal of the American Statistical Association* 86: 1065—1073.

Marchenko, Y. V. and J. P. Reiter. 2009. Improved degrees of freedom for multivariate significance tests obtained from multiply imputed, small-sample data. *Stata Journal*. Forthcoming.

Rubin, D. B. 1987. *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley.

Rubin, D. B. 1996. Multiple imputation after 18+ years. *Journal of the American Statistical Association* 91: 473—489.

Schafer, J. L. 1997. *Analysis of Incomplete Multivariate Data*. Boca Raton, FL: Chapman & Hall/CRC.

van Buuren, S., H. C. Boshuizen, and D. L. Knook. 1999. Multiple imputation of missing blood pressure covariates in survival analysis.