The Stata Journal (2005)
5, Number 2, pp. 1–14

Multiple imputation of missing values: update

Patrick Royston
Cancer Division
MRC Clinical Trials Unit
222 Euston Road
London NW1 2DA UK

Abstract. This article describes a substantial update to `mvis`, which brings it more closely in line with the feature set of S. van Buuren and C. G. M. Oudshoorn's implementation of the MICE system in R and S-PLUS (for details, see *http://www.multiple-imputation.com*). To make a clear distinction from `mvis`, the principal program of the new Stata release is called `ice`. I will give details of how to use the new features and a practical illustrative example using real data. All the facilities of `mvis` are retained by `ice`. Some improvements to `micombine` for computing estimates from multiply imputed datasets are also described.

Keywords: st0067_1, ice, mvis, uvis, micombine, mijoin, misplit, missing data, missing at random, multiple imputation, multivariate imputation, regression modeling

1 Introduction

Royston (2004) introduced `mvis`, an implementation for Stata of a method of multiple multivariate imputation of missing values under missing-at-random (MAR) assumptions. The method is known as MICE, an acronym for multiple imputation by chained equations (van Buuren et al. 1999). See van Buuren's article for details of the theory behind MICE and his interesting web site *http://www.multiple-imputation.com* for abundant literature references, reports, information, and software links. The material will not be repeated here.

Royston (2004) presented five ado-files: `mvis` to create multiple multivariate imputations; `uvis` to impute missing values of a single variable as a function of several covariates, each with complete data; `micombine` to fit a wide variety of regression models to a multiply imputed dataset, combining the estimates using Rubin's rules (1987); and `misplit` and `mijoin`, utilities to inter-convert datasets created by `mvis` and by Carlin et al.'s `miset` routine (2003).

In this article, I will describe a substantial update to `mvis`, which brings it more closely in line with the feature set of S. van Buuren and C. G. M. Oudshoorn's implementation of MICE in R and S-PLUS (for details, see *http://www.multiple-imputation.com*). To avoid confusion with `mvis`, the principal program of the new Stata release is called `ice`. I will give details of how to use the new features and a practical illustrative example using real data. All the facilities of `mvis` are retained by `ice`. Some improvements to `micombine` are also described.

2 Syntax

`ice` *mainvarlist* `using` *filename*[`.dta`] [`if` *exp*] [`in` *range*] [*weight*] [`,`
`boot`[`(`*varlist*`)`] `cc(`*ccvarlist*`)` `cmd(`*cmdlist*`)` `cycles(`*#*`)` `draw`[`(`*varlist*`)`]
`dryrun` `eq(`*eqlist*`)` `genmiss(`*string*`)` `id(`*string*`)` `m(`*#*`)` `on(`*varlist*`)` `noconstant`
`noshoweq` `passive(`*passivelist*`)` `substitute(`*sublist*`)` `replace` `seed(`*#*`)`]

`uvis` *regression_cmd yvar xvarlist* [`if` *exp*] [`in` *range*] [*weight*]`,`
`gen(`*newvarname*`)` [`noconstant` `boot` `draw` `replace` `seed(`*#*`)`]

where *regression_cmd* may be `logistic`, `logit`, `mlogit`, `ologit`, or `regress`. All weight types supported by *regression_cmd* are allowed.

`micombine` *regression_cmd* [*yvar*] [*covarlist*] [`if` *exp*] [`in` *range*] [*weight*] [`,`
`br` `noconstant` `detail` `eform(`*string*`)` `genxb(`*newvarname*`)` `impid(`*varname*`)`
`lrr` `obsid(`*varname*`)` *regression_cmd_options*]

where *regression_cmd* may be `clogit`, `cnreg`, `glm`, `logistic`, `logit`, `poisson`, `probit`, `qreg`, `regress`, `rreg`, `xtgee`, `streg`, `stcox`, `ologit`, `oprobit`, or `mlogit`. All weight types supported by *regression_cmd* are allowed.

`mijoin, clear` [`m(`*#*`)` `impid(`*varname*`)`]

`misplit, clear` [`m(`*#*`)` `impid(`*varname*`)`]

The options are described in the help files. Features new to `mvis` and now in `ice` are discussed in some detail in the next section. New features of `micombine` are discussed in the section *Changes to micombine*.

3 Options

I shall give details here only of options that are new or modified from the previous release of the suite of programs.

3.1 New options for ice

`dryrun` does a "dry run"; that is, `ice` reports the prediction equations it has constructed from the various inputs. No imputation is done, and no files are created. It is not mandatory to specify an output file with `using` for a dry run. The prediction equation setup needs to be carefully checked before running what may be a lengthy imputation process.

`eq(`*eqlist*`)` allows customization of prediction equations for any subset of variables in *mainvarlist*. The `eq()` option, particularly when used with `passive()`, allows great flexibility in the possible imputation schemes. The syntax of *eqlist* is *varname1*`:`*varlist1* [`,` *varname2*`:`*varlist2* ...] where each *varname#* (or *varlist#*) is a member or subset of *mainvarlist*. It is your responsibility to ensure that each equation is sensible. `ice` places no restrictions except to check that all variables mentioned are indeed in *mainvarlist*, and that an equation is not defined for a variable specified to be passively imputed (see the `passive()` option). Note that `eq()` takes precedence over all default definitions and assumptions about the way a given variable in *mainvarlist* will be imputed. The default, if the `passive()` and `substitute()` options are not invoked, is that each variable in *mainvarlist* with any missing data is imputed from all the other variables in *mainvarlist*.

`noshoweq` suppresses the presentation of the prediction equations.

`passive(`*passivelist*`)` allows the use of "passive" imputation of variables that depend on other variables, some of which are imputed. The syntax of *passivelist* is *varname*`:`*exp* [*varname*`:`*exp* ...]. Here *exp* denotes a valid Stata expression (see `help exp` in Stata). Notice the requirement to use '\' as a separator between items in *passivelist*, rather than the usual comma; the reason is that a comma may be a valid part of an expression. The option is most easily explained by example. Suppose that `x1` is a categorical variable with 3 levels, and that two dummy variables `x1a` and `x1b` have been created by the commands

```
. generate byte x1a=(x1==2)
. generate byte x1b=(x1==3)
```

Now suppose that `x1` is to be imputed by the `mlogit` command and is to be treated as the two dummy variables `x1a` and `x1b` when predicting other variables. Use of `mlogit` is achieved by the option `cmd(x1:mlogit)`. When `x1` is imputed, we want `x1a` and `x1b` to be updated with new values that depend on the imputed values of `x1`. This may be achieved by specifying `passive(x1a:x1==2\x1b:x1==3)`. It is necessary also to remove `x1` from the list of predictors when variables other than `x1` are being imputed, and this is done by using the `substitute()` option; in the present example, you would specify `substitute(x1:x1a x1b)`. In this example, the `generate` statements given above will make `x1a` take the (possibly unintended) value of 0 when `x1` is missing. However, `ice` is careful to ensure that `x1a` and `x1b` inherit the missingness of `x1` and are passively imputed following active imputation of missing values of `x1`. If this were not done, incorrect results would occur. The only responsibility of the user is to make sure that `x1a` and `x1b` exist before running `ice`. The values they assume initially are irrelevant since `ice` recalculates them anyway.

A second example is multiplicative interactions between variables, say, between `x1` and `x2` (e.g., `x12=x1*x2`); this could be implemented through `passive(x12:x1*x2)`. It would cause the interaction term `x12` to be omitted when either `x1` or `x2` was being imputed since it would make no sense to impute `x1` from its interaction with `x2`. The `substitute()` option is not needed here. It should be stressed that variables to be imputed passively must be included in *mainvarlist*; otherwise, they will not be recognized.

`substitute(`*sublist*`)` is typically used with the `passive()` option to represent multilevel categorical variables as dummy variables in models for predicting other variables. See `passive()` for more details. The syntax of *sublist* is *varname*`:`*dummyvarlist* [`,`*varname*`:`*dummyvarlist* ...], where *varname* is the name of a variable to be substituted and *dummyvarlist* is the list of dummy variables representing it.

3.2 Options for uvis, mijoin, misplit, micombine

None of the options for `uvis`, `mijoin`, `misplit` have been changed. Details of some changes to `micombine` are given in the section *Changes to micombine*.

4 What is new?

The changes to `mvis` implemented in `ice` mainly affect two areas:

1. The flexibility of the prediction equations used in the system of chained equations and
2. The way the program handles categorical variables, interactions, and transformations.

4.1 Flexible prediction equations

The earlier program `mvis` is restricted to imputing missing observations for each variable in *mainvarlist* from all the other members of *mainvarlist*. While this is sensible in many applications, there are many other cases in which greater flexibility is needed. I will give a relatively simple example in the section *Example: Fetal size*. With the `eq()` option of `ice`, an equation (i.e., a list of right-hand-side variables, comprising part of a model) may be specified in order to impute any variable in *mainvarlist* from any subset of *mainvarlist*. The `eq()` option overrides all automatic behavior invoked by the `passive()` and `substitute()` options (described in context below), giving the user ultimate control over the prediction equations.

4.2 Categorical variables

`mvis` does not handle categorical variables in an ideal fashion. For example, suppose that `x1` is continuous and `x2` is a three-level unordered categorical variable taking the values 0, 1, 2, both variables having missing data. It would be preferable to impute `x1` by linear regression on dummy variables `x21` and `x22` indicating values of 1 and 2, respectively, of `x2`. A natural way to impute `x2` is by multinomial logistic regression (`mlogit` command) of `x2` on `x1`. A possible way to achieve this with `mvis` may be thought to be

```
. generate byte x21=(x2==1) if x2<.
. generate byte x22=(x2==2) if x2<.
. mvis x1 x2 using temp, m(5) cmd(x2:mlogit)
```

However, although appropriate for predicting `x2`, this approach predicts `x1` from `x2` but not from the dummy variables `x21` and `x22`, which are not used. An alternative scheme with dummy variables for `x2`,

```
. mvis x1 x21 x22 using temp, m(5) cmd(x21 x22:logit)
```

does not allow constrained imputation of `x2` (as would be achieved by using `mlogit`). The result could be inconsistent estimates of the missing values of `x2` when the latter is reconstucted from the original and imputed values of `x21` and `x22`. For example, suppose that `x2` = 2 in a particular observation. Then `x21` = 0 and `x22` = 1. Suppose that `x2` were missing for this observation and that `x21` and `x22` were imputed independently, as in the command above. It could happen by chance that `x21` = 1 and `x22` = 1, which do not encode a possible value of `x2`. An appropriate solution with `ice` is

```
. ice x1 x2 x12 x22 using temp, m(5) cmd(x2:mlogit)
    passive(x21:x2==1\x22:x2==2) substitute(x2:x21 x22)
```

The logic here is worth spelling out. *mainvarlist* must include all variables that are involved in the imputation; hence `x2` and both its dummy variables are listed. However, we do not wish to predict `x1` from `x2 x12 x22`, but only from `x12 x22`. This is achieved by the option `substitute(x2:x21 x22)`, which replaces `x2` with `x21 x22` whenever `x2` is a predictor and removes redundant mentions of variables already included. Finally, it is necessary to construct imputed values of `x21` and `x22` from imputed values of `x2`, the latter being obtained internally via the model `mlogit x2 x1`. This is done by the option `passive(x21:x2==1\x22:x2==2)`, which recalculates the dummy variables from the newly imputed values of `x2`.

Although it may at first sight appear complicated, the approach is consistent. The options `passive()`, `substitute()`, and `eq()` together cover all practical possibilities with categorical variables in a unified syntax. More generally, passive imputation (a term coined, I believe, by Steff van Buuren) is an important feature of the MICE system for dealing flexibly with categorical covariates, interactions, transformations and the like. The `substitute()` option is mainly a convenience feature, since substitution could be achieved explicitly by use of the `eq()` option, for example, `eq(x1:x21 x22)` instead of `substitute(x2:x21 x22)`.

4.3 Interactions

Correct imputation involving multiplicative interactions requires the `passive()` option. Suppose that we plan to impute continuous variables `x1` and `x2` and a binary variable `z`. Suppose also that there are good reasons for imputing `x2` from the main effects and interaction of `x1` with `z`, that is from `x1`, `z`, and a new variable `x1z` = `x1*z`. How do we impute `x1` and `z` in this situation? If we were to specify

```
. generate x1z=x1*z
. ice x1 x2 z x1z using temp, m(5)
```

we would get `x2` imputed from `x1`, `z`, and `x1z` (as required); `x1` from `x2`, `z`, and `x1z`; and `z` from `x1`, `x2`, and `x1z`. However, it makes no sense to impute a variable from its interaction with another variable. We wish to eliminate the interaction term `x1z` from the prediction equations for `x1` and `z`. Even if the problem were tackled by using the `eq()` option, because of the danger of inconsistency we do not want the imputation of `x1z` to be treated independently of the imputation of its components, `x1` and `z`. The way to solve the problem is to use the `passive()` option. Note also the `dryrun` option that allows us to see and check the prediction equations but not yet to run the imputations:

```
. ice x1 x2 z x1z using temp, m(5) passive(x1z:x1*z) dryrun
    Variable | Command      | Prediction equation
-------------+--------------+----------------------------------------------
             |              |
          x1 |              | [No missing data in estimation sample]
          x2 | regress      | x1 z x1z
           z | mlogit       | x1 x2
         x1z |              | [Passively imputed from x1*z]
End of dry run. No imputations were done, no files were created.
```

As seen above, `ice` automatically removes the composite passive variable `x1z` from the relevant prediction equations. When it discerns that `x1z` is computed from `x1` and `z`, it *knows* to exclude `x1z` from the prediction equations for imputing `x1` and `z`. This would happen with any variable that is a combination of others.

4.4 Transformations

Simple or complex transformations may be handled by combining the `passive()` and `eq()` options. Here is a fairly complicated but realistic example, which prefigures the example presented in detail in the section *Example: Fetal size*. Suppose that we have three continuous variables `y1`, `y2`, and `x`, all with missing values. Preliminary analysis indicates that the following prediction equations are appropriate:

$$\begin{aligned} E\left(y_1\right) &= \beta_1 x^{-2} + \beta_2 x^{-1} \\ E\left(y_2\right) &= \gamma_1 x + \gamma_2 x^2 \\ E\left(x\right) &= \delta_1 y_1 + \delta_2 y_2 \end{aligned}$$

In other words, in expectation, `y1` is a fractional polynomial function of `x` with powers $(-2, -1)$, `y2` is a quadratic function of `x`, and `x` is a linear function of `y1` and `y2`. You may imagine `y1` and `y2` to be two response variables linked to a common covariate, `x`. With this imputation scheme, missing values may be imputed in `ice` as follows:

```
. generate xa=x^-2
. generate xb=x^-1
. generate xc=x^2
. ice y1 y2 x xa xb xc using temp, m(5) passive(xa:x^-2\xb:x^-1\xc:x^2)
> eq(y1:xa xb, y2:x xc, x:y1 y2)
    Variable | Command      | Prediction equation
------------+--------------+-------------------------------------------
            |              |
          y1 | regress      | xa xb
          y2 | regress      | x xc
           x | regress      | y1 y2
          xa |              | [Passively imputed from x^-2]
          xb |              | [Passively imputed from x^-1]
          xc |              | [Passively imputed from x^2]
```

Here the equation for each variable must be defined explicitly, and this is easily done by using the `eq()` option.

4.5 More on passively imputed variables

Passive (passively imputed) variables in general are computed directly as a function of one or more active (actively imputed) variables. They inherit their missing values from their component active variables. An active variable is a member of *mainvarlist* that is not defined as passive and that has at least one missing value in the estimation sample. Finally, there is a third class of fully observed variables that have no missing values within the estimation sample defined by `if`, `in`, and weights.

It is sometimes convenient to create passive variables as a function of other passive (and perhaps also of active and fully observed) variables. This will work fine provided that the chain of computation of passive variables is correctly reflected in the order of the variables presented in *mainvarlist*. For example, the command

```
. ice y x z xa xaz using temp, m(5) passive(xa:x^-2\xaz:xa*z)
    eq(y:xa z xaz, x:y z, z:y xa)
```

will compute `xa=x^-2` followed by `xaz=xa*z`, which is appropriate since `x` and `z` precede `xa` and `xa` precedes `xaz` in *mainvarlist*. On the other hand, on reversing `xa` and `xaz` in *mainvarlist*, namely

```
. ice y x z xaz xa using temp, m(5) passive(xa:x^-2\xaz:xa*z)
    eq(y:xa z xaz, x:y z, z:y xa)
```

will cause `ice` to recalculate `xaz` before `xa` has been updated from `x`. This will give incorrect results. The rule is simple: structure *mainvarlist* such that primary passive variables (that is, variables that are derived only from active variables) precede secondary passive variables (which depend on primary passive variables) and the latter precede tertiary passive variables (which involve secondary passive variables), and so on. The order of definition of passive variables in the `passive()` option is immaterial.

Note that secondary and higher-order passive variables are not traced back to their original active variables and are therefore not automatically removed from the prediction equations of the relevant active variables. Consider the example

```
. ice x1 x2 z z1 z2 x2z1 x2z2 using temp, replace m(5) substitute(z:z1 z2)
> passive(z1:z==1\z2:z==2\x2z1:x2*z1\x2z2:x2*z2)
   Variable | Command      | Prediction equation
------------+--------------+-------------------------------------------------
         x1 | regress      | x2 z1 z2 x2z1 x2z2
         x2 | regress      | x1 z1 z2
          z | mlogit       | x1 x2 x2z1 x2z2
         z1 |              | [Passively imputed from z==1]
         z2 |              | [Passively imputed from z==2]
       x2z1 |              | [Passively imputed from x2*z1]
       x2z2 |              | [Passively imputed from x2*z2]
```

We see that `z` is inappropriately going to be predicted from `x1`, `x2`, and `z`'s interaction with `x2`, expressed indirectly through the interaction between `x2` and `z1`, `z2`. Here, `x2z1` and `x2z2` are defined as secondary passive variables. To correct the problem, you could either specify the equation for `z` directly, e.g., `eq(z:x1 x2)` or define only primary passive variables, as in `passive(z1:z==1\z2:z==2\x2z1:x2*(z==1)\x2z2:x2*(z==2))`. The second solution is perhaps cleaner, though more laborious.

5 Example: Fetal size

5.1 Data

I will present an example that is rather different from the usual sort of imputation problem encountered in clinical biostatistics and epidemiology. Some 15 years ago, Altman and Chitty (1993) designed and performed an influential study of the growth of the fetus. The aim was to establish gestational age-specific reference intervals ("normal ranges") for each of a large number of in-utero "parameters" (anthropometric measurements) of fetal size. In each of 649 singleton pregnancies, ultrasound scanning of the abdomen was carried out once on each mother according to a predefined study schedule. The intention was adequately to cover the important gestational period between 12 and 42 weeks, and this was largely achieved. The dataset is therefore cross-sectional in character. A longitudinal study of growth was also performed, but I will not consider that here.

In this example, I will consider a subset of the study dataset comprising just 4 variables: gestational age (`ga`) in weeks (measured to the nearest day), and three fetal-size measurements: abdominal circumference (`ac`), head circumference (`hc`), and mandible (jawbone) length (`ml`). The pairwise rank correlations and relative sample sizes are shown in table 1.

The data for `ac` and `hc` are $>$ 90% complete, whereas only about a quarter of fetuses had mandible measurements. The clinical reason is that it is difficult to make accurate measurements of this structure. Values taken after 28 weeks' gestation are regarded by ultrasonographers as unreliable and have been discarded. Note the high rank correlations among all four variables.

Table 1: Rank correlations among variables in the fetal-size dataset ($n = 649$), based on available pairs of nonmissing observations. Values in parentheses are the percentage of complete observations for the variable or pair of variables in question. Only `ga` had complete data

Variable	`ac`	`hc`	`ml`	`ga`
`ac`	1.000 (94%)			
`hc`	0.991 (86%)	1.000 (92%)		
`ml`	0.948 (25%)	0.951 (25%)	1.000 (26%)	
`ga`	0.988 (94%)	0.989 (92%)	0.947 (26%)	1.000 (100%)

5.2 Developing the imputation model

I will use the data to illustrate certain aspects of multiple imputation and the use of `ice`. Imagine that we wished to get some idea of what mandible length looks like after 28 weeks. Such a result would clearly be an extrapolation beyond observed data and should not be regarded as reliable. However, the strong relationships among the body-structure measurements indicated by table 1 makes it appealing to investigate what (if anything) can be done with `ml`.

All three fetal-size variables show variance increasing with `ga`, the trend being largely removed by log transformation of fetal size. I will therefore work with the logarithmic transformations `lnac`, `lnhc`, and `lnml`. Investigation of the relationships between each log fetal-size variable and the others and `ga` was done with Stata's `mfp` command for multivariable fractional polynomial (MFP) modeling. To avoid instability, the log transformed variables were not further transformed in the model-building phase, whereas `ga` was allowed up to a second degree FP transformation. Further, due to the large proportion (74%) of missing data in `ml`, it seemed sensible to exclude `lnml` from the prediction of `lnac` and `lnhc`. Model selection was performed at the $\alpha = 0.05$ significance level. For `lnac`, for example, the command was

```
. mfp lnac lhhc ga, select(0.05) df(1, ga:4)
```

FP2 functions of `ga` were selected, with powers of $(-2, 3)$ for `lnac` and $(0, 2)$ for `lnhc`. For predicting `lnml`, only `lnac` and `lnhc` were needed, `ga` being eliminated as not significant at the 0.05 level, conditional on `lnac` and `lnhc`. Variance explained by the model was 99% for `lnac` and `lnhc` and 94% for `lnml`. Gestational age was complete and therefore required no imputation model.

5.3 Imputation

The prediction-equation matrix for `lnac`, `lnhc`, and `lnml` obtained by MFP modeling is shown in table 2.

Table 2: Prediction equations for the fetal-size dataset

Variable	**Predictor**			
	`lnac`	`lnhc`	`lnml`	`ga`
`lnac`	–	Linear	–	FP2$(-2, 3)$
`lnhc`	Linear	–	–	FP2$(0, 2)$
`lnml`	Linear	Linear	–	–
`ga`	–	–	–	–

The transformations FP2$(-2, 3)$ and FP2$(0, 2)$ were applied to $x =$ `ga` denoting, respectively, the models $\beta_0 + \beta_1 x^{-2} + \beta_2 x^3$ and $\beta_0 + \beta_1 \ln x + \beta_2 x^2$. The implementation in `ice` is as follows: first, the FP transformations are computed and then the imputation itself. For present purposes, we will create just one imputation and store the data in a new file called `fetalimp.dta`. The following commands are required:

```
. generate ga_1=ga^-2

. generate ga_2=ga^3

. generate ga_3=ln(ga)

. generate ga_4=ga^2

. ice lnac lnhc lnml ga_1 ga_2 ga_3 ga_4 using fetalimp, m(1) genmiss(m_)
> eq(lnac:lnhc ga_1 ga_2, lnhc:lnac ga_3 ga_4, lnml:lnac lnhc)
   Variable | Command      | Prediction equation
------------+--------------+-------------------------------------------
            |              |
       lnac | regress      | lnhc ga_1 ga_2
       lnhc | regress      | lnac ga_3 ga_4
       lnml | regress      | lnac lnhc
       ga_1 |              | [No missing data in estimation sample]
       ga_2 |              | [No missing data in estimation sample]
       ga_3 |              | [No missing data in estimation sample]
       ga_4 |              | [No missing data in estimation sample]

Imputing 1..file fetalimp.dta saved
```

The results for `lnml` are shown as a graph of `lnml` against `ga` in figure 1.

(*Continued on next page*)

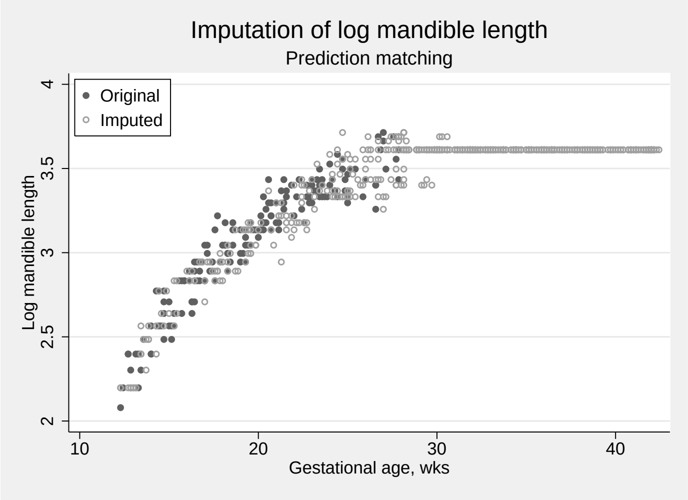

Figure 1: Original and imputed values of log mandible length using prediction matching

Although all missing values of `lnml` have been replaced, the results are unsatisfactory since a constant has been imputed for `ga` $>$ 30 weeks. The reason is that, by default, `ice` has used prediction matching to obtain imputed values. Since the distribution of `lnml` is not represented in the data beyond 28 weeks' gestation, `ice` has been forced to match according to the predictions at the highest available `ga` values for which `lnml` is not missing.

To make the imputation more realistic and useful, prediction matching may be replaced by random draws from the posterior distribution of the log fetal-size variables. This is achieved by adding the option `draw(lnac lnhc lnml)` to the `ice` command. The plot equivalent to figure 1 is shown in figure 2.

(*Continued on next page*)

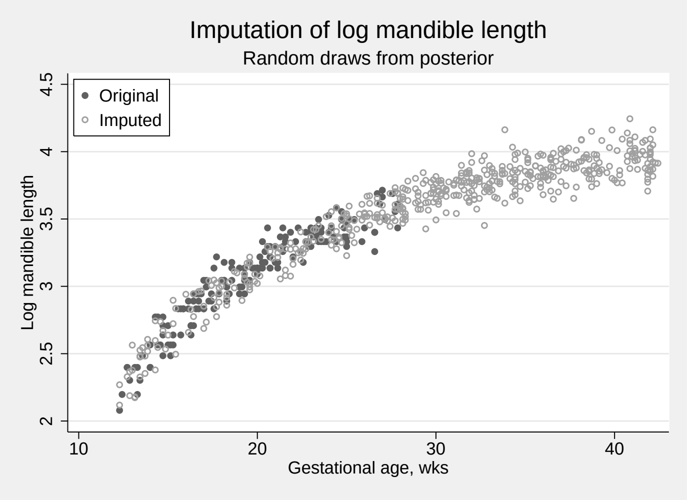

Figure 2: Original and imputed values of log mandible length using random draws from the posterior distribution of the three log fetal-size variables

The results are visually much more satisfactory. Mean log mandible length continues to increase smoothly right to the end of pregnancy, flattening off as the birth size of the fetus is approached. It should of course be stressed once again that the behavior seen in figure 2 is an extrapolation based on the assumption that the relationship between the four variables is similar in the extrapolation region to that in the region where data are observed. This is a strong assumption and, in the present example, is essentially uncheckable. However, the example does show the ability of `ice` to produce what seem to be reasonable estimates of unobserved quantities, with a plausible amount of random variation injected.

If `ga` had missing values, the `ice` command would need to include the option `passive(ga_1:ga^-2\ga_2:ga^3\ga_3:ln(ga)\ga_4:ga^2)`. The revised command would create the prediction equation `lnac lnhc lnml` for `ga`.

6 Changes to micombine

6.1 New options

There are two new options for `micombine`: `impid(`*varname*`)` and `br`. Both have been added in response to user requests. The first of these options allows data created outside of `ice` to be analyzed by using `micombine`. *varname* is the name of a variable identifying the imputations. The number of imputations is determined as the number of unique values of *varname*. The default *varname* is `_j`, the name used by `ice` when creating a dataset of imputations. The second new option, `br`, defines more precise degrees of freedom (d.f.) and confidence interval for each estimated regression coefficient.

The method is due to Barnard and Rubin (1999), hence the acronym `br`. Note that `br` is implemented for all models fit by `micombine`, not just linear regression, even though the methodology was developed in a linear regression context. Whether this is an improvement over the default inference and confidence intervals in the nonlinear-regression case remains to be investigated.

6.2 Postestimation aspects

Stata's `ereturn local`, `ereturn scalar`, and `ereturn matrix` commands set `e()` macros, scalars, and matrices returned by estimation commands. `micombine` respects this convention. It stores the sample size for one imputation (i.e., of the original data); mean regression coefficients; and estimated variance–covariance matrix, calculated according to the Rubin rules for multiple imputation, in `e(N)`, `e(b)`, and `e(V)`, respectively. The `test` and `testparm` commands work as expected following use of `micombine`, giving Wald tests of the desired parameters. Furthermore, `micombine` stores in `e(m_df)` the model d.f. and in `e(ll)` and `e(chi2)` the mean log likelihood and the mean model χ^2 statistic, each averaged over the m imputations.

The `predict` command evaluates predictions for all observations in the estimation sample using the parameters stored in `e(b)` and `e(V)`. Since the data vary across imputations, the predictions will vary correspondingly. The same principle applies to other postestimation commands that work at the level of the individual observations.

7 Conclusion

The methodology originally developed by Donald Rubin (1976) and others in the late 1970s and brilliantly implemented as the MICE system for more general use by Stef van Buuren and colleagues in the late 1990s is finally coming of age for the practitioner. I believe that this latest development will prove useful to Stata users. I hope also that it will provide a platform on which further numerical experiments in multiple imputation may be carried out. There are many outstanding questions to be tackled in the practical use of multiple imputation, including, for example, how to select an appropriate model and how to do regression diagnostics.

8 Acknowledgments

I thank numerous email correspondents and, in particular, Jennifer Hill for suggesting improvements and new features for `mvis` and `micombine`, and equally important, for pointing out unwanted "features" of the software. Without such helpful feedback, it is that much more difficult to develop flexible, reliable, and useful software.

9 References

Altman, D. G. and L. S. Chitty. 1993. Design and analysis of studies to derive charts of fetal size. *Ultrasound in Obstetrics and Gynecology* 3: 378–384.

Barnard, J. and D. B. Rubin. 1999. Small-sample degrees of freedom with multiple imputation. *Biometrika* 86: 948–955.

Carlin, J. B., N. Li, P. Greenwood, and C. Coffey. 2003. Tools for analyzing multiple imputed datasets. *Stata Journal* 3(3): 226–244.

Royston, P. 2004. Multiple imputation of missing values. *Stata Journal* 4(3): 227–241.

Rubin, D. B. 1976. Inference and missing data (with discussion). *Biometrika* 63: 581–592.

—. 1987. *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley.

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.

About the Author

Patrick Royston is a medical statistician with 25 years of experience, with a strong interest in biostatistical methodology and in statistical computing and algorithms. At present, he works in clinical trials and related research issues in kidney cancer and other cancers. Currently, he is focusing on problems of model building and validation with survival data, including prognostic factors studies; on complex sample size problems in clinical trials with a survival-time endpoint; on issues in the design and analysis of microarray studies; and on novel trial designs.