A Guide to Imputing Missing Data When Using Propensity Scores with Stata
Revision: 1.3
(For Stata Version 12)

Mark Lunt

October 25, 2022

Contents

List of Figures

List of Listings

1 Introduction

Very often in observational studies of treatment effects, we have missing data for some of the variables that we wish to balance between the treated and untreated arms of the study. This leaves us with a number of options:

1. Omit the variable with the missing data from the propensity model
2. Omit the individuals with the missing data from the analysis
3. Reweight the individuals with complete data to more nearly approximate the distribution in all subjects
4. Impute the missing data

Option 1 is likely to give a biased estimate of the effect of treatment, since the treated and untreated subjects will not be balanced for the variable with missing values. Option 2 is also likely to produce a biased answer [**?**], as well as increasing the width of the confidence interval around the answer by reducing the number of subjects included in the analysis. Therefore, options 3 and 4 are preferable: this document applies to option 4.

Imputation by Chained Equations (ICE: see [**?**]) is very useful for performing imputation when you have a mixture of continuous and categorical variables. It can impute variables of various types (continuous, categorical, ordinal etc) using different regression methods, and uses an iterative procedure to allow for multiple missing values. For example, if you are imputing HAQ from DAS and disease duration, you may have subjects with both HAQ and DAS missing. You would then need to impute DAS, and use the imputed DAS to impute the HAQ.

The imputations produced by `mi impute chained` take into account the uncertainty in the predictions. That is, random noise will be added to the regression coefficients to allow for sampling error, and an error term will be added to allow for the population variation. In this way, both the mean and variance of the imputed values ought to be correct, as well as the correlations between variables.

I will illustrate all of these procedures using an analysis of data from the BSRBR. We aim to perform a survival analysis of death, with the following variables regarded as confounders:

`age`	Age
`disdur`	Disease duration
`pgen`	Gender
`haq`	HAQ Score
`dascore`	Disease Activity Score (DAS)
`dm_grp`	Number of previous DMARDs (Disease Modifying Anti-Rheumatic Drugs)

I strongly recommend that you work through obtain the dataset that I used for this example, and work through the commands yourself (the dataset can be obtained by typing

```
use http://personalpages.manchester.ac.uk/staff/mark.lunt/mi_example.dta
```

in a Stata command window. The best way to really understand how this works is to do it yourself. If you wish to work through this example, I suggest you start by entering the following commands:

```
mkdir P:/mi_guide
cd P:/mi_guide
set more off
set memory 128m

log using P:/mi_guide/mi_guide.log, text replace
```

Now you will have somewhere to store the results of your analysis.

2 Installing Packages

We are going to use one user-written add-on to Stata, `nscore`, which will tranform non-normal continuous data to normality and back. You will need to install this yourself: type

```
net from http://personalpages.manchester.ac.uk/staff/mark.lunt
```

then click on the blue `nscore` and finally `click here to install`.

3 How big is the problem ?

First, we need to see how much missing data we have. We can do this with the `misstable` command:

```
misstable summarize age disdur haq dascore pgen dm_grp
misstable patterns age disdur haq dascore pgen dm_grp
```

The output of these commands is shown in Listing 1.

Gender data is complete, and age is almost complete (9 missing values). About 11% of subjects have only missing HAQ scores, with substantially fewer having other patterns of missing data. Most subjects with missing data have only one or two variables missing, although a couple have 4 or 5.

We will want to look at the distributions of our imputed variables and compare them to the observed variables later. To do this, we set up some flags to identify observations in which particular variables are missing.

```
foreach var of varlist age disdur haq dascore pgen dm_grp {
  gen `var'_miss = `var' == .
}
```

4 First steps in imputation

Now we can start to look at how the imputation works. Using the commands built into Stata is a little different to using `ice`, which was how multiple imputation was done in Stata prior to version 11. First you need to set the data as multiple imputation data with the command

```
mi set mlong
```

. There are four styles of storing data for multiple imputation, and which one is best depends on the size of your dataset, how many variables have missing data and how many observations have missing data. The styles are

wide For each variable with missing data, Stata creates m new variables to contain the imputed values.

flong Stata creates m copies of the entire dataset in a single dataset, replacing missing values with imputed values

flongsep Stata creates m copies of the entire dataset in m separate datasets, replacing missing values with imputed values

mlong Like `flong`, but Stata only creates m new copies of observations with missing data: complete observations are only stored once.

Listing 1 Patterns of missing data

```
. misstable summarize age disdur haq dascore pgen dm_grp,  all
                                                                Obs<.
                                                +-------------------------------
             |                                  | Unique
    Variable |     Obs=.     Obs>.     Obs<.   | values        Min          Max
-------------+----------------------------------+-------------------------------
         age |         9               13,615   |     75         16           90
      disdur |       139               13,485   |     63          0           65
         haq |     1,770               11,854   |     52          0            3
     dascore |       404               13,220   |   >500          0           10
        pgen |                         13,624   |      2          0            1
      dm_grp |       190               13,434   |      6          1            6
-------------------------------------------------------------------------------

. misstable patterns age disdur haq dascore pgen dm_grp

     Missing-value patterns
       (1 means complete)

             |   Pattern
  Percent    |  1  2  3  4    5
-------------+------------------
      84%    |  1  1  1  1    1
             |
      11     |  1  1  1  1    0
       2     |  1  1  1  0    1
       1     |  1  1  1  0    0
      <1     |  1  1  0  1    1
      <1     |  1  0  1  1    1
      <1     |  1  1  0  1    0
      <1     |  1  0  1  1    0
      <1     |  1  1  0  0    1
      <1     |  1  1  0  0    0
      <1     |  0  1  1  1    1
      <1     |  1  0  0  0    0
      <1     |  1  0  0  1    0
      <1     |  0  0  0  0    0
      <1     |  0  1  0  1    0
      <1     |  0  1  1  1    0
      <1     |  1  0  0  0    1
      <1     |  1  0  0  1    1
      <1     |  1  0  1  0    0
      <1     |  1  0  1  0    1
-------------+------------------
     100%    |

  Variables are  (1) age  (2) disdur  (3) dm_grp  (4) dascore  (5) haq
```

Style `mlong` is best if most observations are complete, and just a few have missing data, but most variables have some missing data. Style `wide` would be better if there are few variables with missing data, but lots of observations. Style `flong` rarely offers any advantage over `mlong`, and requires more space, so use `mlong` instead[1]. Style `flongsep` comes into its own when your dataset is so large that it will not fit into memory using any of the other styles. However, it can be a bit tricky to use, so avoid it if you can. If you can't, make sure to read the section "Advice for using flongsep" in the Multiple Imputation Manual first.

Next, we register the variables that we need to impute data into with the command

```
mi register imputed age haq dascore disdur dm_grp
mi register regular pgen
```

This tells Stata which variables require imputation (age haq dascore disdur dm_grp) and which do not (pgen). Variables that have been registered as imputed can have different values in different imputations, whereas it would usually be a mistake to have other variables differing between imputations, and Stata would warn you.

Now we can start to impute the data. We will use linear regression to impute age, HAQ, DAS and disease duration, and ordinal regression to impute dm_grp. Since this is the first time that we have imputed any data, we need to to use the option `add()` to say how many imputations we need: we will create 20.

Note that we can include `pgen` as a predictor in our imputations without needing to impute any values for it (since there are none). The most important thing to remember with imputation is that your imputation model should be as rich as possible: any variables (or terms, such as interactions) that will be included in your analysis should be included in the imputation model. In particular, the outcome variable must be included, since if it is not then you are assuming that there is no association between the outcome and the predictors in the imputed data, and hence are diluting any association that might exist in the observed data.

As you can see, Stata will impute each of the missing variables from all of the other 5 variables. Before running the imputations, I have set the random number seed to 999, so that the same random numbers are always produced

[1]Ironically, it may be better than `mlong` for IPTW analysis, which we will be doing shortly, although I'm not aware of any evidence either way yet.

Listing 2 Initial Imputation

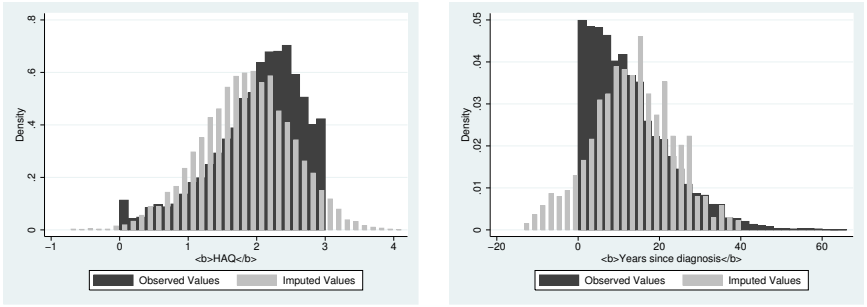
```
. preserve
. mi impute chained (regress) disdur dascore age haq ///
                    (ologit) dm_grp = pgen, add(20)
note: variable pgen contains no soft missing (.) values; imputing nothing

Conditional models:
               age: regress age i.pgen disdur i.dm_grp dascore haq
            disdur: regress disdur i.pgen age i.dm_grp dascore haq
            dm_grp: ologit dm_grp i.pgen age disdur dascore haq
           dascore: regress dascore i.pgen age disdur i.dm_grp haq
              haq: regress haq i.pgen age disdur i.dm_grp dascore

Performing chained iterations ...

Multivariate imputation                     Imputations =        5
Chained equations                                 added =        5
Imputed: m=1 through m=5                        updated =        0

Initialization: monotone                     Iterations =       50
                                                burn-in =       10

            disdur: linear regression
           dascore: linear regression
               age: linear regression
              haq: linear regression
            dm_grp: ordered logistic regression
              pgen: logistic regression

------------------------------------------------------------------
                  |               Observations per m
                  |-----------------------------------------------
         Variable |   Complete   Incomplete   Imputed |     Total
------------------+-----------------------------------+----------
            disdur |      13485          139       139 |     13624
           dascore |      13220          404       404 |     13624
               age |      13615            9         9 |     13624
               haq |      11854         1770      1770 |     13624
            dm_grp |      13434          190       190 |     13624
              pgen |      13624            0         0 |     13624
------------------------------------------------------------------
(complete + incomplete = total; imputed is the minimum across m
 of the number of filled-in observations.)
```

and the analysis is reproducible. If you copy the commands in this document, your output should be identical to mine.

Listing 3 Producing histograms of imputed data: first attempt

```
tw histogram haq if haq_miss == 0, width(0.125) color(gs4) || ///
   histogram haq if haq_miss == 1, gap(50) color(gs12)        ///
               width(0.125) legend(label(1 "Observed Values")         ///
               label(2 "Imputed Values"))
graph export haq112.eps, replace
tw histogram disdur if disdur_miss == 0, width(2) color(gs4) ||     ///
   histogram disdur if disdur_miss == 1, gap(50) color(gs12)        ///
               width(2) legend(label(1 "Observed Values")             ///
               label(2 "Imputed Values"))
graph export disdur112.eps, replace
tw histogram age if age_miss == 0, width(2) color(gs4) ||           ///
   histogram age if age_miss == 1, gap(50) color(gs12)              ///
               width(2) legend(label(1 "Observed Values")             ///
               label(2 "Imputed Values"))
graph export age112.eps, replace
tw histogram dascore if dascore_miss == 0 , width(0.2) color(gs4) || ///
   histogram dascore if dascore_miss == 1, gap(50) color(gs12)       ///
               width(0.2) legend(label(1 "Observed Values")   ///
               label(2 "Imputed Values"))
graph export dascore112.eps, replace
```

Now we can look at the imputed data, starting with continuous variables, histograms of which are shown in Figure 1. For the HAQ score, some of the imputed values are impossible: a HAQ score lies between 0 and 3, and in fact can only take certain values in this range. Imputing data with the appropriate mean and variance leads to impossible values. A similar problem exists for `disdur`: values below 0 are impossible (they correspond to negative disease durations i.e. subjects who will develop their disease in the future), and yet they are imputed. The DAS has fewer problems: it is possible that out-of-range DAS values are imputed, but it does not seem to have happened here. Only 9 values of age needed to be imputed, and they are all reasonable.



(a) HAQ

(b) Disease Duration

(c) DAS

(d) Age

Figure 1: Distribution of imputed and observed values: first attempt

5 Imputation of non-normal distributions

There is some debate about whether or not the impossible values for HAQ and disease duration are a problem: for a discussion see [**?**].

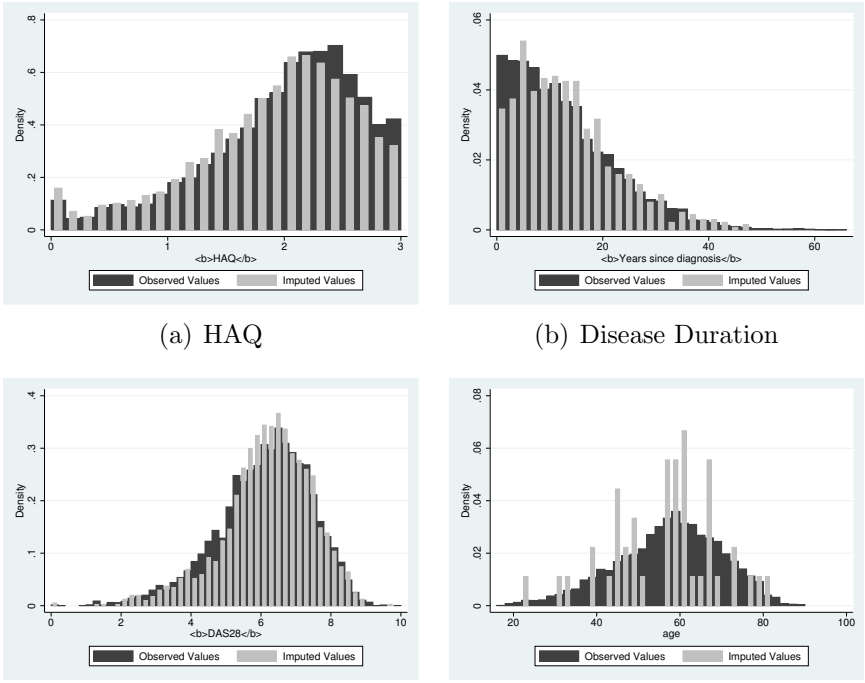
One way around the problem of impossible values is to use the command `nscore`. This will transform the variables to normality, so that they can be imputed. Then `invnscore` can be used to convert back from the normally distributed imputed variable to the (bizarre) distributions of the original variables. The command `invnscore` guarantees that imputed values cannot lie outside the observed data range. The commands needed to do this given in Listing 4[2].

It is obvious from Figure 2 that the distributions of the imputed data

[2]The figures dascore3a.eps and dascore3b.eps produced by this listing will only be needed for Figure 3 later, but since the imputed data will be changed by then, I have generated them now.

Listing 4 Producing histograms of imputed data: second attempt

```
restore
preserve

nscore age disdur haq dascore, gen(nscore)
mi register imputed nscore1-nscore4

mi impute chained (regress) nscore1-nscore4 (ologit) dm_grp ///
                  = pgen, add(5)

invnscore age disdur haq dascore

tw histogram haq if haq_miss == 0, width(0.125) color(gs4) || ///
   histogram haq if haq_miss == 1, gap(50) color(gs12)        ///
             width(0.125) legend(label(1 "Observed Values")         ///
             label(2 "Imputed Values"))
graph export haq212.eps, replace
tw histogram disdur if disdur_miss == 0, width(2) color(gs4) ||     ///
   histogram disdur if disdur_miss == 1, gap(50) color(gs12)        ///
             width(2) legend(label(1 "Observed Values")             ///
             label(2 "Imputed Values"))
graph export disdur212.eps, replace
tw histogram age if age_miss == 0, width(2) color(gs4) ||           ///
   histogram age if age_miss == 1, gap(50) color(gs12)              ///
             width(2) legend(label(1 "Observed Values")             ///
             label(2 "Imputed Values"))
graph export age212.eps, replace
tw histogram dascore if dascore_miss == 0 , width(0.2) color(gs4) || ///
   histogram dascore if dascore_miss == 1, gap(50) color(gs12)       ///
             width(0.2) legend(label(1 "Observed Values")   ///
             label(2 "Imputed Values"))
graph export dascore212.eps, replace

tw histogram dascore if dascore_miss == 0 & treated == 0, ///
             width(0.2) color(gs4) ||                     ///
   histogram dascore if dascore_miss == 1 & treated == 0, ///
             gap(50) color(gs12)       ///
             width(0.2) legend(label(1 "Observed Values")   ///
             label(2 "Imputed Values"))
graph export dascore3a12.eps, replace

tw histogram dascore if dascore_miss == 0 & treated == 1, ///
             width(0.2) color(gs4) ||                     ///
   histogram dascore if dascore_miss == 1 & treated == 1, ///
             gap(50) color(gs12)       ///
             width(0.2) legend(label(1 "Observed Values")   ///
             label(2 "Imputed Values"))
graph export dascore3b12.eps, replace
```



(a) HAQ

(b) Disease Duration

(c) DAS

(d) Age

Figure 2: Distribution of imputed and observed values: second attempt

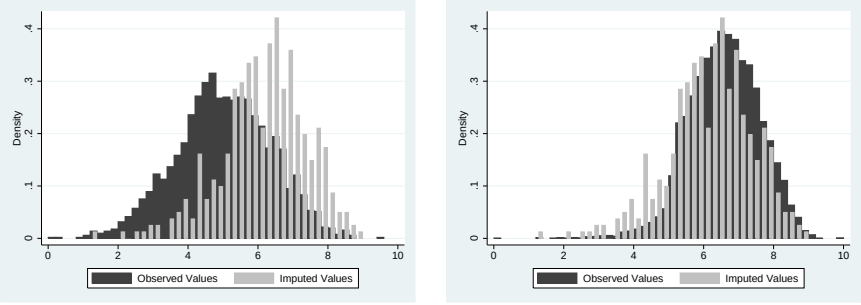
are much more similar to the distribution of the observed data now, and no longer follow a normal distribution.

6 Imputing Treated and Untreated Separately

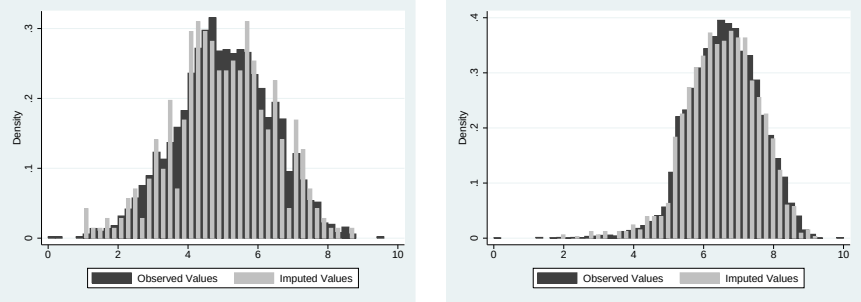
Although the distributions of imputed values look reasonable now, there is still problem. The same imputation equation is used to impute data in treated and untreated subjects, despite the big differences in these variables between the two groups. We could simply add treatment as a predictor to all of the imputation equations, but there are still differences in the associations between (for example) age and DAS in the treated and untreated that are not catered for in this way. Fitting interactions between treatment and all of the predictors is possible, but it would be easier to perform the imputations completely separately in the treated and control arms. The way to do this is illustrated in Listing 5.

Listing 5 Imputing in treated and untreated separately

```
restore
preserve

nscore age disdur haq dascore, gen(nscore)
mi register imputed nscore1-nscore4

mi impute chained (regress) nscore1-nscore4 (ologit) dm_grp ///
                  = pgen, add(20) by(treated)

invnscore age disdur haq dascore

tw histogram dascore if dascore_miss == 0 & treated == 0, ///
             width(0.2) color(gs4) ||                     ///
   histogram dascore if dascore_miss == 1 & treated == 0, ///
             gap(50) color(gs12)       ///
             width(0.2) legend(label(1 "Observed Values")   ///
             label(2 "Imputed Values"))
graph export dascore3c12.eps, replace

tw histogram dascore if dascore_miss == 0 & treated == 1, ///
             width(0.2) color(gs4) ||                     ///
   histogram dascore if dascore_miss == 1 & treated == 1, ///
             gap(50) color(gs12)       ///
             width(0.2) legend(label(1 "Observed Values")   ///
             label(2 "Imputed Values"))
graph export dascore3d12.eps, replace
```



(a) Common Imputation, Untreated (b) Common Imputation, Treated



(c) Separate Imputation, Untreated (d) Separate Imputation, Treated

Figure 3: Distribution of imputed and observed values of DAS: common & separate imputations

Figure 3 shows the effect of imputing in the treated and untreated separately. In the top panel, the imputation was performed in the treated and untreated as a single group. The distribution of observed DAS scores differ greatly between treated and untreated subjects, but the distribution of imputed DAS scores are similar in the treated and untreated, but unlike the observed values for the untreated subjects. In the lower panel, the imputation was performed in the treated and untreated separately. Now the distribution of imputed values in the treated and untreated subjects is similar to the observed values in that group of subjects.

(Bear in mind that the distributions of observed and imputed data do not need to be the same. For example, it may be that older subjects are less likely to complete a HAQ. Then the missing HAQs are likely to be higher than the observed HAQs. However, the associations between missingness and each of the variables is only minor in this instance, so the distributions

of imputed and observed data should be similar.)

7 Using the imputed data

Having generated a set of imputations, you will want to analyse them. This is not straightforward: you need to analyse each imputed dataset separately, then combine the separate estimates in a particular way (following "Rubin's Rules"[**?**]) to obtain your final estimate. Fortunately, Stata will do that for you without you needing to know what is going on. You simply precede whichever regression command you wanted use with `mi estimate:` (provided there are at least 2 imputations, which is why we used `m(5)` in the `mi impute` command earlier). So, to obtain a propensity score from our imputed data, we would enter the command

```
drop _merge
xi: mi estimate, saving(propensity): logistic treated age disdur haq dascore  ///
                  i.pgen i.dm_grp
```

Note that the `xi` command has to come *before* the `mi estimate` command. Also note that we have to save the estimation results (`saving(propensity)` if we want to use them later for prediction.

There are two ways of predicting after issuing an `mi estimate` command. The one that you will usually use is `mi predict` (or `mi predictnl`. However, this does not offer all of the options that are usually available after a regression command, since Rubin's rules are used to combine predictions in each imputation to make a single imputation per observation, and they are only valid for normally distributed variables. So you cannot use `mi predict` to obtain probabilities after a logistic regression equation, for example: you have to predict the linear predictor, which is normally distributed, then transform that into the predicted probability.

```
mi predict lp using(propensity), xb
mi passive: generate prop = exp(lp)/(1+exp(lp))
```

The above commands will created variables called `lp` and `prop` which will be the same for all imputations for a given observation, even if there was missing data in that observation. If you want predictions that vary between imputations (for example residuals from a regression equation), there is a command `mi xeq` (short for "multiple imputation execute") for running command on each impuation separately, but I'll cover that in a separate

tutorial.

We can compare the effects of predicting from the complete cases and predicting from the imputed data if we also obtain the complete case propensity scores:

```
xi: logistic treated age disdur haq dascore i.pgen i.dm_grp ///
            if _mi_m == 0
predict pc if _mi_m == 0
corr pc prop
```

If you enter the above commands, you will see that for the subjects with complete data, the propensity scores are very similar ($r = 0.9963$) whether we use the observed or imputed logistic regression equations. However, we can include substantially more subjects in our analysis by using the imputed data, as shown in Listing **??**

Listing 6 Subjects with complete and missing data

```
. tab _mi_miss treat, co

+-------------------+
| Key               |
|-------------------|
|     frequency     |
| column percentage |
+-------------------+

           |        treated
  _mi_miss |         0          1 |     Total
-----------+----------------------+----------
         0 |     1,632      9,758 |    11,390
           |     62.99      88.44 |     83.60
-----------+----------------------+----------
         1 |       959      1,275 |     2,234
           |     37.01      11.56 |     16.40
-----------+----------------------+----------
     Total |     2,591     11,033 |    13,624
           |    100.00     100.00 |    100.00
```

We have been able to include an extra 2,234 subjects in our analysis. More importantly, more than 1/3 of untreated subjects had at least one missing variable, compared to 1/8 treated subjects. Since we are short of controls to

begin with, the fact that we don't need to lose such a substantial number is a definite bonus.

It may be tempting to impute a single dataset, so that you don't need to worry about `mi estimate`. Particularly when you are exploring the data and checking for the balance of the various predictor variables, it would be easier to use standard Stata modelling commands. However, there are theoretical and empirical grounds for believing that multiple imputations can improve the precision of your parameter estimates. I would therefore recommend, having decided on your analysis strategy, to perform an entire analysis on a multiply imputed dataset.

8 Imputation Diagnostics

When analysing imputed data, it is vital to get some idea of how much uncertainty in your answer is due to the variation between imputations, and how much is inherent in the data itself. Ideally, you want very little variation between imputations: if your answer is consistent for multiple sets of imputed data, then it is more likely to be correct. In addition, there is always a concern that the imputations were not performed correctly: either there are associations between the variables that were not modelled, or the associations between the variables are different in those who did not respond compared to those who did respond (data Missing Not At Random). Even if the imputed data are incorrect, the answer may still be adequate if the imputations all give similar answers.

A very useful parameter to look at to answer this question is the proportion of missing information about a particular parameter, referred to as λ in [**?**]. Note that this parameter is not the same as the proportion of missing *data*: it may be that there is a lot of missing data about a weak confounder, which does not affect the parameter of interest greatly at all.

The variance of the parameter you are interested is

$$T = W + (1 + 1/m)B$$

Where W is the mean of the variances of the parameter in each imputation, and B is the variance between imputations. So, if we had complete data, the variance would be W in each imputation, so the relative increase in variance due to missing data is

$$\frac{(1 + 1/m)B}{W}$$

There is a related number, the fraction of missing information, which has a complicated definition but generally takes similar values and assesses the same concept: how much have we lost through the missing data.

We can look at at the fraction of missing information (FMI) and the relative variance increase (RVI) due to the missing date by using the `vartable` option with `mi estimate`. If no estimation command is given to `mi estimate`, it uses the last one, provided that the last estimation command was `mi estimate`. Since we have run a logistic regression since, we will have to rerun the `mi` estimation:

Listing 7 Missing information due to missing data

```
. xi: mi estimate, vartable: logistic treated age disdur haq dascore  ///
>                   i.pgen i.dm_grp
i.pgen            _Ipgen_0-1          (naturally coded; _Ipgen_0 omitted)
i.dm_grp          _Idm_grp_1-6        (naturally coded; _Idm_grp_1 omitted)

Multiple-imputation estimates                       Imputations     =         20
Logistic regression

Variance information
------------------------------------------------------------------------------
             |        Imputation variance                                Relative
             |    Within   Between     Total       RVI       FMI     efficiency
-------------+----------------------------------------------------------------
         age |   6.3e-06   1.5e-07   6.4e-06   .025345   .024781       .998762
      disdur |   .000011   4.4e-07   .000011   .042448   .040887        .99796
         haq |   .002346   .000603   .002979   .269874   .216238       .989304
     dascore |   .000842   .000093    .00094   .115731    .10474        .99479
    _Ipgen_1 |    .00473   .000066   .004799   .014627   .014438       .999279
  _Idm_grp_2 |    .01272   .000821   .013582   .067803   .063895       .996815
  _Idm_grp_3 |   .013199   .000526   .013751   .041835   .040318       .997988
  _Idm_grp_4 |    .01552   .000764   .016322   .051667   .049371       .997538
  _Idm_grp_5 |   .019619   .000701   .020355   .037518   .036294       .998189
  _Idm_grp_6 |   .020282   .001013   .021345   .052444   .050079       .997502
       _cons |   .054295   .002656   .057084   .051369   .049098       .997551
------------------------------------------------------------------------------
```

There are a few surprises here. First, there was no missing data for `pgen`, yet there is about 2.5% missing information. This is due to confounding:

HAQ scores are higher in the women than they are in the men, so the difference in treatment rates between men and women is partly a direct effect, and partly due to differences in HAQ. The coefficient for `pgen` is adjusted for differences in HAQ, but the values of HAQ (and hence the adjustment) vary between imputations. Hence, the coefficent of `pgen` also varies. A similar argument explains the effect of missing data on age, despite very few missing values for age: there is a very strong association between age and HAQ, so the missing values for HAQ affect the coefficient for age quite markedly.

The missing information about the HAQ is even more extreme: 13% missing data, but 22% missing information. This is because most of the missing data is in the untreated subjects, and there are already fewer of these. In fact, the 30% missing data in the untreated is very close to the 30% missing information overall.

The "Relative Efficiency" column in the table above refers to how precise each estimate is given the number of imputations used, relative to how precise it could be with an infinite number of imputations. You can see that for all parameters except HAQ score, the relative efficiency is over 99%. It is recommended that the number of imputations should be at least 100 times the largest FMI[], which would suggest we need 22 imputations here, rather than the 20 that we actually did. That should be enough to take the relative efficiency for HAQ to 99%.

References

[1] Shafer JL, Graham JW Missing data: Our view of the state of the art Psychological Methods 2002;7:147–177.

[2] van Buren S, Boshuizen HC, Knook DL Multiple imputation of missing blood pressure covariates in survival analysis Statistics in Medicine 1999; 18:681–694.

[3] White IR, Royston P, Wood AM Multiple imputation using chained equations: Issues and guidance for practice Statistics in Medicine 2011;30:377–399.

[4] Rubin DB *Multiple Imputation for Nonresponse in Surveys* New York: J. Wiley and Sons 1987.

[5] Royston P Multiple imputation of missing values The Stata Journal 2004; 4:227–241.