Glossary

arbitrary missing pattern. Any missing-value pattern. Some imputation methods are suitable only when the pattern of missing values is special, such as a monotone-missing pattern. An imputation method suitable for use with an arbitrary missing pattern may be used regardless of the pattern.

augmented regression. Regression performed on the augmented data, the data with a few extra observations with small weights. The data are augmented in a way that prevents perfect prediction, which may arise during estimation of categorical data. See *The issue of perfect prediction during imputation of categorical data* under *Remarks and examples* of [MI] **mi impute**.

burn-between period. The number of iterations between two draws of an MCMC sequence such that these draws may be regarded as independent.

burn-in period. The number of iterations it takes for an MCMC sequence to reach stationarity.

casewise deletion. See *listwise deletion*.

chained equations. See *fully conditional specification*.

complete and incomplete observations. An observation in the $m = 0$ data is said to be complete if no imputed variable in the observation contains soft missing (`.`). Observations that are not complete are said to be incomplete.

complete data. Data that do not contain any missing values.

complete degrees of freedom. The degrees of freedom that would have been used for inference if the data were complete.

complete DF. See *complete degrees of freedom*.

complete-cases analysis. See *listwise deletion*.

complete-data analysis. The analysis or estimation performed on the complete data, the data for which all values are observed. This term does not refer to analysis or estimation performed on the subset of complete observations. Do not confuse this with completed-data analysis.

completed data. See *imputed data*.

completed-data analysis. The analysis or estimation performed on the made-to-be completed (imputed) data. This term does not refer to analysis or estimation performed on the subset of complete observations.

conditional imputation. Imputation performed using a conditional sample, a restricted part of the sample. Missing values outside the conditional sample are replaced with a conditional constant, the constant value of the imputed variable in the nonmissing observations outside the conditional sample. See *Conditional imputation* under *Remarks and examples* of [MI] **mi impute**.

DA. See *data augmentation*.

data augmentation. An MCMC method used for the imputation of missing data.

EM. See *expectation-maximization algorithm*.

expectation-maximization algorithm. In the context of MI, an iterative procedure for obtaining maximum likelihood or posterior-mode estimates in the presence of missing data.

FCS. See *fully conditional specification*.

flong data. See *style*.

flongsep data. See *style*.

FMI. See *fraction of missing information*.

fraction of missing information. The ratio of information lost due to the missing data to the total information that would be present if there were no missing data.

An equal FMI test is a test under the assumption that FMIs are equal across parameters.

An unrestricted FMI test is a test without the equal FMI assumption.

fully conditional specification. Consider imputation variables $X_1, X_2, \ldots, X_p$. Fully conditional specification of the prediction equation for X_j includes all variables except X_j; that is, variables $\mathbf{X}_{-j} = (X_1, X_2, \ldots, X_{j-1}, X_{j+1}, \ldots, X_p)$.

hard missing and soft missing. A hard missing value is a value of `.a`, `.b`, ..., `.z` in $m = 0$ in an imputed variable. Hard missing values are not replaced in $m > 0$.

A soft missing value is a value of `.` in $m = 0$ in an imputed variable. If an imputed variable contains soft missing, then that value is eligible to be imputed, and perhaps is imputed, in $m > 0$.

Although you can use the terms hard missing and soft missing for passive, regular, and unregistered variables, it has no special significance in terms of how the missing values are treated.

ignorable missing-data mechanism. The missing-data mechanism is said to be ignorable if missing data are missing at random and the parameters of the data model and the parameters of the missing-data mechanism are distinct; that is, the joint distribution of the model and the missing-data parameters can be factorized into two independent marginal distributions of model parameters and of missing-data parameters.

imputed, passive, and regular variables. An imputed variable is a variable that has missing values and for which you have or will have imputations.

A passive variable is a varying variable that is a function of imputed variables or of other passive variables. A passive variable will have missing values in $m = 0$ and varying values for observations in $m > 0$.

A regular variable is a variable that is neither imputed nor passive and that has the same values, whether missing or not, in all m.

Imputed, passive, and regular variables can be registered using the `mi register` command; see [MI] **mi set**. You are required to register imputed variables, and we recommend that you register passive variables. Regular variables can also be registered. See *registered and unregistered variables*.

The names of imputation and passive variables may not exceed 29 characters. In the wide style, the names of these variables may be restricted to less than 29 characters depending on the number of imputations. In the flongsep style, the names of regular variables in addition to the names of imputation and passive variables also may not exceed 29 characters.

imputed data. Data in which all missing values are imputed.

incomplete observations. See *complete and incomplete observations*.

ineligible missing value. An ineligible missing value is a missing value in a to-be-imputed variable that is due to inability to calculate a result rather than an underlying value being unobserved. For instance, assume that variable `income` had some missing values and so you wish to impute it. Because `income` is skewed, you decide to impute the log of income, and you begin by typing

```
. generate lnincome = log(income)
```

If `income` contained any zero values, the corresponding missing values in `lnincome` would be ineligible missing values. To ensure that values are subsequently imputed correctly, it is of vital importance that any ineligible missing values be recorded as hard missing. You would do that by typing

```
. replace lnincome = .a if lnincome==. & income!=.
```

As an aside, if after imputing `lnincome` using `mi impute` (see [MI] **mi impute**), you wanted to fill in `income`, `income` surprisingly would be a passive variable because `lnincome` is the imputed variable and `income` would be derived from it. You would type

```
. mi register passive income
. mi passive: replace income = cond(lnincome==.a, 0, exp(lnincome))
```

In general, you should avoid using transformations that produce ineligible missing values to avoid the loss of information contained in other variables in the corresponding observations. For example, in the above, for zero values of `income` we could have assigned the log of income, `lnincome`, to be the smallest value that can be stored as `double`, because the logarithm of zero is negative infinity:

```
. generate lnincome = cond(income==0, mindouble(), log(income))
```

This way, all observations for which `income==0` will be used in the imputation model for `lnincome`.

jackknifed standard error. See *Monte Carlo error*.

listwise deletion, **casewise deletion**. Omitting from analysis observations containing missing values.

M, **m**. M is the number of imputations. m refers to a particular imputation, $m = 1, 2, \dots, M$. In `mi`, $m = 0$ is used to refer to the original data, the data containing the missing values. Thus `mi` data in effect contain $M + 1$ datasets, corresponding to $m = 0$, $m = 1$, $\dots$, and $m = M$.

MAR. See *missing at random*.

Markov chain Monte Carlo. A class of methods for simulating random draws from otherwise intractable multivariate distributions. The Markov chain has the desired distribution as its equilibrium distribution.

MCAR. See *missing completely at random*.

MCE. See *Monte Carlo error*.

MCMC. See *Markov chain Monte Carlo*.

mi data. Any data that have been `mi set` (see [MI] **mi set**), whether directly by `mi set` or indirectly by `mi import` (see [MI] **mi import**). The `mi` data might have no imputations (have $M = 0$) and no imputed variables, at least yet, or they might have $M > 0$ and no imputed variables, or vice versa. An `mi` dataset might have $M > 0$ and imputed variables, but the missing values have not yet been replaced with imputed values. Or `mi` data might have $M > 0$ and imputed variables and the missing values of the imputed variables filled in with imputed values.

missing at random. Missing data are said to be missing at random (MAR) if the probability that data are missing does not depend on unobserved data but may depend on observed data. Under MAR, the missing-data values do not contain any additional information given observed data about the missing-data mechanism. Thus the process that causes missing data can be ignored.

missing completely at random. Missing data are said to be missing completely at random (MCAR) if the probability that data are missing does not depend on observed or unobserved data. Under MCAR, the missing data values are a simple random sample of all data values, so any analysis that discards the missing values remains consistent, albeit perhaps inefficient.

missing not at random. Missing data are missing not at random (MNAR) if the probability that data are missing depends on unobserved data. Under MNAR, a missing-data mechanism (the process that causes missing data) must be modeled to obtain valid results.

mlong data. See *style*.

MNAR. See *missing not at random*.

monotone-missing pattern, **monotone missingness**. A special pattern of missing values in which if the variables are ordered from least to most missing, then all observations of a variable contain missing in the observations in which the prior variable contains missing.

Monte Carlo error. Within the multiple-imputation context, a Monte Carlo error is defined as the standard deviation of the multiple-imputation results across repeated runs of the same imputation procedure using the same data. The Monte Carlo error is useful for evaluating the statistical reproducibility of multiple-imputation results. See *Example 6: Monte Carlo error estimates* under *Remarks and examples* of [MI] **mi estimate**.

original data. Original data are the data as originally collected, with missing values in place. In `mi` data, the original data are stored in $m = 0$. The original data can be extracted from `mi` data by using `mi extract`; see [MI] **mi extract**.

passive variable. See *imputed, passive, and regular variables*.

registered and unregistered variables. Variables in `mi` data can be registered as imputed, passive, or regular by using the `mi register` command; see [MI] **mi set**.

You are required to register imputed variables.

You should register passive variables; if your data are style wide, you are required to register them. The `mi passive` command (see [MI] **mi passive**) makes creating passive variables easy, and it automatically registers them for you.

Whether you register regular variables is up to you. Registering them is safer in all styles except wide, where it does not matter. By definition, regular variables should be the same across m. In the long styles, you can unintentionally create variables that vary. If the variable is registered, `mi` will detect and fix your mistakes.

Super-varying variables, which rarely occur and can be stored only in flong and flongsep data, should never be registered.

The registration status of variables is listed by the `mi describe` command; see [MI] **mi describe**.

regular variable. See *imputed, passive, and regular variables*.

relative efficiency. Ratio of variance of a parameter given estimation with finite M to the variance if M were infinite.

relative variance increase. The increase in variance of a parameter estimate due to nonresponse.

RVI. See *relative variance increase*.

style. Style refers to the format in which the `mi` data are stored. There are four styles: flongsep, flong, mlong, and wide. You can ignore styles, except for making an original selection, because all `mi` commands work regardless of style. You will be able to work more efficiently, however, if you understand the details of the style you are using; see [MI] **styles**. Some tasks are easier in one style than another. You can switch between styles by using the `mi convert` command; see [MI] **mi convert**.

The flongsep style is best avoided unless your data are too big to fit into one of the other styles. In flongsep style, a separate `.dta` set is created for $m = 0$, for $m = 1, \ldots,$ and for $m = M$. Flongsep is best avoided because `mi` commands work more slowly with it.

In all the other styles, the $M + 1$ datasets are stored in one `.dta` file. The other styles are both more convenient and more efficient.

The most easily described of these `.dta` styles is flong; however, flong is also best avoided because mlong style is every bit as convenient as flong, and mlong is memorywise more efficient. In flong, each observation in the original data is repeated M times in the `.dta` dataset, once for $m = 1$, again for $m = 2$, and so on. Variable `_mi_m` records m and takes on values 0, 1, 2, ..., M. Within each value of m, variable `_mi_id` takes on values 1, 2, ..., N and thus connects imputed with original observations.

The mlong style is recommended. It is efficient and easy to use. Mlong is much like flong except that complete observations are not repeated.

Equally recommended is the wide style. In wide, each imputed and passive variable has an additional M variables associated with it, one for the variable's value in $m = 1$, another for its value in $m = 2$, and so on. If an imputed or passive variable is named *vn*, then the values of *vn* in $m = 1$ are stored in variable `_1_`*vn*; the values for $m = 2$, in `_2_`*vn*; and so on.

What makes mlong and wide so convenient? In mlong, there is a one-to-one correspondence of your idea of a variable and Stata's idea of a variable—variable *vn* refers to *vn* for all values of m. In wide, there is a one-to-one correspondence of your idea of an observation and Stata's idea—physical observation 5 is observation 5 in all datasets.

Choose the style that matches the problem at hand. If you want to create new variables or modify existing ones, choose mlong. If you want to drop observations or create new ones, choose wide. You can switch styles with the `mi convert` command; see [MI] **mi convert**.

For instance, if you want to create new variable `ageXexp` equal to `age*exp` and your data are mlong, you can just type `generate ageXexp = age*exp`, and that will work even if `age` and `exp` are imputed, passive, or a mix. Theoretically, the right way to do that is to type `mi passive: generate agexExp = age*exp`, but concerning variables, if your data are mlong, you can work the usual Stata way.

If you want to drop observation 20 or drop `if sex==2`, if your data are wide, you can just type `drop in 20` or `drop if sex==2`. Here the "right" way to do the problem is to type the `drop` command and then remember to type `mi update` so that `mi` can perform whatever machinations are required to carry out the change throughout $m > 0$; however, in the wide form, there are no machinations required.

super-varying variables. See *varying and super-varying variables*.

unregistered variables. See *registered and unregistered variables*.

varying and super-varying variables. A variable is said to be varying if its values in the incomplete observations differ across m. Imputed and passive variables are varying. Regular variables are nonvarying. Unregistered variables can be either.

Imputed variables are supposed to vary because their incomplete values are filled in with different imputed values, although an imputed variable can be temporarily nonvarying if you have not imputed its values yet. Similarly, passive variables should vary because they are or will be filled in based on values of varying imputed variables.

A variable is said to be super varying if its values in the complete observations differ across m. The existence of super-varying variables is usually an indication of error. It makes no sense for a variable to have different values in, say, $m = 0$ and $m = 2$ in the complete observations—in observations that contain no missing values. That is, it makes no sense unless the values of the variable is a function of the values of other variables across multiple observations. If variable `sumx` is the sum of x across observations, and if x is imputed, then sumx will differ across m in all observations after the first observation in which x is imputed.

The mi varying command will identify varying and super-varying variables, as well as nonvarying imputed and passive variables. [MI] **mi varying** explains how to fix problems when they are due to error.

Some problems that theoretically could arise cannot arise because mi will not let them. For instance, an imputed variable could be super varying and that would obviously be a serious error. Or a regular variable could be varying and that, too, would be a serious error. When you register a variable, mi fixes any such problems and, from that point on, watches for problems and fixes them as they arise.

Use mi register to register variables; see [MI] **mi set**. You can perform the checks and fixes at any time by running mi update; see [MI] **mi update**. Among other things, mi update replaces values of regular variables in $m > 0$ with their values from $m = 0$; it replaces values of imputed variables in $m > 0$ with their nonmissing values from $m = 0$; and it replaces values of passive variables in incomplete observations of $m > 0$ with their $m = 0$ values. mi update follows a hands-off policy with respect to unregistered variables.

If you need super-varying variables, use flong or flongsep style and do not register the variable. You must use one of the flong styles because in the wide and mlong styles, there is simply no place to store super-varying values.

wide data. See *style*.

WLF. See *worst linear function*.

worst linear function. A linear combination of all parameters being estimated by an iterative procedure that is thought to converge slowly.

Also see

[MI] **intro** — Introduction to mi