The Stata Journal (2008)
8, Number 1, pp. 49–67

A new framework for managing and analyzing multiply imputed data in Stata

John B. Carlin
Clinical Epidemiology & Biostatistics Unit
Murdoch Children's Research Institute &
University of Melbourne
Parkville, Australia
john.carlin@mcri.edu.au

John C. Galati
Clinical Epidemiology & Biostatistics Unit
Murdoch Children's Research Institute &
University of Melbourne
Parkville, Australia

Patrick Royston
Cancer and Statistical Methodology Groups
MRC Clinical Trials Unit
London, UK

Abstract. A new set of tools is described for performing analyses of an ensemble of datasets that includes multiple copies of the original data with imputations of missing values, as required for the method of multiple imputation. The tools replace those originally developed by the authors. They are based on a simple data management paradigm in which the imputed datasets are all stored along with the original data in a single dataset with a vertically stacked format, as proposed by Royston in his `ice` and `micombine` commands. Stacking into a single dataset simplifies the management of the imputed datasets compared with storing them individually. Analysis and manipulation of the stacked datasets is performed with a new prefix command, `mim`, which can accommodate data imputed by any method as long as a few simple rules are followed in creating the imputed data. `mim` can validly fit most of the regression models available in Stata to multiply imputed datasets, giving parameter estimates and confidence intervals computed according to Rubin's results for multiple imputation inference. Particular attention is paid to limiting the available postestimation commands to those that are known to be valid within the multiple imputation context. However, the user has flexibility to override these defaults. Features of these new tools are illustrated using two previously published examples.

Keywords: st0139, mim, mimstack, ice, micombine, miset, mifit, multiple imputation, missing data, missing at random

1 Introduction

The presence of missing data raises challenges for many statistical analyses, especially those based on multivariable methods where the absence of values on just one or two variables for a case will, in general, render that observation unusable in standard methods of analysis. Loss of observations from the analysis dataset in this way raises two potential threats: first, that of bias due to selection processes that may be related to the variables or—more importantly—to the associations of interest, and second, that of loss of precision (or power) due to reduction in the available sample size. Over the past two decades, considerable literature has arisen on statistical approaches to handling missing data: in particular, see the influential texts by Little and Rubin (2002) and Schafer (1997).

The leading general approach to the problem now appears to be the method of multiple imputation (MI). Briefly, this method has two distinct stages. First, a set of copies of the original dataset must be created, in which each of the missing values is imputed using an appropriate modeling procedure. Second, standard analyses are performed on each of these completed or imputed datasets, and the results (in the form of whatever parameter estimates are of substantive interest—typically regression coefficients) are then combined according to Little and Rubin's theory (2002) to obtain a set of final estimates and standard errors. This process has been outlined in more detail in an earlier article (Carlin et al. 2003).

Although at first glance MI may appear cumbersome for use in everyday data analysis, with appropriate software tools the method is not difficult to apply. Recent publication of software both for performing imputation and for analyzing the imputed datasets has led to an upsurge of usage in applied research papers. The boundaries for safe application of the method have not been fully delineated although it is well understood that the standardly available approaches all rely on an assumption that the missing data can be regarded as missing at random. This assumption is a tricky one to characterize clearly in many applications, especially those involving large datasets with many variables (Potthoff et al. 2006). Furthermore, there are a number of possible approaches for performing imputation (Schafer 1997; van Buuren, Boshuizen, and Knook 1999). Although we think the method is very effective and reliable in many situations, the underlying assumptions need to be considered carefully in any application, and further research is needed to better define the types of problems where reliable answers can be expected.

In order to facilitate better research on MI and its validity in the context of departures from assumptions, as well as to facilitate more widespread adoption of the method in practice, we have developed a comprehensive new architecture for managing the process of data analysis using MI in Stata. Within this new framework, users are able to apply all the commonly used estimation commands available in Stata, including those based on the `svy` prefix, providing a substantial extension of the previously available tools for MI analysis. We have also refined and extended the available postestimation commands, including an implementation of `predict` for multiply imputed data. These advances are provided in the form of a new prefix command, `mim`, which this article introduces.

2 Overview

In this section, we give an outline of our approach and relate the new structure to previous work by the authors.

2.1 Background

Earlier publications described a system for managing imputed datasets and performing combined analyses in Stata (Carlin et al. 2003) and described a command for creating imputations using the method of "chained equations" (`ice`) along with another command (`micombine`) for combined analyses (Royston 2004, 2005a, 2005b). The latter publications were a substantial advance on the former for two reasons: (1) they provided a method for performing the imputations, and (2) they highlighted the fact that the MI process could be handled in Stata by storing imputed versions of a dataset in a stacked format within a single dataset.

The earlier `mitools` package of commands (Carlin et al. 2003) had no facility for generating imputed values (which had to be generated externally, for example, by using the freeware NORM from http://www.stat.psu.edu/~jls/misoftwa.html) and assumed that the imputed datasets were to remain distinct files. The `mifit` command in that package performed combined analysis for a range of regression commands by repeatedly loading each imputed dataset, storing the results obtained, and performing the combined calculations at the end. The package also introduced methods for postestimation in the MI framework (commands `milincom` and `mitestparm`) and was a first attempt to create a general environment for flexible management of imputed datasets. However, the architecture adopted meant that a special-purpose command needed to be written to perform manipulations (recoding, transformation, etc.) within each imputed dataset because this required successive reloading of the datasets.

Royston's focus was on developing the `ice` command as an implementation of the method of "multiple imputation using chained equations", or "MICE" (Van Buuren, Boshuizen, and Knook 1999), with the `micombine` command provided to allow inferences to be obtained by combining analyses over the resulting imputations. Again, a wide range of regression estimation commands was accommodated. Stata's `ereturn` commands were used to allow the standard Stata postestimation commands `test` and `testparm` to work as might be expected, using estimated regression coefficients and the variance–covariance matrix obtained by pooling across the imputed datasets.

It seems clear that the best environment for managing the method of MI in Stata is based on storing the imputed datasets in stacked form in a single dataset, as in `ice` and `micombine`. The `mim` prefix command described in this article provides a new integrated framework for MI in Stata using this paradigm. To be compatible with `mim`, a dataset must contain two variables `_mj` and `_mi` that index, respectively, the individual datasets within the stack and observations within the datasets. Thus `_mi` should contain the same value i for each observation from the ith individual across datasets, with the datasets being identified by `_mj` taking the values $0, 1, \ldots, m$ for the original data (`_mj` $= 0$) and

each of the imputed datasets. Most `mim` subcommands use only the imputed data (so ignore cases for which `_mj` = 0), but retaining the original data in the stack enables parallel manipulation and transformation of variables within incomplete and imputed data. Retaining the original data also allows complete-case analyses to be performed by applying the restriction `if _mj==0`.

2.2 Estimation for MI datasets

`mim` is designed mainly for the creation of combined parameter estimates from an ensemble of imputed datasets. It allows the creation of combined estimates for regression coefficients obtained from any command that has the standard Stata estimation command structure. All commonly used commands,[1] including those taking the `svy` prefix, are recognized directly by the `mim` prefix command, and others can also be used by specifying the `category(fit)` option with `mim` (see section 3). In the latter case, the user must take responsibility for the results because `mim` will not automatically reflect any nonstandard characteristics of commands that are not in the recognized list. While most Stata estimation commands—including those using multiple-equation models—should work seamlessly with `mim`, the user should pay attention to a command's handling of any ancillary parameters. Often these are calculated on the log scale but back-transformed for display purposes, and the associated t or z statistics, and their p-values, are sometimes suppressed. When a command that has these characteristics but is not in the recognized list is used with `mim`, all parameters will be displayed on the same scale in which they are calculated, and the corresponding t statistics and p-values will be displayed, whether or not they are valid. This behavior is consistent with Stata's `ereturn display` command.

2.3 Postestimation with MI

The method of MI was developed with a focus on the canonical activity of estimating regression models. We have maintained this focus here although Rubin's rules can be applied to any estimand for which approximate normality of the estimate is reasonably assured (and in a later release of `mim`, we plan to provide a more generic capability to create combined estimates for any user-defined scalar estimator). Rubin's combination rules have been shown to work well for scalar estimands, especially when a small-sample adjustment is applied to the degrees of freedom used for the t reference distribution (Barnard and Rubin 1999) (and assuming that the method of imputation is *proper*; Rubin [1996]). For standard fitting of regression models, the scalar approach is adequate because the estimation of each coefficient in the linear predictor may be treated separately from the other coefficients, using each coefficient's estimated standard error or variance.

1. These are `regress`, `mean`, `proportion`, `ratio`, `logistic`, `logit`, `ologit`, `mlogit`, `probit`, `oprobit`, `poisson`, `glm`, `binreg`, `nbreg`, `gnbreg`, `blogit`, `clogit`, `cnreg`, `mvreg`, `rreg`, `qreg`, `iqreg`, `sqreg`, `bsqreg`, `stcox`, `streg`, `xtgee`, `xtreg`, `xtlogit`, `xtnbreg`, `xtpoisson`, `xtmixed`, `svy: regress`, `svy: mean`, `svy: proportion`, `svy: ratio`, `svy: logistic`, `svy: logit`, `svy: ologit`, `svy: mlogit`, `svy: probit`, `svy: oprobit`, and `svy: poisson`.

However, several subsidiary estimation tasks or hypothesis tests that involve more than one coefficient are often of interest. These are managed for all standard estimation commands in Stata with a range of auxiliary commands under the heading of postestimation. We believe that some, but not all, of these standard postestimation commands can be validly translated to the MI context with our current understanding of MI. `mim` currently has the facility to handle `lincom`, `testparm`, and `predict`, which respectively provide estimates for linear combinations of the regression parameters, Wald-type hypothesis tests for groups of regression coefficients considered simultaneously, and estimates of predicted values for the units of the original dataset. (Note that postestimation methods relying on likelihood comparisons (`lrtest`) are not applicable because MI does not involve calculation of likelihood functions for the data.)

The MI version of `lincom` is straightforward; it simply requires application of Rubin's rules to the (scalar) linear combination that is of interest. However, multiparameter hypothesis testing is less straightforward because it is not clear that a valid pooled variance–covariance matrix (in a multiparameter problem) can always be obtained by a simple averaging process (Schafer 1997). We have implemented an MI version of `testparm` using the method of Li, Raghunathan, and Rubin (1991), but in this first release we have not provided a full translation of the `test` command, of which `testparm` is a special case with a more limited range of syntax. Users may apply any of Stata's postestimation commands that rely on the standard structure of Stata's returned results (in particular the vector of estimates `e(b)` and variance–covariance matrix `e(V)`) by requesting that MI values of these quantities be placed in the standard returned results. `mim` does not do this by default because we do not believe that there is adequate theory to support all the possible resulting calculations and, in particular, because of the difficulty just mentioned of ensuring a valid variance–covariance matrix. The user is referred to `help mim` for details of the objects returned in `e()` when `mim` is used with an estimation command.

We have also provided a limited implementation of Stata's `predict` command under the `mim` prefix. This produces estimates of predicted values at each observation in the estimation dataset by treating the estimand $X_i\beta$ for each observation i as a scalar parameter to which the Rubin combination formulas are applied. This calculation will often use values of X_i that are missing in the original data. A more general approach to prediction, which would allow predictions to be created for "synthetic" observations (appended as new rows of data), is a more complex task that we have not yet addressed. It requires a method for creating a joint inference for the vector β of regression coefficients in a linear predictor and then applying this to whatever set or sets of X values are specified.

2.4 Data manipulation with MI datasets

The final category of subcommands that `mim` handles are those that manipulate and transform data. Our experience is that, for practical work with complex datasets, it is essential to have the capacity to work flexibly with data after imputation has been performed. For example, imputation may be performed on raw variables that must

then be categorized or transformed in various ways to be used in planned analyses. With the previous `mitools`, imputed datasets were stored separately, so a command for managing manipulation of each imputed dataset in an ensemble was needed. In the `mim` environment, most data manipulations (`generate`, `replace`, `recode`, etc.) can be simply applied to the single stacked dataset. Assuming that the original data with missing values has been retained in the stacked `mim` dataset (with `_mj` = 0), the specification of data transformations should appropriately allow for any missing values, i.e., in general by explicit exclusions such as "`if var!=.`".

`mim` was specifically programmed for three data manipulation commands (`reshape`, `append`, and `merge`) that cannot simply be applied to the stacked dataset because they require that proper attention be paid to the repeated dataset structure. The `sortorder()` option is required for the use of `merge`, in order to guarantee preservation of the observation identifier across merged datasets, because the `_mi` index must be dropped while the data manipulation is performed.

As with estimation commands, other data manipulation commands may be applied at the user's discretion by specifying the option `category(manip)`, which essentially allows a command to be applied to each dataset separately, with the resulting datasets stacked back into the same structure as used originally. Note, however, that certain data transformations, such as those that generate new observations (e.g., `expand`), may produce meaningless results in the context of an MI dataset.

The `mim` prefix also supports two newly written utility subcommands: `check` and `genmiss`. The former provides a check as to whether the dataset in memory has a `mim`-compatible structure containing the indexing variables `_mj` and `_mi`. The main checks are that nonmissing values must be constant across imputed datasets and that all missing values must have been imputed. `genmiss` creates an indicator variable to contain the missing/observed status of a selected variable. These utility subcommands require that the original dataset with missing values has been included in the stacked dataset.

While `mim` is designed to facilitate the handling of multiply imputed datasets, the user should be aware of a number of other utilities that are available in Stata for managing and manipulating missing data more generally. These range from the user-written command `mvpatterns`, which enables a detailed summary of patterns of missing data, to various usages of standard Stata functions. In particular, the `rowmiss` function of `egen` is a handy tool for identifying the extent to which missing data affect observations in a dataset, as for example in `egen int nmiss = rowmiss(`*varlist*`) if _mj==0`, which would create a variable containing the number of missing values in *varlist*.

Finally, the `mim` package includes one auxiliary command, `mimstack`, which creates a `mim`-compatible dataset from an appropriate set of imputed datasets, with or without the original incomplete data.

3 Syntax

`mim` [, *mim_options*]: *command*

`mim` [, *replay_options*]

mim_options	description
General	
`category(fit|manip)`	specify whether *command* is estimation or data manipulation
`noisily`	display output from execution of *command* within each of the imputed datasets
Estimation (valid only for estimation commands)	
`dots`	display progress dots during model fitting
`noindividual`	suppress capture of results from each application of *command*
`storebv`	fill the standard list (`e(b)`, `e(V)`, etc.) of returned results for estimation commands with MI estimates
Manipulation (valid only for data manipulation commands)	
`sortorder(`*varlist*`)`	one or more variables that uniquely identify the observations in a given imputed dataset following each execution of *command*

replay_options	description
`clearbv`	clears the standard list (`e(b)`, `e(V)`, etc.) of returned results for estimation commands, but leaves intact all other items returned by `mim`
`j(#)`	fills the standard list (`e(b)`, `e(V)`, etc.) of returned results for estimation commands with the estimates corresponding to imputed dataset #
reporting_options	`level()` and `eform` options supported by *command*
`storebv`	same as for estimation, unless `j()` option is specified

`xi` is allowed as a prefix to `mim` but not as a prefix to *command*.
`svy` is allowed as a prefix to *command*.
`version` is allowed as a prefix to *command*.

4 Options

4.1 General

`category(fit|manip)` is not required for the estimation and data manipulation commands that are listed in section 2. However, it is required when any other command is used to specify the type of command that is being passed to `mim:` either estimation (`category(fit)`) or data manipulation (`category(manip)`).

`noisily` specifies that the results of the application of *command* to each of the individual imputed datasets should be displayed.

4.2 Estimation

`dots` specifies that progress dots should be displayed.

`noindividual` specifies that capture of the estimation results corresponding to the fitting of the given estimation command to each of the individual imputed datasets should be suppressed.

`storebv` specifies that the standard list of returned results for estimation commands be filled using the MI results, forcing the MI coefficient and covariance matrix estimates into `e(b)` and `e(V)`, respectively. This enables subsequent application, at the user's discretion, of Stata postestimation commands that use these quantities directly.

4.3 Manipulation

`sortorder(`*varlist*`)` must specify a list of one or more variables that uniquely identifies the observations in each of the datasets in a `mim`-compatible dataset after *command* has been applied to the given dataset (*varlist* cannot include `_mi` because the `_mj` and `_mi` variables are dropped from each dataset prior to the call to *command*). This option is not valid for `append` and `reshape` but is *mandatory* for all other data manipulation commands.

4.4 Replay

`clearbv` specifies that the standard list (`e(b)`, `e(V)`, etc.) of returned results for estimation commands be cleared. All other (`eclass`) items returned specifically by `mim` are left intact.

`j(#)` specifies that the standard list (`e(b)`, `e(V)`, etc.) of returned results for estimation commands be filled with the estimates from the #th imputed dataset.

reporting_options may include any `level()` and `eform` options supported by *command*.

`storebv` specifies that the standard list (`e(b)`, `e(V)`, etc.) of returned results for estimation commands be filled with the MI estimates, unless the `j()` option is specified.

(There are no *mim_options* for `mim: predict`, `mim: check`, and `mim: genmiss`.)

5 Example: Adolescent health cohort study

Our first illustration uses a dataset adapted from an adolescent health cohort study that was used by Carlin et al. (2003) to introduce the original `mitools` commands. Imputation was performed for this study using the stand-alone package NORM, based on fitting a multivariate normal distribution (followed by appropriate rounding of categorical variables). This produces imputations in separate files, which we may combine into a `mim` format by applying `mimstack`. Imputations were performed separately for males and females in order to preserve interactions with gender, so we first load and stack five imputed datasets (`smiF*.dta`) with the female participants, followed by a similar process with the male participants:

```
. mimstack, m(5) sortorder(id wave) istub(smiF) clear
. save smifimp5, replace
file smifimp5.dta saved
. mimstack, m(5) sortorder(id wave) istub(smiM) clear
. save smimimp5, replace
file smimimp5.dta saved
```

We then join the two `mim` datasets into one by using `mim: append`, at the same time creating a variable `sex` to identify the two genders. The `check` utility is used to verify that we have created a dataset in `mim`-compatible format.

```
. use smifimp5, clear
. gen byte sex = 1
. mim: append using smimimp5
. replace sex = 0 if sex == .
(3420 real changes made)
. label define sexlb 0 "male" 1 "female"
. label values sex sexlb
. mim: check
..........
PASS
. save smiall, replace
file smiall.dta saved
```

The `mim` dataset `smiall.dta` is now ready for analysis and will remain in memory during the course of subsequent `mim` commands.

(*Continued on next page*)

```
. describe
Contains data from smiall.dta
  obs:         7,020
 vars:            12                          13 Mar 2008 07:48
 size:       140,400 (86.6% of memory free)   (_dta has notes)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
_mj             byte   %8.0g                  imputation identifier
_mi             int    %8.0g                  observation identifier
id              long   %9.0g
wave            byte   %9.0g                  survey wave
mmetro          byte   %9.0g                  school in metro area
parsmk          byte   %9.0g                  either parent smokes
drkfre          byte   %16.0g      drkfre      drinking frequency
alcdos          byte   %21.0g      alcdos      av units/drinking day
alcdhi          byte   %9.0g                  drank >=5 units at least once
smk             byte   %13.0g      smk         smoking status
cistot          byte   %9.0g                  CIS total score
sex             byte   %8.0g       sexlb
-------------------------------------------------------------------------------
Sorted by:  _mj  _mi
```

When using MI, it is sometimes useful to informally examine the variation in values across imputed datasets. This can be done with standard Stata syntax by using the `_mj` index. For example, one could examine the distribution of drinking frequency (a four-category variable) among imputed and nonimputed cases by running tables as follows:

```
. mim: genmiss drkfre
. by _mj: tabulate drkfre _mim_drkfre, col
```
(*output omitted*)

To illustrate a more targeted analysis, we generate a binary variable `drkreg` and obtain estimates of the frequency of regular drinking at each wave by using the command `mim: proportion`. (`proportion` is a Stata estimation command—available from release 9—and so has been incorporated into the standard `mim` structure, making it unnecessary to have a separate command such as `mici` in the previous `mitools`.)

```
. gen drkreg = (drkfre >= 2) if drkfre != .
(163 missing values generated)
. forvalues num = 1/6 {
  2. dis "wave: " `num´
  3. mim: proportion drkreg if wave==`num´
  4. }
wave: 1
Multiple-imputation estimates (proportion)              Imputations =       5
Proportion estimation                                   Minimum obs =     195
                                                        Minimum dof =   180.3
```

	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Int.]	MI.df
0	.876923	.023918	36.66	0.000	.829728	.924118	180.3
1	.123077	.023918	5.15	0.000	.075882	.170272	180.3

(*output omitted*)

Issuing the command `mim` on its own replays the last set of results produced by a `mim` estimation command, in this case for `wave==6`:

```
. mim
Multiple-imputation estimates (proportion)              Imputations =       5
Proportion estimation                                   Minimum obs =     195
                                                        Minimum dof =    35.2
```

	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Int.]	MI.df
0	.644103	.040777	15.80	0.000	.561336	.726869	35.2
1	.355897	.040777	8.73	0.000	.273131	.438664	35.2

The MI (combined) estimates are displayed using a standard Stata format with a few variations to convey important information about the MI results. The number of imputed datasets is shown, and under this we have the *minimum* number of observations available for each of the separate analyses. In many cases (including the example shown here), the number of observations will be identical across imputed datasets, but this is not the case if the estimation is performed on a subset of the data defined by restriction according to a variable that is subject to missing values. In that case, the sample used for estimation will generally differ across imputations. Displaying the minimum sample size is a conservative approach; for some purposes, the user may prefer to obtain the average to display in tables of results. The final column in the table contains the approximate degrees of freedom (Barnard and Rubin 1999) that are used for defining the t multiplier underlying the confidence interval calculation. This column also gives a useful index of the extent to which missingness has affected the information available for the estimation of each parameter. The value "Minimum dof" gives the minimum of the "`MI.df`" across the effects that have been estimated (as well as across the datasets of varying size, if applicable). In this example of `proportion` applied to a binary variable, the standard error and associated degrees of freedom are identical for each of the two complementary proportions.

The variation in results underlying the combined estimate, across imputed datasets, could be examined by replaying the single imputation results, as follows:

```
. forvalues num = 1/5 {
  2. mim, j(`num´)
  3. }
```

(*output omitted*)

An important feature is that `mim` can take the `xi` prefix to generate interactions and dummy variables in the standard way. We illustrate this with a logistic regression that examines evidence for a different rate of change with `wave` between the `sex`es by fitting an interaction model. The (incorrect) independent-observations likelihood is used for estimation (i.e., the standard `logistic` command) with standard errors obtained by the robust sandwich method in order to allow for correlation between repeated measures on the same subjects.

```
. xi: mim: logistic drkreg i.sex*wave, cluster(id)
i.sex             _Isex_0-1           (naturally coded; _Isex_0 omitted)
i.sex*wave        _IsexXwave_#        (coded as above)
Multiple-imputation estimates (logistic)                Imputations =       5
Logistic regression                                     Minimum obs =    1170
                                                        Minimum dof =   228.7
```

drkreg	Odds Rat.	Std. Err.	t	P>|t|	[95% Conf.	Int.]	MI.df
_Isex_1	.522541	.203362	-1.67	0.096	.243466	1.12151	975.9
wave	1.22544	.071734	3.47	0.001	1.09194	1.37526	228.7
_IsexXwave_1	1.03796	.084476	0.46	0.647	.884479	1.21807	391.7

This model may be used to illustrate the use of `mim: lincom`; we estimate the odds ratio for regular drinking among males as follows:

```
. mim: lincom wave + _IsexXwave_1
Multiple-imputation estimates for lincom                    Imputations = 5
 ( 1)  wave + _IsexXwave_1 = 0
```

drkreg	Odds Rat.	Std. Err.	t	P>|t|	[95% Conf.	Int.]	MI.df
(1)	1.27195	3.37316	0.09	0.928	.006988	231.506	997.9

We note again that `mim` recognizes the `logistic` command and so by default returns estimates in exponentiated form, labeled appropriately as odds ratios. When a similar logistic regression model is estimated using the generalized estimating equations method, the default display of the estimates is in the log scale, i.e., as the coefficients in the linear predictor. However, exponentiated coefficients may be obtained as usual by using the `eform` option.

```
. mim: xtgee drkreg sex wave, fam(binom) i(id)
Multiple-imputation estimates (xtgee)                    Imputations =       5
                                                         Minimum obs =    1170
                                                         Minimum dof =    78.6
```

drkreg	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Int.]	MI.df
sex	-.493965	.23835	-2.07	0.040	-.964602	-.023328	163.7
wave	.219717	.038719	5.67	0.000	.142644	.29679	78.6
_cons	-1.67304	.224338	-7.46	0.000	-2.11679	-1.22929	132.3

We illustrate the multiparameter postestimation capabilities of `mim` by including further covariates in the model:

```
. gen cisgp = cistot
(165 missing values generated)
. recode cisgp 0/5=1 6/11=2 12/100=3
(cisgp: 6300 changes made)
. xi: mim: xtgee drkreg sex wave i.cisgp, fam(binom) i(id) eform
i.cisgp           _Icisgp_1-3         (naturally coded; _Icisgp_1 omitted)
Multiple-imputation estimates (xtgee)                    Imputations =       5
                                                         Minimum obs =    1170
                                                         Minimum dof =    74.1
```

drkreg	exp(b)	Std. Err.	t	P>|t|	[95% Conf.	Int.]	MI.df
sex	.558385	.133963	-2.43	0.016	.348033	.895874	226.0
wave	1.28066	.052581	6.03	0.000	1.18006	1.38983	74.1
_Icisgp_2	1.00571	.192688	0.03	0.976	.689186	1.4676	189.2
_Icisgp_3	1.77553	.353741	2.88	0.004	1.19932	2.62857	255.8

```
. mim: testparm _Icis*
 ( 1)  _Icisgp_2 = 0
 ( 2)  _Icisgp_3 = 0
       F(  2, 221.4) =    4.76
            Prob > F =    0.0095
```

A test of the overall null hypothesis of no differences between the three groups defined by the `cisgp` variable (a categorical indicator of mental health) was obtained by using the `mim: testparm` command.

Because these data relate to measures taken on repeated occasions, some analyses may best be handled by reshaping the data to wide form. This is accomplished by using `mim: reshape`:

```
. gen drkany = (drkfre >= 1) if drkfre != .
(163 missing values generated)
. keep _mj _mi id wave drkany cisgp sex
. mim: reshape wide drkany cisgp, i(id) j(wave)
```

We can now obtain an estimate of the incidence of alcohol use between waves 1 and 2:

```
. mim: proportion drkany2 if drkany1 == 0
Multiple-imputation estimates (proportion)              Imputations =       5
Proportion estimation                                   Minimum obs =     136
                                                        Minimum dof =   122.6
```

	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Int.]	MI.df
0	.696501	.040309	17.28	0.000	.616709	.776292	122.6
1	.303499	.040309	7.53	0.000	.223708	.383291	122.6

Logistic regression can be used to examine the association between incidence and covariates of interest:

```
. xi: mim: logistic drkany2 i.cisgp1 if drkany1 == 0
i.cisgp1          _Icisgp1_1-3        (naturally coded; _Icisgp1_1 omitted)
Multiple-imputation estimates (logistic)                Imputations =       5
Logistic regression                                     Minimum obs =     136
                                                        Minimum dof =   524.7
```

drkany2	Odds Rat.	Std. Err.	t	P>|t|	[95% Conf.	Int.]	MI.df
_Icisgp1_2	.864098	.406523	-0.31	0.756	.343045	2.17658	642.2
_Icisgp1_3	1.06857	.484563	0.15	0.884	.438557	2.60365	595.8

This analysis provides an example where the size of the imputed dataset used in each of the single-imputation analyses varies because the condition `drkany1==0` produces a different set of observations (because `drkany` was subject to missingness and so varies across imputations).

6 Example: Breast cancer

We use a second example, taken from Royston (2004), to illustrate the use of `ice` to obtain multiply imputed data, followed by `mim` to handle and analyze the imputations.

First, the raw data containing missing values is loaded, and `stset` is used to specify a survival time structure for later analysis. This could have been done subsequently although the summary information provided would be potentially misleading because it would reflect the number of imputed datasets that were created. Second, five imputations of the missing values are created using `ice` (version 1.4.0; Royston 2007), saving the imputations to a new file, `brcaeximp2b.dta`. We use the `match()` option for the variable `mx6` because it has an extremely skewed, semicontinuous distribution that makes it difficult to impute using a parametric model.

```
. use brcaex, clear
(German breast cancer data)
. stset rectime, fail(censrec)
     failure event:  censrec != 0 & censrec < .
obs. time interval:  (0, rectime]
 exit on or before:  failure
------------------------------------------------------------------------------
      686  total obs.
        0  exclusions
------------------------------------------------------------------------------
      686  obs. remaining, representing
      299  failures in single record/single failure data
   771400  total analysis time at risk, at risk from t =         0
                            earliest observed entry t =         0
                                 last observed exit t =      2659
. ice mx1 mx4a mx5e mx6 mhormon lnt _d using brcaeximp2b, match(mx6) m(5)
> genmiss(m_) seed(101) replace
  #missing |
    values |      Freq.     Percent        Cum.
-----------+-----------------------------------
         0 |        231       33.67       33.67
         1 |        290       42.27       75.95
         2 |        126       18.37       94.31
         3 |         33        4.81       99.13
         4 |          6        0.87      100.00
-----------+-----------------------------------
     Total |        686      100.00
  Variable | Command | Prediction equation
-----------+---------+-------------------------------------------------------
       mx1 | regress | mx4a mx5e mx6 mhormon lnt _d
      mx4a | logit   | mx1 mx5e mx6 mhormon lnt _d
      mx5e | regress | mx1 mx4a mx6 mhormon lnt _d
       mx6 | regress | mx1 mx4a mx5e mhormon lnt _d
   mhormon | logit   | mx1 mx4a mx5e mx6 lnt _d
       lnt |         | [No missing data in estimation sample]
        _d |         | [No missing data in estimation sample]
-----------+---------+-------------------------------------------------------
Imputing 1..2..3..4..5..(note: file brcaeximp2b.dta not found)
file brcaeximp2b.dta saved
```

A plot of the distributions of observed and imputed values of one of the variables subject to missing data (`mx1`) illustrates the variability between imputations but reveals a similar distribution for the imputed values as for the observed, although one of the imputed distributions is somewhat different from the others (figure 1). Slightly different results will be obtained each time the imputation procedure is performed (unless the `seed()` option is used in `ice`); this is a natural feature of the method.

(*Continued on next page*)

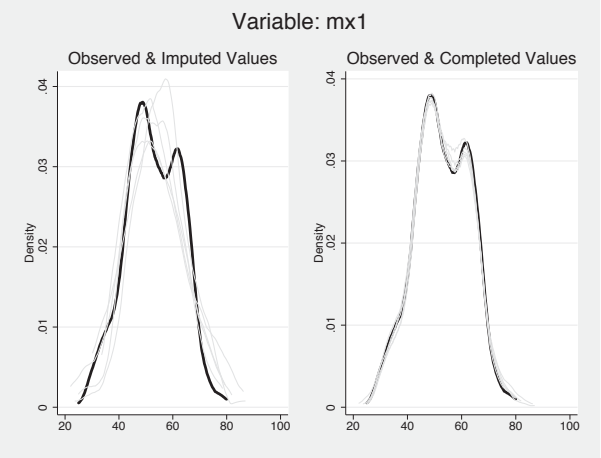

Figure 1. Frequency plots of the observed and imputed values of the variable `mx1` in the breast cancer example. The left-hand panel superimposes the distribution of the five sets of imputed values (light-gray lines) on the distribution of the observed values (black lines), while the right-hand panel displays the distribution of the completed data—observed and imputed values combined—along with the incomplete observed data distribution.

Fractional polynomial transformations are applied to `mx1` and `mx6` for modeling purposes:

```
. use brcaeximp2b, clear
(German breast cancer data)
. fracgen mx1 -2 -0.5
-> gen double mx1_1 = X^-2
-> gen double mx1_2 = X^-0.5
   (where: X = mx1/10)
. fracgen mx6 0.5
-> gen double mx6_1 = X^0.5
   (where: X = (mx6+1)/1000)
```

The model is fitted in each imputed dataset and combined estimates are obtained:

```
. mim: stcox mx1_1 mx1_2 mx4a mx5e mx6_1 mhormon, nohr
Multiple-imputation estimates (stcox)                    Imputations =       5
                                                         Minimum obs =     686
                                                         Minimum dof =     8.8
```

_t	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Int.]	MI.df
mx1_1	36.459	19.3132	1.89	0.092	-7.38798	80.3059	8.8
mx1_2	-14.9337	8.25602	-1.81	0.103	-33.5639	3.69646	9.1
mx4a	.5223	.290443	1.80	0.075	-.053744	1.09834	102.7
mx5e	-1.86353	.273425	-6.82	0.000	-2.417	-1.31005	38.1
mx6_1	-1.97985	.440288	-4.50	0.000	-2.87119	-1.0885	38.0
mhormon	-.422055	.163597	-2.58	0.016	-.757682	-.086429	27.1

7 Conclusions

The field of MI data analysis is still young, but it is quickly growing and increasingly offers the possibility of more efficient and more informative analyses of important datasets, particularly in the social and health sciences. Following the success of our earlier package `mitools` (Carlin et al. 2003) and of the package `ice` for multiple imputation of missing values (Royston 2004, 2005b), our new package `mim` further rationalizes and advances the management and analysis of MI datasets. The approach used by `mim` requires all imputed copies of the dataset to be stored together in stacked format, allowing all analysis to take place using the single dataset in memory. This approach is conceptually appealing in that it reminds the analyst that the individual imputed datasets should not be taken too seriously on their own: it is only by analyzing the multiply imputed datasets and appropriately combining results that valid inferences may be obtained. In this sense, the imputed data are naturally viewed as an extension of the original data. Use of the `mim` framework does, however, require that the user not forget that they are using a multiply imputed dataset; it is easy to mistakenly apply commands to the entire stacked dataset with the illusion of having several times more observations than actually exist. Clearly, there is an inherent complexity in using MI, which requires that the user always needs to be alert to such issues.

While there is certainly room for further development of `mim` (for example, to extend the `test` postestimation command), we believe the current version already provides a rich set of facilities for the analysis of MI data and for research on MI inference. Examples of research questions with MI data include how to build multivariable models from a set of candidate variables and how to construct suitable model performance summaries and diagnostics. More broadly, important questions remain unanswered about the use of MI: for example, how sensitive are results to the use of inappropriate imputation methods, and are there ways in which users can check the validity of their imputations and resulting analytic conclusions? As mentioned in this article's introduction, the only imputation methods that are widely available in standard software assume that the data are "missing at random" according to Rubin's technical definition (Little and Rubin 2002). Although this assumption cannot, by definition, be tested in the data being analyzed, the user should consider whether it has a good basis in the context of his or her application, and it would be helpful to have more research on the sensitivity of results to departures from missing at random (e.g., Carpenter, Kenward, and White 2007).

We hope to be able to update `mim` on a regular basis as relevant research on handling statistical issues with MI data is published and in response to user queries and suggestions. We also hope that users will develop Stata implementations of alternative methods for imputation and make them compatible with the `mim` environment so that comparative analyses are facilitated. (For example, it would be valuable to have a Stata version of Schafer's NORM.) Therefore, we welcome user input to help us further develop `mim`.

8 Acknowledgments

We are grateful to Ian White and Carolyn Coffey for testing earlier versions of `mim` and for making helpful suggestions during its development, and to Mark Lunt for suggesting a substantial improvement in the coding of `mim: predict`.

9 References

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: 226–244.

Carpenter, J. R., M. G. Kenward, and I. R. White. 2007. Sensitivity analysis after multiple imputation under missing at random: a weighting approach. *Statistical Methods in Medical Research* 16: 259–275.

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

Little, R. J. A., and D. B. Rubin. 2002. *Statistical Analysis with Missing Data*. 2nd ed. New York: Wiley.

Potthoff, R. F., G. E. Tudor, K. S. Pieper, and V. Hasselblad. 2006. Can one assess whether missing data are missing at random in medical studies? *Statistical Methods in Medical Research* 15: 213–234.

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

———. 2005a. Multiple imputation of missing values: update. *Stata Journal* 5: 188–201.

———. 2005b. Multiple imputation of missing values: update of ice. *Stata Journal* 5: 527–536.

———. 2007. Multiple imputation of missing values: further update of ice, with an emphasis on interval censoring. *Stata Journal* 7: 445–464.

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

Schafer, J. L. 1997. *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall.

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 authors

John Carlin is a biostatistician with broad experience across a range of collaborative research relating mainly to child and adolescent health, and with current methodological research interests in the handling of missing data in large longitudinal studies. He is the director of the Clinical Epidemiology and Biostatistics Unit, Murdoch Children's Research Institute, at the Royal Children's Hospital in Melbourne, Australia, and has professorial appointments in the Department of Paediatrics and School of Population Health at the University of Melbourne.

John Galati was a research officer within the Clinical Epidemiology and Biostatistics Unit, Murdoch Children's Research Institute, at the time this work was completed. He has a PhD in pure mathematics and has worked on various projects in health and medical research since 2004, as well as having substantial computer programming and systems analysis experience.

Patrick Royston is a medical statistician with 30 years of experience, with a strong interest in biostatistical methodology and in statistical computing and algorithms. 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 writing a book on multivariable regression modeling; and on new trial designs.