mi impute monotone — Impute missing values in monotone data

Description | Menu | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | References
Also see

Description

`mi impute monotone` fills in missing values in multiple variables by using a sequence of independent univariate conditional imputation methods. Variables to be imputed, *ivars*, must follow a monotone-missing pattern (see [MI] **Intro substantive**). You can perform separate imputations on different subsets of the data by specifying the `by()` option. You can also account for frequency, analytic (with continuous variables only), importance, and sampling weights.

Menu

Statistics > Multiple imputation

Syntax

Default specification of prediction equations, basic syntax

`mi impute monotone (`*uvmethod*`)` *ivars* [`=` *indepvars*] [*if*] [*weight*] [`,` *impute_options options*]

Default specification of prediction equations, full syntax

`mi impute monotone` *lhs* [`=` *indepvars*] [*if*] [*weight*] [`,` *impute_options options*]

Custom specification of prediction equations

`mi impute monotone` *cmodels* [*if*] [*weight*]`, custom` [*impute_options options*]

where *lhs* is *lhs_spec* [*lhs_spec* [...]] and *lhs_spec* is

`(`*uvmethod* [*if*] [`,` *uvspec_options*]`)` *ivars*

cmodels is `(`*cond_spec*`)` [`(`*cond_spec*`)` [...]] and a conditional specification, *cond_spec*, is

uvmethod ivar [*rhs_spec*] [*if*] [`,` *uvspec_options*]

rhs_spec includes *varlist* and expressions of imputation variables bound in parentheses.

ivar(*s*) (or *newivar* if *uvmethod* is `intreg`) is the name(s) of the imputation variable(s).

uvspec_options are `asc`ontinuous, `noi`sily, and the method-specific *options* as described in the manual entry for each univariate imputation method.

uvmethod	Description
regress	linear regression for a continuous variable; [MI] **mi impute regress**
pmm	predictive mean matching for a continuous variable; [MI] **mi impute pmm**
truncreg	truncated regression for a continuous variable with a restricted range; [MI] **mi impute truncreg**
intreg	interval regression for a continuous partially observed (censored) variable; [MI] **mi impute intreg**
logit	logistic regression for a binary variable; [MI] **mi impute logit**
ologit	ordered logistic regression for an ordinal variable; [MI] **mi impute ologit**
mlogit	multinomial logistic regression for a nominal variable; [MI] **mi impute mlogit**
poisson	Poisson regression for a count variable; [MI] **mi impute poisson**
nbreg	negative binomial regression for an overdispersed count variable; [MI] **mi impute nbreg**

impute_options	Description
Main	
* add(#)	specify number of imputations to add; required when no imputations exist
* replace	replace imputed values in existing imputations
rseed(#)	specify random-number seed
double	store imputed values in double precision; the default is to store them as float
by(*varlist*[, *byopts*])	impute separately on each group formed by *varlist*
Reporting	
dots	display dots as imputations are performed
noisily	display intermediate output
nolegend	suppress all table legends
Advanced	
force	proceed with imputation, even when missing imputed values are encountered
noupdate	do not perform mi update; see [MI] **noupdate option**

*add(#) is required when no imputations exist; add(#) or replace is required if imputations exist.

noupdate does not appear in the dialog box.

options	Description
Main	
* custom	customize prediction equations of conditional specifications
augment	perform augmented regression in the presence of perfect prediction for all categorical imputation variables
bootstrap	estimate model parameters using sampling with replacement
Reporting	
dryrun	show conditional specifications without imputing data
verbose	show conditional specifications and impute data; implied when custom prediction equations are not specified
report	show report about each conditional specification
Advanced	
nomonotonechk	do not check whether variables follow a monotone-missing pattern

*custom is required when specifying customized prediction equations.

You must mi set your data before using mi impute monotone; see [MI] **mi set**.

You must mi register *ivars* as imputed before using mi impute monotone; see [MI] **mi set**.

indepvars and *rhs_spec* may contain factor variables; see **[U] 11.4.3 Factor variables**.

collect is allowed; see **[U] 11.1.10 Prefix commands**.

fweights, aweights (regress, pmm, truncreg, and intreg only), iweights, and pweights are allowed; see **[U] 11.1.6 weight**.

Options

Main

custom is required to build customized prediction equations within the univariate conditional specifications. Otherwise, the default specification of prediction equations is assumed.

add(), replace, rseed(), double, by(); see [MI] **mi impute**.

augment specifies that augmented regression be performed if perfect prediction is detected. By default, an error is issued when perfect prediction is detected. The idea behind the augmented-regression approach is to add a few observations with small weights to the data during estimation to avoid perfect prediction. See *The issue of perfect prediction during imputation of categorical data* under *Remarks and examples* in [MI] **mi impute** for more information. augment is not allowed with importance weights. This option is equivalent to specifying augment within univariate specifications of all categorical imputation methods.

bootstrap specifies that posterior estimates of model parameters be obtained using sampling with replacement; that is, posterior estimates are estimated from a bootstrap sample. The default is to sample the estimates from the posterior distribution of model parameters or from the large-sample normal approximation of the posterior distribution. This option is useful when asymptotic normality of parameter estimates is suspect. This option is equivalent to specifying bootstrap within all univariate specifications.

The following options appear on a Specification dialog that appears when you click on the **Create...** button on the **Main** tab.

uvspec_options are options specified within each univariate imputation method, *uvmethod*. *uvspec_options* include asccontinuous, noisily, and the method-specific *options* as described in the manual entry for each univariate imputation method.

ascontinuous specifies that categorical imputation variables corresponding to the current *uvmethod* be included as continuous in all prediction equations. This option is only allowed when *uvmethod* is logit, ologit, or mlogit.

noisily specifies that the output from the current univariate model fit to the observed data be displayed.

Reporting

dots, noisily, nolegend; see [MI] **mi impute**. noisily specifies that the output from all univariate conditional models fit to the observed data be displayed. nolegend suppresses all imputation table legends which include a legend with the titles of the univariate imputation methods used, a legend about conditional imputation when conditional() is used within univariate specifications, and group legends when by() is specified.

dryrun specifies to show the conditional specifications that would be used to impute each variable without actually imputing data. This option is recommended for checking specifications of conditional models prior to imputation.

verbose specifies to show conditional specifications and impute data. verbose is implied when custom prediction equations are not specified.

report specifies to show a report about each univariate conditional specification. This option, in combination with dryrun, is recommended for checking specifications of conditional models prior to imputation.

Advanced

force; see [MI] **mi impute**.

nomonotonechk specifies not to check that imputation variables follow a monotone-missing pattern. This option may be used to avoid potentially time-consuming checks. The monotonicity check may be time consuming when a large number of variables is being imputed. If you use nomonotonechk with a custom specification, make sure that you list the univariate conditional specifications in the order of monotonicity or you might obtain incorrect results.

The following option is available with mi impute but is not shown in the dialog box:

noupdate; see [MI] **noupdate option**.

Remarks and examples

Remarks are presented under the following headings:

Multivariate imputation when a missing-data pattern is monotone
First use
Using mi impute monotone
Default syntax of mi impute monotone
The alternative syntax of mi impute monotone—custom prediction equations
Examples of using default prediction equations
Examples of using custom prediction equations

See [MI] **mi impute** for a general description and details about options common to all imputation methods, *impute_options*. Also see [MI] **Workflow** for general advice on working with mi.

Multivariate imputation when a missing-data pattern is monotone

When a pattern of missingness in multiple variables is monotone (or, more rigorously, when the missingness-modeling structure is monotone distinct), a multivariate imputation can be replaced with a set of conditional univariate imputations (Rubin 1987, 170–178). Let $X_1, X_2, \ldots, X_p$ be ordered such that if X_{1j} is missing, then X_{2j} is also missing, although X_2 may also be missing in other observations; if X_{2j} is missing, then X_{3j} is missing, although X_3 may also be missing in other observations; and so on. Then a simultaneous imputation of variables $X_1, X_2, \ldots, X_p$ according to a model, $f_{\mathbf{X}}(\cdot)$, and complete predictors (independent variables), $\mathbf{Z}$, is equivalent to the sequential conditional imputation

$$
\begin{aligned}
X_1^\star &\sim f_1(X_1|\mathbf{Z}) \\
X_2^\star &\sim f_2(X_2|X_1^\star, \mathbf{Z}) \\
&\ldots \\
X_p^\star &\sim f_p(X_p|X_1^\star, X_2^\star, \ldots, X_{p-1}^\star, \mathbf{Z})
\end{aligned}
\tag{1}
$$

where for brevity we omit conditioning on the model parameters. The univariate conditional imputation models $f_j(\cdot)$ can each be of a different type (normal, logistic, etc.), as is appropriate for imputing X_j.

The specification of a conditional imputation model $f_j(\cdot)$ includes an imputation method and a prediction equation relating an imputation variable to other explanatory variables. In what follows, we distinguish between the default specification in which the identities of the complete explanatory variables are the same for all imputed variables, and the custom specification in which the identities are allowed to differ.

Under the default specification, prediction equations of each imputation variable include all complete independent variables and all preceding imputation variables that have already been imputed. Under the custom specification, each prediction equation may include a subset of the predictors that would be used under the default specification. The custom specification implies nothing more than the assumption of conditional independence between certain imputation variables and certain sets of predictors.

Model (1) corresponds to the default specification. For example, consider imputation variables X_1, X_2, and X_3, ordered from the most observed to the least observed, and complete predictors Z_1 and Z_2. Under the default specification, the individual prediction equations are determined as follows. The most observed variable, X_1, is predicted from Z_1 and Z_2. The next most observed variable, X_2, is predicted from Z_1, Z_2, and previously imputed X_1. The least observed variable, X_3, is predicted from Z_1, Z_2, and previously imputed X_1 and X_2. (A constant is included in all prediction equations, by default.) We use the following notation to refer to the above sequence of prediction equations (imputation sequence): $X_1|Z_1, Z_2 \rightarrow X_2|X_1, Z_1, Z_2 \rightarrow X_3|X_1, X_2, Z_1, Z_2$.

A sequence such as $X_1|Z_1 \rightarrow X_2|X_1, Z_1, Z_2 \rightarrow X_3|X_1, Z_2$ would correspond to a custom specification. Here X_1 is assumed to be independent of Z_2 given Z_1, and X_3 is assumed to be independent of Z_1 and X_2 given X_1 and Z_2.

The monotone-distinct structure offers much flexibility in building a multivariate imputation model. It simplifies the often intractable multivariate imputation task to a set of simpler univariate imputation tasks. In addition, it accommodates imputation of a mixture of types of variables. So, what's the catch? The catch is that the pattern of missingness is rarely monotone in practice. There are types of data for which a monotone-missing data pattern can occur naturally (for example, follow-up measurements). Usually, however, this happens only by chance.

There are several ways to proceed if your data are not monotone missing. You can discard the observations that violate the monotone-missing pattern, especially if there are very few such observations. You can assume independence among the sets of variables to create independent monotone patterns. For example, the missingness pattern for X_1, X_2, X_3, X_4, X_5 may not be monotone, but it may be for X_1, X_3 and for X_2, X_4, X_5. If it is reasonable to assume independence between these two sets of variables, you can then impute each set separately by using monotone imputation. Other alternatives are to use certain techniques to complete the missing-data pattern to monotone (see, for example, Schafer 1997), to use an iterative sequential (fully conditional) imputation (see [MI] **mi impute chained**; Royston 2005, 2007, 2009; van Buuren, Boshuizen, and Knook 1999; Raghunathan et al. 2001), or to assume an explicit multivariate parametric model for the imputation variables (see [MI] **mi impute mvn**; Schafer 1997). Also see *Multivariate imputation* of [MI] **mi impute** for a general discussion of multivariate imputation.

Throughout this entry, we will assume that the considered imputation variables are monotone missing.

First use

Before we describe various uses of `mi impute monotone`, let's look at an example.

Consider the heart attack data examining the relationship between heart attack and smoking. The `age` and `bmi` variables contain missing values and follow a monotone-missing pattern. Recall multivariate imputation of `bmi` and `age` using `mi impute monotone` described in *Multivariate imputation* of [MI] **mi impute**:

```
. use https://www.stata-press.com/data/r19/mheart5s0
(Fictional heart attack data)
. mi impute monotone (regress) bmi age = attack smokes hsgrad female, add(10)
Conditional models:
               age: regress age attack smokes hsgrad female
               bmi: regress bmi age attack smokes hsgrad female

Multivariate imputation                     Imputations =       10
Monotone method                                   added =       10
Imputed: m=1 through m=10                       updated =        0
               bmi: linear regression
               age: linear regression
```

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

```
(Complete + Incomplete = Total; Imputed is the minimum across m
 of the number of filled-in observations.)
```

The `age` and `bmi` variables have monotone missingness, and so `mi impute monotone` is used to fill in missing values. Ten imputations are created (`add(10)` option). The linear regression imputation method (`regress`) is used to impute both continuous variables. The `attack`, `smokes`, `hsgrad`, and `female` variables are used as complete predictors (independent variables).

The conditional models legend shows that `age` (having the least number of missing values) is imputed first using the `regress` method, even though we specified `bmi` before `age` on the `mi impute` command. After that, `bmi` is imputed using the `regress` method and the previously imputed variable `age` and the other predictors.

The header and table output were described in detail in [MI] **mi impute**. The additional information above the imputation table is the legend describing what univariate imputation method was used to impute each variable. (If desired, this legend may be suppressed by specifying the `nolegend` option.)

Using mi impute monotone

Below we summarize general capabilities of `mi impute monotone`.

1. `mi impute monotone` requires that the specified imputation variables follow a monotone-missing pattern. If they do not, it will stop with an error:

```
. mi impute monotone x1 x2 ...
(1 m=0 obs now marked as incomplete)
x1 x2: not monotone;
    imputation variables must have a monotone-missing
    structure; see mi misstable nested
r(459);
```

   As indicated by the error message, we can use `mi misstable nested` to verify for ourselves that the imputation variables are not monotone missing. We could also use other features of `mi misstable` to investigate the pattern.

2. `mi impute monotone` offers two main syntaxes—one using the default prediction equations,

```
. mi impute monotone ...
```

   and the other allowing customization of prediction equations,

```
. mi impute monotone ..., custom ...
```

   We will refer to the two syntaxes as default and custom, respectively.

3. `mi impute monotone` allows specification of a global (outer) `if` condition,

```
. mi impute monotone ... if exp ...
```

   and equation-specific (inner) `if` conditions,

```
. mi impute monotone ... (... if exp ...) ...
```

   A global `if` is applied to all equations (conditional specifications). You may combine global and equation-specific `if` conditions:

```
. mi impute monotone ... (... if exp ...) ... if exp ...
```

4. `mi impute monotone` allows specification of global weights, which are applied to all equations,

```
. mi impute monotone ... [weight] ...
```

Use a combination of options `dryrun` and `report` to check the specification of each univariate imputation model prior to imputing data.

In the next two sections, we describe the use of `mi impute monotone` first using hypothetical situations and then using real examples.

Default syntax of mi impute monotone

We showed in *First use* an example of `mi impute monotone` with default prediction equations using the heart attack data. Here we provide more details about this default specification.

By default, `mi impute monotone` imputes missing values by using the full specification of prediction equations. It builds the corresponding univariate conditional imputation models based on the supplied information: *uvmethod*, the imputation method; *ivars*, the imputation variables; and *indepvars*, the complete predictors or independent variables.

Suppose that continuous variables `x1`, `x2`, and `x3` contain missing values with a monotone-missing pattern. We want to impute these variables, and we decide to use the same univariate imputation method, say, linear regression, for all. We can do this by typing

```
. mi impute monotone (regress) x1 x2 x3 ...
```

The above corresponds to the first syntax diagram of `mi impute monotone`: *uvmethod* is `regress` and *ivars* is `x1 x2 x3`. Relating the above to the model notation used in (1), f_1, f_2, and f_3 represent linear regression imputation models and the prediction sequence is $X_1 \rightarrow X_2|X_1 \rightarrow X_3|X_2, X_1$.

If we have additional covariates containing no missing values (say, `z1` and `z2`) that we want to include in the imputation model, we can do it by typing

```
. mi impute monotone (regress) x1 x2 x3 = z1 z2 ...
```

Now *indepvars* is `z1 z2` and the prediction sequence is $X_1|Z_1, Z_2 \rightarrow X_2|X_1, Z_1, Z_2 \rightarrow X_3|X_2, X_1, Z_1, Z_2$. Independent variables are included in the prediction equations of all conditional models.

Suppose that we want to use a different imputation method for one of the variables—we want to impute `x3` using predictive mean matching. We can do this by typing

```
. mi impute monotone (regress) x1 x2  (pmm, knn(5)) x3  = z1 z2 ...
```

The above corresponds to the second syntax diagram of `mi impute monotone`, a generalization of the first that accommodates differing imputation methods. The right-hand side of the equation is unchanged. `z1` and `z2` are included in all three prediction equations. The left-hand side now has two specifications: `(regress) x1 x2` and `(pmm, knn(5)) x3`. In previous examples, we had only one left-hand-side specification, *lhs_spec*—`(regress) x1 x2 x3`. (Note that the number of left-hand-side specifications does not necessarily correspond to the number of conditional models; the latter is determined by the number of imputation variables.) In this example, `x1` and `x2` are imputed using linear regression, and `x3` is imputed using predictive mean matching with five nearest neighbors specified in `pmm`'s option `knn()`. All method-specific options must be specified within the parentheses surrounding the method:

```
. mi impute monotone (regress) x1 x2 (pmm, knn(5)) x3 = z1 z2 ...
```

Under the default specification, you can list imputation variables in any order and `mi impute monotone` will determine the correct ordering that follows the monotone-missing pattern.

Suppose now we want to restrict the imputation sample for `x2` to observations where `z1` is one; also see *Imputing on subsamples* of [MI] **mi impute**. The corresponding syntax is

```
. mi impute monotone (regress) x1 (regress if z1==1) x2 (pmm, knn(5))
> x3 = z1 z2 ...
```

If, in addition to the above, we want to impute all variables using an overall subsample where `z3` is one, we can specify the global `if z3==1` condition:

```
. mi impute monotone (regress) x1 (regress if z1==1) x2 (pmm, knn(5))
> x3 = z1 z2 if z3==1 ...
```

When any imputation variable is imputed using a categorical method, `mi impute monotone` automatically includes it as a factor variable in the prediction equations of other imputation variables. Suppose that `x1` is a categorical variable and is imputed using the multinomial logistic method:

```
. mi impute monotone (mlogit) x1 (regress) x2 x3 ...
```

The above will result in the prediction sequence $X_1 \rightarrow X_2|\mathtt{i}.X_1 \rightarrow X_3|X_2, \mathtt{i}.X_1$, where $\mathtt{i}.X_1$ denotes the factors of X_1.

If you wish to include factor variables as continuous in prediction equations, you can use the `ascontinuous` option within a specification of the univariate imputation method for that variable:

```
. mi impute monotone (mlogit, ascontinuous) x1 (regress) x2 x3 ...
```

As we discussed in *The issue of perfect prediction during imputation of categorical data* of [MI] **mi impute**, perfect prediction often occurs during imputation of categorical variables. One way of dealing with it is to use the augmented-regression approach (White, Daniel, and Royston 2010), available through the `augment` option. For example, if perfect prediction occurs during imputation of `x1` in the above, you can specify `augment` within the method specification of `x1` to perform augmented regression:

```
. mi impute monotone (mlogit, augment) x1 (regress) x2 x3 ...
```

Alternatively, you can use the `augment` option with `mi impute monotone` to perform augmented regression for all categorical variables for which the issue of perfect prediction arises:

```
. mi impute monotone (mlogit) x1 (logit) x2 (regress) x3 ..., augment ...
```

The above command is equivalent to specifying `augment` within each specification of a univariate categorical imputation method:

```
. mi impute monotone (mlogit, augment) x1 (logit, augment) x2 (regress) x3 ...
```

Also see *Default prediction equations* in [MI] **mi impute chained** for other uses of the default syntax.

The alternative syntax of mi impute monotone—custom prediction equations

Consider the prediction sequence $X_1 \rightarrow X_2|X_1 \rightarrow X_3|X_2, X_1$. Suppose that we want to predict X_3 from X_1 rather than from X_1 and X_2. This could be achieved by simply imputing X_1 and X_2 and then X_3 given X_1 separately because of the implied assumption that X_3 and X_2 are independent given X_1. However, with a larger number of variables and more complicated prediction rules, separate imputations may not be appealing. So customization of the prediction equations is a good alternative.

You customize prediction equations using the custom syntax (the third syntax) of `mi impute monotone`. You must specify the `custom` option to notify `mi impute monotone` that you are specifying custom prediction equations.

Under the custom syntax, you specify a separate conditional imputation model for each imputation variable. The specification of a conditional model is the same as that for the chosen univariate imputation method, but the entire model must be bound in parentheses, for example,

```
. mi impute monotone (regress x1)
                     (regress x2 x1)
                     (regress x3 x1)
                                 , custom ...
```

Here we have three conditional specifications: `(regress x1)`, `(regress x2 x1)`, and `(regress x3 x1)`. The corresponding prediction sequence is $X_1 \rightarrow X_2|X_1 \rightarrow X_3|X_1$. Prediction equations have the syntax *ivar* [*rhs_spec*].

When specifying custom prediction equations, you are required to list the conditional models in the correct order of missing monotonicity. `mi impute monotone` will issue an error if you are wrong:

```
mi impute monotone: incorrect equation order
    equations must be listed in the monotone-missing order of the imputation
    variables (from most observed to least observed); x2(2) -> x1(5) -> x3(10)
r(198);
```

If we have additional covariates `z1` and `z2` containing no missing values, we can include them in the imputation model:

```
. mi impute monotone (regress x1 z1 z2)
                     (regress x2 x1 z1 z2)
                     (regress x3 x1 z1 z2), custom ...
```

To use the predictive mean matching method for `x3`, we simply change the method from `regress` to `pmm` and specify, say, five nearest neighbors in `pmm`'s required option `knn()` in the last conditional specification:

```
. mi impute monotone (regress x1 z1 z2)
                     (regress x2 x1 z1 z2)
                     (pmm x3 x1 z1 z2, knn(5)), custom ...
```

Under the custom syntax, you can also include expressions of previously imputed variables in prediction equations. For example, if you want to model `x3` using main and squared effects of `x1` (ignoring predictors `z1` and `z2`), you can type

```
. mi impute monotone (regress x1)
                     (regress x2 x1)
                     (pmm x3 x1 (x1^2), knn(5)), custom ...
```

Note that we bound the expression `x1^2` in parentheses. Any expression may appear inside the parentheses.

Similar to the default specification, we can include equation-specific `if`s,

```
. mi impute monotone (regress x1)
                     (regress x2 x1 if z1==1)
                     (pmm x3 x1, knn(5)), custom ...
```

and we can specify a global `if`,

```
. mi impute monotone (regress x1 z1 z2)
                     (regress x2 x1 z2 if z1==1)
                     (pmm x3 x1 z1 z2, knn(5))
                                    if z3==1, custom ...
```

Suppose that one of the imputed variables is categorical. We can use the multinomial logistic method to impute its values:

```
. mi impute monotone (mlogit x1)
                     (regress x2 i.x1)
                     (regress x3 i.x1)
                                 , custom ...
```

Also see *Link between mi impute chained and mi impute monotone* in [MI] **mi impute chained** for a discussion of custom syntaxes.

Examples of using default prediction equations

▷ Example 1: Different imputation methods

Recall the heart attack example from *First use*. If we wanted to impute age using predictive mean matching instead of linear regression, we could type

```
. use https://www.stata-press.com/data/r19/mheart5s0, clear
(Fictional heart attack data)
. mi impute monotone (regress) bmi (pmm, knn(3)) age = attack smokes hsgrad
> female, add(10)
Conditional models:
               age: pmm age attack smokes hsgrad female, knn(3)
               bmi: regress bmi age attack smokes hsgrad female

Multivariate imputation                     Imputations =       10
Monotone method                                added =       10
Imputed: m=1 through m=10                    updated =        0
               bmi: linear regression
               age: predictive mean matching
```

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

```
(Complete + Incomplete = Total; Imputed is the minimum across m
 of the number of filled-in observations.)
```

As previously, we listed age and bmi in the reverse order here, and `mi impute monotone` determined the correct order of missing monotonicity.

◁

▷ Example 2: Imputing a variable on a subsample

Consider an `mi set` version of the heart attack data containing the indicator for smoking high-tar cigarettes (variable `hightar`):

```
. use https://www.stata-press.com/data/r19/mheart6s0, clear
(Fictional heart attack data; bmi, age, and hightar missing)
. mi describe
Style: mlong
       last mi update 04feb2025 12:58:57, 11 days ago
Observations:
   Complete          124
   Incomplete         30  (M = 0 imputations)
   ---------------------
   Total             154
Variables:
   Imputed: 3; bmi(24) age(30) hightar(8)
   Passive: 0
   Regular: 4; attack smokes female hsgrad
   System:  3; _mi_m _mi_id _mi_miss
   (there are no unregistered variables)
```

mi describe reports that there are no imputations, three registered imputed variables (hightar is one of them), and four registered regular variables.

Next we use mi misstable nested to examine missing-data patterns in the data.

```
. mi misstable nested
     1.  hightar(8) -> bmi(24) -> age(30)
```

There is one monotone-missing pattern in the data. According to the output, missing values of hightar are nested within bmi, whose missing values are nested within age. So hightar, bmi, and age follow a monotone-missing pattern.

As before, to impute missing values of age and bmi, we use the regression method. The hightar variable is a binary variable, so we choose the logistic method to fill in its values (see [MI] **mi impute logit**). Because hightar records whether a subject smokes high-tar cigarettes, we use only those who smoke to impute its missing values. (If there were any missing values of hightar for the subjects who do not smoke, we would have replaced them with zeros.)

```
. mi impute monotone (reg) age bmi (logit if smokes) hightar
> = attack smokes hsgrad female, add(10)

Conditional models:
           hightar: logit hightar attack smokes hsgrad female if smokes
               bmi: regress bmi i.hightar attack smokes hsgrad female
               age: regress age bmi i.hightar attack smokes hsgrad female

note: smokes omitted because of collinearity.

Multivariate imputation                     Imputations =       10
Monotone method                                    added =       10
Imputed: m=1 through m=10                        updated =        0

               age: linear regression
               bmi: linear regression
           hightar: logistic regression
```

	Observations per *m*			
Variable	Complete	Incomplete	Imputed	Total
age	124	30	30	154
bmi	130	24	24	154
hightar	56	8	8	64

```
(Complete + Incomplete = Total; Imputed is the minimum across m
 of the number of filled-in observations.)
```

mi impute monotone reports which univariate conditional model was used to impute each variable. Because hightar has the least number of missing observations, it is imputed first using the specified complete predictors and using only observations for smokers. From the output, all incomplete values of each of the variables are imputed in all 10 imputations. Notice that because we restricted the imputation sample of hightar to smokers, the total number of observations reported for hightar is 64 and not 154.

It is safe to use the if restriction in the above because smokes does not contain any missing values and hightar does not contain any missing values in observations with smokes==0. Otherwise, the conditional() option should be used instead; see *Conditional imputation* of [MI] **mi impute** for details.

◁

Examples of using custom prediction equations

▷ Example 3: Using different sets of predictors within individual conditional models

Let's take a closer look at the conditional model for `hightar` used in the above example:

```
hightar: logit hightar attack smokes hsgrad female if (smokes)
```

Notice that predictor `smokes` is redundant in this model because it is collinear with the constant (included in the model by default) on the restricted sample of smokers. In fact, if we specify the `noisily` option (`noi` for short) within the `logit` specification to see the estimation results, we will notice that, as expected, `smokes` was omitted from the estimation model for `hightar`; that is, its coefficient is zero.

```
. mi impute monotone (reg) age bmi (logit if smokes, noi) hightar
> = attack smokes hsgrad female, replace

Conditional models:
           hightar: logit hightar attack smokes hsgrad female if smokes,
                     noisily
               bmi: regress bmi i.hightar attack smokes hsgrad female
               age: regress age bmi i.hightar attack smokes hsgrad female

Running logit on observed data:

note: smokes omitted because of collinearity.
Iteration 0:  Log likelihood = -38.673263
Iteration 1:  Log likelihood = -38.455029
Iteration 2:  Log likelihood = -38.454991
Iteration 3:  Log likelihood = -38.454991

Logistic regression                                     Number of obs =     56
                                                        LR chi2(3)    =   0.44
                                                        Prob > chi2   = 0.9326
Log likelihood = -38.454991                             Pseudo R2     = 0.0056

------------------------------------------------------------------------------
     hightar | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
      attack |   .0773715   .5630513     0.14   0.891    -1.026189    1.180932
      smokes |          0  (omitted)
      hsgrad |  -.1663937   .5977995    -0.28   0.781    -1.338059    1.005272
      female |  -.3331926    .617736    -0.54   0.590    -1.543933    .8775477
       _cons |   .0138334   .6263152     0.02   0.982    -1.213722    1.241389
------------------------------------------------------------------------------

Multivariate imputation                      Imputations =       10
Monotone method                                    added =        0
Imputed: m=1 through m=10                        updated =       10

               age: linear regression
               bmi: linear regression
           hightar: logistic regression

------------------------------------------------------------------------------
                  |                Observations per m                         |
                  |-----------------------------------------------------------|
          Variable |   Complete   Incomplete   Imputed |      Total
-------------------+-----------------------------------+----------------------|
               age |        124           30        30 |        154
               bmi |        130           24        24 |        154
           hightar |         56            8         8 |         64
------------------------------------------------------------------------------
(Complete + Incomplete = Total; Imputed is the minimum across m
 of the number of filled-in observations.)
```

Although mi impute handles collinearity problems for us automatically, we can eliminate redundancy manually by removing smokes from the prediction equation for hightar. To do that, we need to specify custom prediction equations.

As discussed in *Using mi impute monotone*, custom prediction equations are available with mi impute monotone when the custom option is used. We also know that within this custom specification, we must fully specify prediction equations within each conditional model and must specify the conditional models in the monotone-missing order of the imputation variables.

Building such conditional models from scratch could be a tedious task except that we can use mi impute monotone, dryrun to display the conditional models with default prediction equations without performing the corresponding imputation:

```
. mi impute monotone (reg) age bmi (logit if smokes) hightar
> = attack smokes hsgrad female, dryrun

Conditional models:
           hightar: logit hightar attack smokes hsgrad female if smokes
               bmi: regress bmi i.hightar attack smokes hsgrad female
               age: regress age bmi i.hightar attack smokes hsgrad female
```

We can use these default conditional specifications as the basis for writing our own customized specifications. We will remove smokes from the predictor list for hightar:

```
. mi impute monotone (logit hightar attack hsgrad female if smokes)
>                    (regress bmi hightar attack smokes hsgrad female)
>                    (regress age bmi hightar attack smokes hsgrad female)
>                                                              , custom replace

Multivariate imputation                     Imputations =       10
Monotone method                                    added =        0
Imputed: m=1 through m=10                        updated =       10

           hightar: logistic regression
               bmi: linear regression
               age: linear regression
```

	Observations per *m*			
Variable	Complete	Incomplete	Imputed	Total
hightar	56	8	8	64
bmi	130	24	24	154
age	124	30	30	154

(Complete + Incomplete = Total; Imputed is the minimum across *m* of the number of filled-in observations.)

◁

Example 4: Including expressions of imputation variables in prediction equations

The distribution of bmi is slightly skewed. To take this into account, we can either use predictive mean matching to impute bmi or impute bmi on a logarithmic scale. We choose to impute the log of bmi here.

Following the steps described in *Imputing transformations of incomplete variables* of [MI] **mi impute**, we create a new variable, lnbmi, containing the log of bmi and register it as imputed. Here we also reset the number of imputations to zero.

```
. mi set M = 0
(10 imputations dropped; M = 0)
. mi unregister bmi
. generate lnbmi = ln(bmi)
(24 missing values generated)
. mi register imputed lnbmi
```

We are now ready to impute lnbmi. However, although we are imputing the log of bmi, we want to use bmi in the original scale when imputing age. To do that, we include exp(lnbmi) in the prediction equation for age. When including expressions in a custom specification, the expressions must appear in parentheses:

```
. mi impute monotone (logit hightar attack hsgrad female if smokes)
>                    (regress lnbmi hightar attack smokes hsgrad female)
>                    (regress age (exp(lnbmi))
>                        hightar attack smokes hsgrad female)
>                                                              , custom add(10)
Multivariate imputation                     Imputations =       10
Monotone method                                   added =       10
Imputed: m=1 through m=10                       updated =        0
           hightar: logistic regression
             lnbmi: linear regression
               age: linear regression
```

	Observations per *m*			
Variable	Complete	Incomplete	Imputed	Total
hightar	56	8	8	64
lnbmi	130	24	24	154
age	124	30	30	154

```
(Complete + Incomplete = Total; Imputed is the minimum across m
 of the number of filled-in observations.)
```

If we also wanted to include a squared term for bmi in the conditional imputation model for age, we would type

```
. mi impute monotone
> (logit hightar attack hsgrad female if smokes)
> (regress lnbmi hightar attack smokes hsgrad female)
> (regress age (exp(lnbmi)) (exp(lnbmi)^2) hightar attack smokes hsgrad female)
>                                                              , custom replace
```

(*output omitted*)

Stored results

mi impute monotone stores the following in r():

Scalars

r(M)	total number of imputations
r(M_add)	number of added imputations
r(M_update)	number of updated imputations
r(k_ivars)	number of imputed variables
r(N_g)	number of imputed groups (1 if by() is not specified)

Macros

r(method)	name of imputation method (monotone)
r(ivars)	names of imputation variables
r(rngstate)	random-number state used
r(uvmethods)	names of univariate conditional imputation methods
r(by)	names of variables specified within by()

Matrices

r(N)	number of observations in imputation sample in each group (per variable)
r(N_complete)	number of complete observations in imputation sample in each group (per variable)
r(N_incomplete)	number of incomplete observations in imputation sample in each group (per variable)
r(N_imputed)	number of imputed observations in imputation sample in each group (per variable)

Methods and formulas

Let $\mathbf{x}_{(i)} = (x_{i1}, x_{i2}, \ldots, x_{ip})$ be the ith observation containing values of the imputation variables ordered from the most observed to the least observed to form a monotone-missing data pattern. Let $\mathbf{z}_{(i)} = (z_{i1}, z_{i2}, \ldots, z_{iq})$ be the corresponding set of predictors of $\mathbf{x}_{(i)}$. Then, if the missingness-modeling structure is monotone distinct (imputation variables have monotone missingness and parameters of the conditional models are distinct as defined in Rubin [1987, 174]), the following decomposition holds:

$$f_{\mathbf{X}}(\mathbf{x}_{(i)}|\mathbf{z}_{(i)}, \boldsymbol{\theta}) = f_1(x_{i1}|\mathbf{z}_{(i)}, \boldsymbol{\theta}_1) f_2(x_{i2}|\mathbf{z}_{(i)}, x_{i1}, \boldsymbol{\theta}_2) \cdots f_p(x_{ip}|\mathbf{z}_{(i)}, x_{i1}, x_{i2}, \ldots, x_{i,p-1}, \boldsymbol{\theta}_p)$$

where the unknown parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_p$ are distinct, that is, $\Pr(\boldsymbol{\theta}) = \prod_{j=1}^{p} \Pr(\boldsymbol{\theta}_j)$. The monotone-distinct structure ensures that the univariate conditional models f_j do not depend on any unobserved values of variable $\mathbf{x}_j$ and the posterior distributions of $\boldsymbol{\theta}_j$ do not involve the imputed values of the previously filled-in variables $\mathbf{x}_1, \ldots, \mathbf{x}_{j-1}$. See Rubin (1987, 174–178) for a rigorous justification of the above decomposition.

The above allows substituting the imputation of $\mathbf{X}$ using the probability model $f_{\mathbf{X}}(\cdot)$ with a sequence of univariate conditional imputations of $\mathbf{x}_j$ using the probability models $f_j(\cdot)$. Note that f_j can be any proper imputation model (for example, linear regression or logistic regression).

mi impute monotone follows the steps below to fill in missing values in $\mathbf{x}_1, \ldots, \mathbf{x}_p$:

1. If the custom option is not used, mi impute monotone first builds univariate conditional models containing the default prediction equations using the supplied information about imputation methods, imputation variables $\mathbf{X}$, and complete predictors $\mathbf{Z}$. The order in which imputation variables are listed is irrelevant. The prediction equations are constructed as follows. Complete predictors *indepvars* are included first. The imputation variables are included next with each previously imputed variable added to the beginning of the prediction equation previously used.

   If the custom option is used, mi impute monotone uses the specified conditional models in the order supplied. The conditional models must be listed in the monotone-missing order of the corresponding imputation variables.

2. Fit univariate conditional models for each $\mathbf{x}_j$ to the observed data to obtain the estimates of $\boldsymbol{\theta}_j$, $j = 1, \ldots, p$. See step 1 in *Methods and formulas* of each respective univariate imputation method's manual entry for details.
3. Sequentially fill in missing values of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ according to the specified imputation model. See step 2 and step 3 in *Methods and formulas* of each respective univariate imputation method's manual entry for details.
4. Repeat step 3 to obtain M multiple imputations.

References

Raghunathan, T. E., J. M. Lepkowski, J. Van Hoewyk, and P. Solenberger. 2001. A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology* 27: 85–95.

Royston, P. 2005. Multiple imputation of missing values: Update. *Stata Journal* 5: 188–201.

———. 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.

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

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

van Buuren, S., H. C. Boshuizen, and D. L. Knook. 1999. Multiple imputation of missing blood pressure covariates in survival analysis. *Statistics in Medicine* 18: 681–694. https://doi.org/10.1002/(SICI)1097-0258(19990330)18:6<681::AID-SIM71>3.0.CO;2-R.

White, I. R., R. M. Daniel, and P. Royston. 2010. Avoiding bias due to perfect prediction in multiple imputation of incomplete categorical data. *Computational Statistics and Data Analysis* 54: 2267–2275. https://doi.org/10.1016/j.csda.2010.04.005.

Also see

[MI] **mi impute** — Impute missing values

[MI] **mi impute chained** — Impute missing values using chained equations

[MI] **mi impute mvn** — Impute using multivariate normal regression

[MI] **mi estimate** — Estimation using multiple imputations

[MI] **Intro** — Introduction to mi

[MI] **Intro substantive** — Introduction to multiple-imputation analysis

[MI] **Glossary**

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.

For suggested citations, see the FAQ on citing Stata documentation.