The Stata Journal (2004)
4, Number 3, pp. 227–241

Multiple imputation of missing values

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

Abstract. Following the seminal publications of Rubin about thirty years ago, statisticians have become increasingly aware of the inadequacy of "complete-case" analysis of datasets with missing observations. In medicine, for example, observations may be missing in a sporadic way for different covariates, and a complete-case analysis may omit as many as half of the available cases. Hotdeck imputation was implemented in Stata in 1999 by Mander and Clayton. However, this technique may perform poorly when many rows of data have at least one missing value. This article describes an implementation for Stata of the MICE method of multiple multivariate imputation described by van Buuren, Boshuizen, and Knook (1999). MICE stands for multivariate imputation by chained equations. The basic idea of data analysis with multiple imputation is to create a small number (e.g., 5–10) of copies of the data, each of which has the missing values suitably imputed, and analyze each complete dataset independently. Estimates of parameters of interest are averaged across the copies to give a single estimate. Standard errors are computed according to the "Rubin rules", devised to allow for the between- and within-imputation components of variation in the parameter estimates. This article describes five ado-files. `mvis` creates multiple multivariate imputations. `uvis` imputes missing values for a single variable as a function of several covariates, each with complete data. `micombine` fits a wide variety of regression models to a multiply imputed dataset, combining the estimates using Rubin's rules, and supports survival analysis models (`stcox` and `streg`), categorical data models, generalized linear models, and more. Finally, `misplit` and `mijoin` are utilities to interconvert datasets created by `mvis` and by the `miset` program from John Carlin and colleagues. The use of the routines is illustrated with an example of prognostic modeling in breast cancer.

Keywords: st0067, mvis, uvis, micombine, mijoin, misplit, missing data, missing at random, multiple imputation, multivariate imputation, regression modeling

1 Introduction

Following the seminal work of Rubin (1976, 1987), awareness has grown of the need to go beyond "complete-case" analysis of datasets with missing observations. In medicine, for example, it is common for observations to be missing in a sporadic way for different covariates; a complete-case analysis may omit as many as half of the available cases. Clark and Altman (2003) reported a study of prognosis in ovarian cancer in

which missing data on ten covariates reduced the complete-case sample size from 1,189 to 518 patients. Imputation of 2,045 missing values, comprising only 17% of the total of $10 \times 1,189 = 11,890$ slots in the data matrix, more than doubled the available sample size.

Hotdeck imputation was implemented in Stata by Mander and Clayton (1999) but may perform poorly when many rows of data have at least one missing value. This happens quite often and occurred in Clark and Altman's dataset just mentioned.

This article describes an implementation for Stata of the "switching regression" method of multiple multivariate imputation described by van Buuren, Boshuizen, and Knook (1999). See van Buuren's article for full details of the theory behind the method and a discussion of how to build the imputation model.

The basic idea of data analysis with multiple imputation is to create a small number, m, of copies of the data, each of which has the missing values suitably imputed. Traditionally, $m = 3$ or 5. Then, each complete dataset is analyzed independently. Estimates of parameters of interest are averaged across the m copies to give a single estimate. Standard errors are computed according to the "Rubin rules" (Rubin 1987). Recently Carlin, Li, Greenwood, and Coffey (2003) have provided a variety of useful tools for managing such imputed datasets and obtaining combined estimates. However, Carlin's routines assume that the imputed datasets have already been created; no algorithms to create imputations are provided.

Old-fashioned imputation typically replaced missing values with the mean or mode of the nonmissing values for that variable. That approach is now regarded as inadequate. For subsequent statistical inference to be valid, it is essential to inject the correct degree of randomness into the imputations and to incorporate that uncertainty when computing standard errors and confidence intervals for parameters of interest.

Here I present five ado-files. `mvis` creates multiple multivariate imputations. `uvis` imputes missing values for a single variable as a function of several covariates, each of the latter having complete data. `micombine` fits a wide variety of regression models to a multiply imputed dataset, combining the estimates using Rubin's rules. `micombine` goes further than Carlin's `mifit` routine in that it supports survival analysis models (`stcox` and `streg`), categorical data models, generalized linear models, and more. Finally, `misplit` and `mijoin` are utilities to interconvert datasets created by `mvis` and by Carlin et al. (2003)'s `miset` routine. I will give examples of the use of the routines and furthermore propose a novel method for selecting the number m of imputations to give acceptable precision for confidence intervals based on the fitted model.

2 Syntax

```
mvis mainvarlist using filename [if exp] [in range] [weight], m(#)
   [boot[(varlist)] cc(ccvarlist) cmd(cmdlist) cycles(#) draw[(varlist)]
   genmiss(string) id(string) on(varlist) noconstant replace seed(#)]
```

`uvis` *regression_cmd yvar xvarlist* [`if` *exp*] [`in` *range*] [*weight*], `gen(`*newvar*`)` [`boot draw noconstant 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*] [`,` `noconstant detail eform(`*string*`) genxb(`*newvar*`) impid(`*varname*`) lrr` *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*`)`]

3 Description

`mvis` imputes missing values in *mainvarlist* m times by using "switching regression", an iterative multivariable regression technique. Missing observations are assumed to be missing at random (MAR) or missing completely at random (MCAR); van Buuren, Boshuizen, and Knook (1999) explain these concepts in a clear way. Imputed variables and all other variables not subject to imputation are stored to a new file called *filename*`.dta` by specifying `using` *filename*. Imputed variables are stored in *filename*`.dta` under their original names, overwriting the original variables. The imputed variables are stored in long format; that is, if the original dataset comprised n observations, *filename*`.dta` will contain $m \times n$ observations.

`uvis` performs univariate imputation of missing values in *yvar* based on multiple regression of *yvar* on *xvarlist* combined with random draws from the conditional distribution of the missing observations, given the observed data and covariates, or by prediction matching (see *Remarks*). `uvis` is called repeatedly by `mvis` in a cyclical fashion to perform multivariate imputation. `uvis` does not create a new dataset but saves the imputed variable in *newvar*, as defined by the `gen(`*newvar*`)` option.

`micombine` estimates the parameters of a regression of *yvar* on *covarlist*. The type of model is determined by *regression_cmd*. Parameter estimates are combined by applying Rubin's rules across several imputations obtained previously by `mvis`.

`mijoin` converts datasets identified by Carlin's `miset` routine to `mvis` format for analysis by `micombine`. The component datasets are stacked (joined vertically). The data must first be `miset`.

`misplit` converts a dataset prepared by `mvis` to `miset` format for analysis by `mifit` and other utilities. The dataset must first be read into memory.

4 Options

4.1 Options for mvis

`m(#)` sets the number of imputations required (the minimum is 1, with no upper limit).

`boot`[`(`*varlist*`)`] specifies that each member of *varlist* be imputed with the `boot` option of `uvis`. If `(`*varlist*`)` is omitted, all relevant variables are imputed with the `boot` option of `uvis`.

`cc(`*ccvarlist*`)` prevents imputation in members of *mainvarlist* for which any member of *ccvarlist* has a missing value. `cc()` signifies "complete case". This is a convenience feature only and could be achieved (cumbersomely) by using `if` *ccvar1* `!=. &` *ccvar2* `!=. ...`. Note that variables in *ccvarlist* are used for imputation only if they also appear in *mainvarlist*.

`cmd(`*cmdlist*`)` defines the regression commands to be used for each variable in *mainvarlist* when it becomes the dependent variable in the switching regression procedure used by `uvis` (see *Remarks*). The first item in *cmdlist* may be a command, such as `regress`, or may have the syntax *vars*`:` *cmd*, specifying that command *cmd* is relevant to all the variables in *vars*. Subsequent items in *cmdlist* must follow the latter syntax. For unordered categorical variables with at least three levels (e.g., blood group), *cmd* should be specified as `mlogit` to ensure sensible imputations by using multinomial logistic regression. The default *cmd* for a variable is `logit` when there are two distinct values, `mlogit` when there are 3–5 values, and `regress` otherwise.

`cycles(#)` determines the number of cycles of regression switching to be carried out. The default # is 10.

`draw`[`(`*varlist*`)`] specifies that each member of *varlist* be imputed with the `draw` option of `uvis`. If `(`*varlist*`)` is omitted, all relevant variables are imputed with the `draw` option of `uvis`.

`genmiss(`*string*`)` creates an indicator variable for the missing data in any variable in *varlist* for which at least one value has been imputed. The indicator variable is set to missing for observations excluded by `if`, `in`, etc. The indicator variable for *xvar* is named *string*xvar.

`id(`*string*`)` creates a variable called *string* containing the original sort order of the data. The default *string* is `_i`.

`on(`*varlist*`)` changes the operation of `mvis` in a major way. With this option, `uvis` imputes each member of *mainvarlist* univariately on *varlist*. This provides a convenient way of producing multiple imputations when imputation for each variable in *mainvarlist* is to be done univariately on a set of complete predictors.

`noconstant` suppresses the regression constant in all regressions.

`replace` permits *filename* to be overwritten with new data.

`seed(#)` sets the random number seed to #. To reproduce a set of imputations, the same random number seed should be used. The default # is 0, meaning no seed is set by the program.

4.2 Options for uvis

`gen(`*newvar*`)` creates *newvar* to hold the original (nonmissing) and imputed (originally missing) values of *yvar*.

`boot` invokes a bootstrap method for creating imputed values (see *Remarks*).

`draw` draws imputations at random from the posterior distribution of the missing values of *yvar*, conditional on the observed values and the members of *xvarlist*. The default method of imputation is prediction matching (see *Remarks*).

`noconstant` suppresses the regression constant in all regressions.

`replace` permits *newvar* (see `gen(`*newvar*`)`) to be overwritten with new data.

`seed(#)` sets the random number seed to #. See *Remarks* for comments on how to ensure reproducible imputations using the `seed()` option. The default # is 0, meaning that no seed is set by the program.

4.3 Options for micombine

`noconstant` suppresses the regression constant in all regressions.

`detail` gives details of the regression model for each imputation.

`eform(`*string*`)` specifies that the exponentiated form of the coefficients be output and that the constant not be reported; *string* is used to label the exponentiated coefficients.

`genxb(`*newvar*`)` creates *newvar* to hold the linear predictor from each regression model, averaged over all the imputations.

`impid(`*varname*`)` specifies that *varname* is the variable identifying the imputations. The number of imputations is determined as the number of unique values of *varname*. The default *varname* is `_j`.

`lrr` specifies that the Li–Raghunathan–Rubin (LRR) robust estimate of the variance–covariance matrix of the regression coefficients be used.

regression_cmd_options may be any of the options appropriate to *regression_cmd*.

4.4 Options for mijoin and misplit

`clear` is not optional and confirms that you are willing to replace the data in memory.

`impid(`*varname*`)` specifies that *varname* is the variable identifying the imputations. The number of imputations is determined as the number of unique values of *varname*. The default *varname* is `_j`.

`m(#)` sets the number of imputed datasets to #.

5 Remarks

5.1 Univariate imputation

`uvis` (univariate imputation sampling) imputes missing values of the variable *yvar* from complete cases in *xvarlist*. The algorithm used is exactly as described by van Buuren, Boshuizen, and Knook (1999, section 3.2). The use of prediction matching ensures that values are imputed only within the observed distribution of *yvar*.

With the `boot` option of `uvis`, the parameter $\widehat{\beta}_*$ is estimated by regression of *yvar* on *xvarlist* within a bootstrap sample. The bootstrap method has the advantage of robustness since it is not necessary to assume that $\widehat{\beta}$ is normally distributed.

Some comments are required about prediction matching when *yvar* is a categoric variable. When *yvar* is binary and logistic regression is used, no issue arises. Prediction matching may be done on the logistic or the probability scale, almost always with identical results (`uvis` uses the logistic scale). With multicategory predictors, either ordinal (`ologit`) or multinomial (`mlogit`) logistic regression may be used to model *yvar*, as appropriate. In such cases, for a given missing value of *yvar*, one finds the observed value of *yvar* that minimizes the mean absolute difference between the logits of the predicted class probabilities. Specifically, let $\widehat{\pi}_{\text{obs};i;j}$ and $\widehat{\pi}_{\text{mis};j}$ denote the predicted probability that the ith nonmissing observation and the target missing observation of *yvar*, respectively, will fall into class j ($j = 1, \ldots, K$). These predictions are obtained from an `ologit` or `mlogit` fit of *yvar* on *xvarlist*. The imputation for the target observation is observation number

$$\arg\min_i \sum_{j=1}^{K} \left|\text{logit}\left(\widehat{\pi}_{\text{obs};i;j}\right) - \text{logit}\left(\widehat{\pi}_{\text{mis};j}\right)\right|$$

among the nonmissing values of *yvar*.

5.2 Multivariate imputation

`mvis` carries out multivariate imputation on *mainvarlist* by using "regression switching" (van Buuren, Boshuizen, and Knook [1999]). The algorithm is a type of Gibbs sampler in which the distribution of missing values of a covariate is sampled conditional on the distribution of the remaining covariates. Each variable in *mainvarlist* becomes in turn

the response variable; `uvis` is used to impute its missing values univariately on the remaining variables. Let the variables in *mainvarlist* be $x_1, x_2, \ldots, x_k$. The procedure is as follows:

1. Ignore observations for which every member of $x_1, x_2, \ldots, x_k$ or any member of *ccvarlist* (if there are any) has a missing value.
2. For each variable with any missing data in $x_1, x_2, \ldots, x_k$, randomly order that variable and replicate its observed values across the missing cases. This step initializes the iterative procedure by filling in missing data at random.
3. For each of $x_1, x_2, \ldots, x_k$, in turn, impute missing values by applying `uvis` with the remaining variables as covariates.
4. Repeat step 3 # times, where # is specified by the `cycles(#)` option. At each cycle, replace previous imputations with updated ones obtained from the latest application of `uvis`. This creates a single imputation sample.
5. To obtain m imputations, repeat the procedure m times independently. The number of imputations is controlled by the `m(#)` option of `mvis`.

At step 4, van Buuren, Boshuizen, and Knook (1999) recommend 20 cycles but go on to say that 10 or even 5 are probably sufficient. I chose to use a default of 10.

6 Example

We will again work with the breast cancer dataset which was analyzed in detail by Sauerbrei and Royston (1999). The data are provided in `brcancer.dta` and relate to a set of 686 patients with node-positive breast cancer. The outcome of interest is the recurrence-free survival time (RFS), that is, the duration in years from entry into the study (typically, the time of diagnosis of primary breast cancer) until either death or disease recurrence, whichever occurred first. There were 299 events for this outcome and the median follow-up time was about 5 years.

Sauerbrei and Royston (1999) derived a Cox proportional-hazards model for RFS that included five covariates: age (`x1`) with a fractional polynomial transformation with powers -2 and -0.5, tumor grade 2/3 (`x4a`), number of positive lymph nodes (`x5`) with the exponential transformation `x5e` $= \exp(-0.12 * \texttt{x5})$, progesterone receptors (`x6`) with a fractional polynomial transformation with power 0.5, and hormonal therapy with tamoxifen (`hormon`). This model may be fit in Stata as follows:

(*Continued on next page*)

```
. webuse brcancer
(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
. fracgen x1 -2 -0.5
-> gen double x1_1 = X^-2
-> gen double x1_2 = X^-0.5
   (where: X = x1/10)
. fracgen x6 0.5
-> gen double x6_1 = X^0.5
   (where: X = (x6+1)/1000)
. stcox x1_1 x1_2 x4a x5e x6_1 hormon, nohr
         failure _d:  censrec
   analysis time _t:  rectime
Iteration 0:   log likelihood = -1788.1731
(Iteration log omitted)
Cox regression -- Breslow method for ties
No. of subjects =          686                     Number of obs   =       686
No. of failures =          299
Time at risk    =       771400
                                                   LR chi2(6)      =    153.11
Log likelihood  =   -1711.6186                     Prob > chi2     =    0.0000
------------------------------------------------------------------------------
          _t |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        x1_1 |   43.55382   8.253433     5.28   0.000     27.37738    59.73025
        x1_2 |  -17.48136   3.911882    -4.47   0.000    -25.14851   -9.814212
         x4a |   .5174351   .2493739     2.07   0.038     .0286713    1.006199
         x5e |  -1.981213   .2268903    -8.73   0.000    -2.425909   -1.536516
        x6_1 |   -1.84008   .3508432    -5.24   0.000     -2.52772    -1.15244
      hormon |  -.3944998    .128097    -3.08   0.002    -.6455654   -.1434342
------------------------------------------------------------------------------
```

Note the use of `fracgen` to transform the covariates and `stcox` to fit the Cox model. (In fact, this model could have been fitted in one step by `fracpoly`, but such usage does not generalize to multiple imputation with `micombine`.)

To illustrate multiple imputation, I created from `brcancer.dta` a new dataset `brcaex.dta` in which approximately 20% of the observations on the five covariates `x1`, `x4a`, `x5e`, `x6`, and `hormon` were replaced completely at random with missing values, creating new covariates called `mx1`, `mx4a`, `mx5e`, `mx6`, and `mhormon`. Five imputations of the missing values were created as follows:

```
. use brcaex, clear
(German breast cancer data)
. mvis mx1 mx4a mx5e mx6 mhormon lnt _d using brcaeximp, m(5) genmiss(m_) seed(
> 101)
imputing 1..2..3..4..5..file brcaeximp.dta saved
```

The `mvis` command saves the imputations to a new file, `brcaeximp.dta`, with the original variable names `mx1`, `mx4a`, `mx5e`, `mx6`, and `mhormon` intact and with all the missing values replaced by imputations. This format is suitable for use by `micombine`. As recommended by van Buuren, Boshuizen, and Knook (1999), the (log) survival time `lnt` and the censoring indicator `_d` are included in the imputation model.

Here we use `micombine` to fit Sauerbrei and Royston's model and obtain appropriate parameter estimates and standard errors. First, the new dataset is loaded and the requisite fractional polynomial transformations are applied to `mx1` and `mx6`:

```
. use brcaeximp, 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)
```

Finally, the model is fit on each imputation, and combined estimates are obtained:

```
. micombine stcox mx1_1 mx1_2 mx4a mx5e mx6_1 mhormon
Multiple imputation parameter estimates (5 imputations)
```

_t	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
mx1_1	42.35447	14.14086	3.00	0.003	14.63889	70.07004
mx1_2	-16.83557	6.331493	-2.66	0.008	-29.24507	-4.426075
mx4a	.695139	.2944367	2.36	0.018	.1180537	1.272224
mx5e	-1.862887	.2759056	-6.75	0.000	-2.403652	-1.322122
mx6_1	-1.81862	.4307429	-4.22	0.000	-2.662861	-.9743798
mhormon	-.3781116	.1681751	-2.25	0.025	-.7077288	-.0484944

```
686 observations.
```

The results may be compared with those from the original data presented above. The parameter estimates are similar, but since some information has been lost, the standard errors are larger with the imputed data.

6.1 A note on prediction

For a given observation, the multiple-imputation estimate of the linear predictor or index (`xb`) is the average of the estimated linear predictors over the m imputations. This may be computed by specifying the `genxb(`*newvar*`)` option of `micombine`. Out-of-

sample predictions from the combined model in each imputation may be found by using the `predict` command immediately after using `micombine`. Predictions from fitting the model separately in each imputation require the use of the `forvalues` command with `if` filtration by the values of the imputation indicator variable (`_j`). The model is fitted in each imputation and `predict` is then used for prediction.

6.2 A note on the imputation model

Inspection of the imputations for the breast cancer example shows that the imputed observations are only weakly correlated with the original ones. Consider the strongest factor, the number of positive lymph nodes (`x5`). The Spearman rank correlations for the five imputations between the original and imputed values of `x5`, computed at the 20% of randomly deleted values are only 0.16, 0.15, 0.09, 0.05, and 0.15. As an experiment, five additional imputations of each prognostic factor were made independently by applying `uvis` with single, random, uniformly distributed variables as predictors. The deviance, or minus twice the maximized partial log likelihood from each dataset, averaged over the five imputations, was 3470.8, compared with 3437.5 for the imputations based on the prognostic factors and survival time, and 3423.2 for the model fitted to the data before deletions. To compute the deviances, the linear predictor from the combined model was calculated in each dataset, and the partial likelihood was calculated for a Cox model with no covariates and with the linear predictor offset. The loss of predictive accuracy (measured by the deviance) was 14.3 for the prognostic factor-based imputations, compared with 47.6 for the random imputations. This simple experiment demonstrates the importance of preserving the multivariate structure of the original predictors in the imputations, even when the predictive power of the imputation model is not large. Use of completely random imputations is likely to give suboptimal results.

7 Choice of m

There is little discussion in the literature as to how large the number m of imputations should be. Quoting Rubin (1987), van Buuren, Boshuizen, and Knook (1999) say that "Simulation studies have shown that ... m can be as low as three for data with 20 percent of missing entries. In the following I use $m = 5$, which is a conservative choice". The basis of this statement seems to concern the precision of the vector $\widehat{\beta}$ of regression coefficients. However, the variance–covariance matrix of $\widehat{\beta}$ is also important; in particular, sufficiently accurate confidence intervals for $\widehat{\beta}$ are desirable. Based on such considerations, I shall suggest an approach to determining m. This should be seen as an initial proposal, which I hope will stimulate further research on the topic.

7.1 Rubin's rules

First, I need to state Rubin's rules for estimating a scalar quantity Q based on values from m complete-data imputations. For example, Q could be one of the components

of β. Let Q_j and W_j denote the point estimate and variance respectively from the jth ($j = 1, \ldots, m$) complete dataset. The multiple-imputation point estimate Q^* of Q is the arithmetic mean of the m complete-data estimates. The estimated variance T of Q is obtained by a components-of-variance argument, leading to the following formulas

$$T = W + \left(1 + \frac{1}{m}\right) B$$

where

$$W = \frac{1}{m} \sum_{j=1}^{m} W_j$$

$$B = \frac{1}{m-1} \sum_{j=1}^{m} (Q_j - Q^*)^2$$

are the within- and between-imputation components of variance, respectively. For confidence intervals, Rubin (1987) gives the approximation

$$Q^* \pm t_\nu \sqrt{T}$$

where the degrees of freedom ν are estimated by

$$\nu = (m-1) \left\{ 1 + \frac{W}{\left(1 + \frac{1}{m}\right) B} \right\}^2$$

and where t_ν is the appropriate fractile of the central t distribution on ν degree of freedom. Note that both ν and T are estimated from the data and that both depend on the quantity B. Note also that ν depends on $(W/B)^2$, a quantity that may have a large variance and a highly skew distribution. B itself is an estimated variance with $m - 1$ degrees of freedom.

7.2 Unreliable confidence intervals?

I explored the potential instability of the confidence coefficient $t_\nu \sqrt{T}$ within the breast cancer data. The confidence level chosen was 95%, so for example, for $\nu = 15.8$ (note that ν may be fractional here), $t_\nu = 2.122$. Using `mvis` as described above, I created a large dataset comprising $M = 450$ complete-data imputations from the artificially censored dataset `brcaex.dta`. For each of $m = 3$, 5, 10, 20, and 50, I computed the mean and 95% confidence interval for $t_\nu \sqrt{T}$. This was done by breaking the M imputed datasets into blocks of M/m datasets, applying `micombine` to each block of m imputations, and finally, summarizing the resulting replicated values of $t_\nu \sqrt{T}$. For example, with $m = 3$, there were $450/3 = 150$ such blocks; therefore, the summary statistics are based on 150 observations. Figure 1 shows for each covariate, numbered from 1 to 6, how $t_\nu \sqrt{T}$, shown as a 95% empirical confidence interval, depends on m.

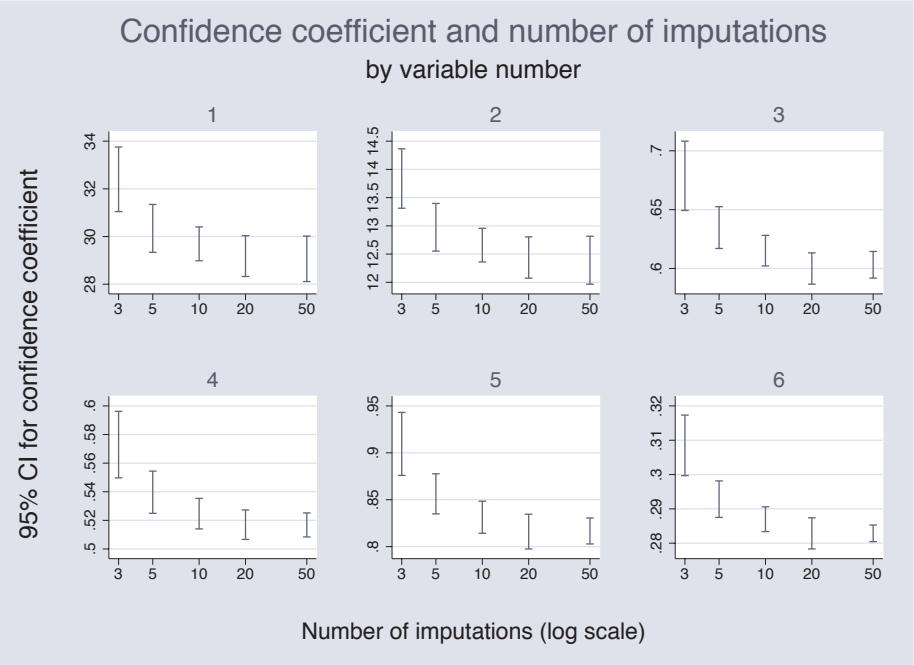

Figure 1: Breast cancer data: dependence of the 95% confidence interval for the confidence coefficient, $t_\nu\sqrt{T}$ on the number of imputations, m.

For each of the six covariates, there is a similar pattern of dependence on m. The gradual reduction of the confidence coefficient with increasing m occurs because both T and t_ν are inversely related to m. Confidence intervals for Q are correspondingly wider for lower m.

Figure 2 shows the coefficient of variation (CV = 100× standard deviation divided by the mean) of the confidence coefficient, in a format similar to figure 1.

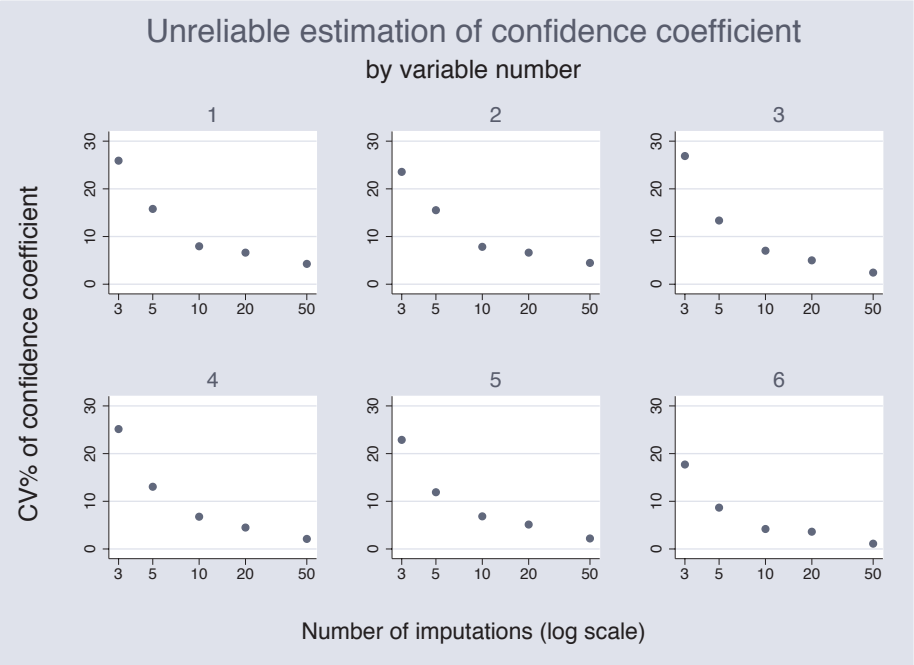

Figure 2: Breast cancer data: Coefficient of variation (100 × S.D. divided by mean) of the confidence coefficient.

The graph illustrates that the estimated confidence coefficient may be very variable for low m, and this translates into unreliable confidence intervals for Q. For example, with $m = 5$, a favorite literature choice, the CV is in the region of 13% in this example. The CV may be doubled to indicate roughly the range of uncertainty in confidence intervals for Q, in this example giving the unacceptably large figure of ±26%.

7.3 A rule of thumb for selecting m

Based on the foregoing analysis, I propose the following rule of thumb: choose m to be large enough such that the CV of the confidence coefficient for the worst-case parameter is $< 5\%$. An approximately equivalent criterion would be to require the standard deviation of ln(confidence coefficient) to be < 0.05. This is somewhat more convenient to compute. The worst-case parameter will typically attach to the variable with the greatest proportion of missing data. The rule implies that the range of uncertainty in confidence intervals for Q will be roughly $< 10\%$. In the example discussed above, the rule would require m to be at least 20 and possibly more (see figure 2).

To apply the rule, the CV should be evaluated for all parameters for which accurate estimates and confidence intervals are of central interest. This might exclude, for example, confounders (adjustment variables) in the analysis of epidemiological data.

Parameters for confounders are not normally of interest. The CV may be evaluated in the way I have indicated, by creating replicate sets of multiple imputations, using `micombine` on each and summarizing the results by standard methods. Pointers to a high CV for a given variable are a large fraction of missing data or a large value (say, > 1.2) of the between-imputation inflation factor or BIF, $\sqrt{T/W}$. The latter quantity reflects the inflation in the standard error of a parameter due to variation between imputations. The quantities B, W, T, ν, and BIF are returned by `micombine` in the matrices `e(B)`, `e(W)`, `e(V)`, `e(nu)`, and `e(BIF)`, respectively. Specifically, `e(B)`, `e(W)`, and `e(V)` are covariance matrices for the full parameter vector β, whereas `e(nu)` and `e(BIF)` are column vectors with an entry for each element of β.

An ado-file called `postmi.ado` to facilitate these calculations is under development and will be published separately. A beta version can be obtained from the author (patrick.royston@ctu.mrc.ac.uk).

8 Further comments

Sometimes it is necessary to investigate possible models, for example, prognostic models, in which selection of influential variables is required and there are missing data. See Clark and Altman (2003) for an interesting example. For example, the stability of the final model across the imputation samples is of interest.

In survival analysis, it is recommended that the log of the survival time and the censoring indicator be used as predictors in the imputation model. van Buuren, Boshuizen, and Knook (1999) give a detailed discussion of the different types of covariate that can be included in the imputation model and suggest an approach to dealing with variables that are missing not at random (MNAR).

In the present implementation of multivariate imputation sampling in `mvis`, all the variables in *varlist* are used for imputation of all the others. This restriction could in principle be lifted, but it is not clear whether the additional complexity would improve the results much.

See also van Buuren's web site *http://www.multiple-imputation.com* for further information and other software sources.

Finally, a note of caution. If the MAR assumption is valid, the methods implemented here will give correct results—but the assumption is hard to check. Common sense suggests that, as the proportion of missing observations on a given variable increases, the reliability of estimates (e.g., regression coefficients) relating to that variable will diminish. It is expected that the variance of estimates will increase, but if the MAR assumption breaks down, their (unknown) bias will also increase. Although no reliable rule of thumb is available, in my practice I am not comfortable with imputing missing values for a variable in which the proportion of missing observations exceeds approximately 50%.

9 Acknowledgment

I thank Ian White for helpful comments on the manuscript and software design and for assistance with some aspects of the programming.

10 References

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

Clark, T. G. and D. G. Altman. 2003. Developing a prognostic model in the presence of missing data: an ovarian cancer case study. *Journal of Clinical Epidemiology* 56: 28–37.

Mander, A. and D. Clayton. 1999. sg116: Hotdeck imputation. *Stata Technical Bulletin* 51: 32–34. In *Stata Technical Bulletin Reprints*, vol. 9, 196–199. College Station, TX: Stata Press.

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

—. 1987. *Multiple imputation for non-response in surveys*. New York: John Wiley & Sons.

Sauerbrei, W. and P. Royston. 1999. Building multivariable prognostic and diagnostic models: transformation of the predictors using fractional polynomials. *Journal of the Royal Statistical Society,* Series A 162: 71–94.

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 cancer. Currently, he is focusing on problems of model building and validation with survival data, including prognostic factors studies; on parametric modeling of survival data; and on novel trial designs.