A Guide to Imputing Missing Data with Stata Revision: 1.4

Mark Lunt

December 6, 2011

Contents

1 Introduction **3**

2 Installing Packages **4**

3 How big is the problem ? **5**

4 First steps in imputation **5**

5 Imputation of non-normal distributions **8**

6 Imputation with categorical variables **10**

7 Imputation with interactions **14**

8 Restricting Predictors in the imputation **14**

9 Imputing Treated and Untreated Separately **15**

10 Using the imputed data **17**

11 Imputation Diagnostics **20**

12 Imputation Options **22**

List of Figures

1 Distribution of imputed and observed values: first attempt . . 10
2 Distribution of imputed and observed values: second attempt . 12
3 Distribution of imputed and observed values of DAS: common & separate imputations 17

List of Listings

1 Patterns of missing data 6
2 Regression equations used by ICE 7
3 Producing histograms of imputed data: first attempt 9
4 Producing histograms of imputed data: second attempt 11
5 Imputing from a categorical variable 13
6 Allowing for interactions in an imputation 14
7 Imputing in treated and untreated separately 16
8 Subjects with complete and missing data 19
9 Missing information due to missing data 21

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 [1], 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.

The Stata `ice` routine (Imputation by Chained Equations: see [2]) is very useful for performing imputation. 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 `ice` 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
`ovmean`	HAQ Score
`dascore`	Disease Activity Score
`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 four user-written add-ons to Stata, `mvpatterns` (to see what patterns of missing data there are), `ice` (to perform the imputation), `mim` to analyse the imputed datasets and `nscore` (to tranform data to and from normality) you will need to install these yourself. To install `mvpatterns`, type `search mvpatterns`, then click on `dm91`, and the subsequent `click here to install`.

Unfortunately, the version of ice found by `search ice` is old and buggy. To get a useable version, you need to type in `ssc install ice` (or possibly `ssc install ice, replace` if you already have an old, buggy version installed). Its best to get `mim` from the same place, since they need to work together: an old version of `mim` may not work with a recent version of `ice` and vice versa. So type

```
ssc install mim
```

Finally, to install `nscore` type

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

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

Note that the latest version of `mim` will not run under stata 9.2: you must use stata 10 (which is found at `M:/Stata10/wsestata.exe`.

3 How big is the problem ?

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

```
mvpatterns age disdur ovmean dascore pgen dm_grp
```

The output of this command is shown in Listing 1.

Gender data is complete, and age is almost complete. About 13% of subjects have missing HAQ scores, with substantially fewer having missing data in the other variables. Most subjects with missing data have only one or two variables missing, although a couple have 4 or 5. So that we dont change the existing data, and can tell which values are observed and which ones are imputed, we will create a new set of variables to put the imputed data into. The commands to do this are

```
foreach var of varlist age disdur ovmean dascore pgen dm_grp {
  gen `var'_i = `var'
}
```

4 First steps in imputation

The next stage is to see how `ice` will impute these variables, using the option `dryrun`. The results are in Listing 2.

As you can see, `ice` will impute each of the missing variables from all of the other 5 variables. However, it is using the `regress` command for all of them, which may not be appropriate. The variable `dm_grp` is categorical, so we should be using the commands `mlogit` or `ologit` (probably `ologit` in this case, since it is an ordered variable. We can use the `cmd` option to achieve this:

Listing 1 Patterns of missing data

```
variables with no mv's: pgen

Variable     | type     obs   mv   variable label
-------------+-------------------------------------------------
age          | byte   13615    9
disdur       | byte   13485  139   <b>Years since diagnosis</b>
ovmean       | double 11854 1770   <b>HAQ</b>
dascore      | double 13220  404   <b>DAS28</b>
dm_grp       | float  13434  190
-------------------------------------------------------------

Patterns of missing values

  +------------------------+
  | _pattern   _mv   _freq |
  |------------------------|
  |    +++++     0   11390 |
  |    ++.++     1    1531 |
  |    +++.+     1     206 |
  |    ++..+     2     170 |
  |    ++++.     1     129 |
  |------------------------|
  |    +.+++     1     108 |
  |    ++.+.     2      30 |
  |    +..++     2      20 |
  |    +++..     2      12 |
  |    ++...     3       9 |
  |------------------------|
  |    .++++     1       6 |
  |    +..+.     3       3 |
  |    +....     4       3 |
  |    +.++.     2       1 |
  |    +.+.+     2       1 |
  |------------------------|
  |    .+.++     2       1 |
  |    +.+..     3       1 |
  |    +...+     3       1 |
  |    .+.+.     3       1 |
  |    .....     5       1 |
  +------------------------+
```

Listing 2 Regression equations used by ICE

```
. ice  age_i disdur_i ovmean_i dascore_i pgen_i dm_grp_i, dryrun

   #missing |
     values |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |     11,390       83.60       83.60
          1 |      1,980       14.53       98.14
          2 |        235        1.72       99.86
          3 |         15        0.11       99.97
          4 |          3        0.02       99.99
          5 |          1        0.01      100.00
------------+-----------------------------------
      Total |     13,624      100.00

   Variable | Command | Prediction equation
------------+---------+-------------------------------------------------------
      age_i | regress | disdur_i ovmean_i dascore_i pgen_i dm_grp_i
   disdur_i | regress | age_i ovmean_i dascore_i pgen_i dm_grp_i
   ovmean_i | regress | age_i disdur_i dascore_i pgen_i dm_grp_i
  dascore_i | regress | age_i disdur_i ovmean_i pgen_i dm_grp_i
     pgen_i |         | [No missing data in estimation sample]
   dm_grp_i | regress | age_i disdur_i ovmean_i dascore_i pgen_i

End of dry run. No imputations were done, no files were created.
```

```
ice age_i disdur_i ovmean_i dascore_i pgen_i dm_grp_i, dryrun ///
    cmd(dm_grp_i: ologit)
```

Now lets go ahead and produce some imputations. In the first instance, we will only produce a single imputation (using the option `m(1)`), since we want to look at the distributions of the imputed data (there are still a number of problems with our imputations that will need to be fixed before we can use them). We also need to give the name of a file in which the imputed data can be stored. First, we will preserve our data so we can continue to work with it later. Then we do the imputation and load the imputed dataset into stata. Finally, we produce histograms of the observed and imputed data to check that the imputations are reasonable.

I have added the option `seed(999)` to the ice command. This sets the starting value of the random number generator in Stata, so that the same set of random numbers will be generated if we repeat the analysis. This means that our analysis is reproducible, and that if you enter the commands in Listing 3, your output should be identical to mine.

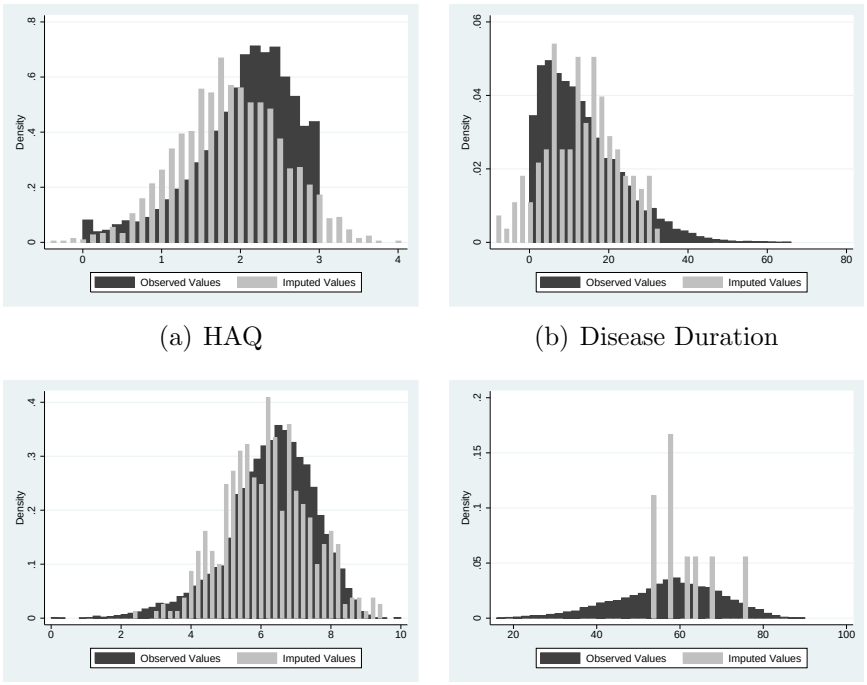
Now we can look at the imputed data, starting with continuous variables, histograms of which are shown in Figure 1. For the HAQ score, some of the imputed values are impossible: a HAQ score lies between 0 and 3, and in fact can only take certain values in this range. Imputing data with the appropriate 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.

5 Imputation of non-normal distributions

We can get around the problem of impossible values by using 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

Listing 3 Producing histograms of imputed data: first attempt

```
preserve
ice age_i disdur_i ovmean_i dascore_i pgen_i dm_grp_i,      ///
    saving("mi", replace) cmd(dm_grp_i: ologit) m(1) seed(999)
use mi, clear
tw histogram ovmean, width(0.125) color(gs4) ||             ///
   histogram ovmean_i if ovmean == ., gap(50) color(gs12)   ///
             width(0.125) legend(label(1 "Observed Values") ///
             label(2 "Imputed Values"))
graph export haq1.eps, replace
tw histogram disdur, width(2) color(gs4) ||                 ///
   histogram disdur_i if disdur == ., gap(50) color(gs12)   ///
             width(2) legend(label(1 "Observed Values")     ///
             label(2 "Imputed Values"))
graph export disdur1.eps, replace
tw histogram age, width(2) color(gs4) ||                    ///
   histogram age_i if age == ., gap(50) color(gs12)         ///
             width(2) legend(label(1 "Observed Values")     ///
             label(2 "Imputed Values"))
graph export age1.eps, replace
tw histogram dascore, width(0.2) color(gs4) ||              ///
   histogram dascore_i if dascore == ., gap(50) color(gs12) ///
             width(0.2) legend(label(1 "Observed Values")   ///
             label(2 "Imputed Values"))
graph export dascore1.eps, replace
```



(a) HAQ (b) Disease Duration

(c) DAS (d) Age

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

in Listing 4[1].

It is obvious from Figure 2 that the distributions of the imputed data are much more similar to the distribution of the observed data now, and no longer follow a normal distribution.

6 Imputation with categorical variables

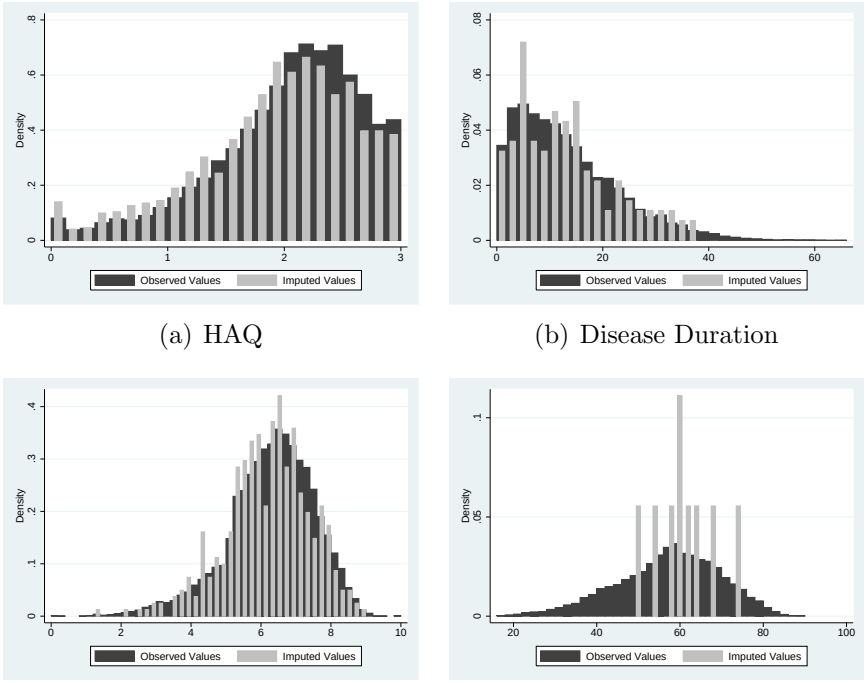
Now the imputed values all take reasonable values. However there is still one minor wrinkle: the variable `dm_grp` is being treated as a continuous predictor when it is used to impute the other variables. In other words, we assume that if the HAQ is x points higher in subjects who have had 2 previous DMARDs than in those who have only had one, then it must also be x points higher

[1]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_i disdur_i ovmean_i dascore_i, gen(nscore)
ice nscore1-nscore4 pgen_i dm_grp_i, saving("mi", replace) ///
    cmd(dm_grp_i: ologit) m(1) seed(999)
use mi, clear
invnscore age_i disdur_i ovmean_i dascore_i
tw histogram ovmean, width(0.125) color(gs4) ||             ///
   histogram ovmean_i if ovmean == ., gap(50) color(gs12)   ///
             width(0.125) legend(label(1 "Observed Values") ///
             label(2 "Imputed Values"))
graph export haq2.eps, replace
tw histogram disdur, width(2) color(gs4) ||                 ///
   histogram disdur_i if disdur == ., gap(50) color(gs12)   ///
             width(2) legend(label(1 "Observed Values")     ///
             label(2 "Imputed Values"))
graph export disdur2.eps, replace
tw histogram age, width(2) color(gs4) ||                    ///
   histogram age_i if age == ., gap(50) color(gs12)         ///
             width(2) legend(label(1 "Observed Values")     ///
             label(2 "Imputed Values"))
graph export age2.eps, replace
tw histogram dascore, width(0.2) color(gs4) ||              ///
   histogram dascore_i if dascore == ., gap(50) color(gs12) ///
             width(0.2) legend(label(1 "Observed Values")   ///
             label(2 "Imputed Values"))
graph export dascore2.eps, replace

tw histogram dascore if treated == 0, width(0.2) color(gs4) || ///
   histogram dascore_i if dascore == ., gap(50) color(gs12) ///
             width(0.2) legend(label(1 "Observed Values")   ///
             label(2 "Imputed Values"))
graph export dascore3a.eps, replace
tw histogram dascore if treated == 1, width(0.2) color(gs4) || ///
   histogram dascore_i if dascore == ., gap(50) color(gs12) ///
             width(0.2) legend(label(1 "Observed Values")   ///
             label(2 "Imputed Values"))
graph export dascore3b.eps, replace
```



(a) HAQ

(b) Disease Duration

(c) DAS

(d) Age

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

in those who have had 6 previous DMARDs compared to those who have had only 5. This may be a reasonable assumption in some cases, but let's suppose that it isn't: what can we do.

We can tell `ice` to treat `dm_grp` as a categorical variable when making imputations, but there are a number of steps. We need to

1. Create our own indicator variables
2. Tell `ice` not to impute these indicators directly, but calculate them from `dm_grp`
3. Tell `ice` not to include `dm_grp` in its predictions, but to use our indicators instead.

Step 1 can be performed via the command `tab dm_grp, gen(dm)` which will produce variables `dm1`, `dm2`, `dm3`, `dm4`, `dm5`, and `dm6` (since `dm_grp` has 6 levels. Although we have produced 6 variables, we will only need to include 5

in our regression models (if you enter all 6, you will get error messages about collinearity, but still get exactly the same results).

Step 2 can be achieved using the `passive` option: this tells `ice` that the variables should be calculated, not imputed, and how to calculate them. It will look like this:

```
passive(dm1:(dm_grp==1)  dm2:(dm_grp==2)  dm3:(dm_grp==3)  ///
dm4:(dm_grp==4)  dm5:(dm_grp==5))
```

The expression in brackets for each variable tells `ice` how to calculate that variable, the backslash character separates each variable.

Finally we need to tell ice to include `dm1-dm5` in our imputations, rather than `dm_grp`. We can do this with the `sub` (for substitute) option, which will take the form `sub(dm_grp:  dm1-dm5)`. Then, in any imputation equation in which `dm_grp` appears, the five variables `dm1-dm5` will be used instead. The full set of commands needed are shown in Listing 5.

Listing 5 Imputing from a categorical variable

```
restore
tab dm_grp, gen(dm)
preserve

ice age_i disdur_i ovmean_i dascore_i pgen_i dm_grp_i dm1-dm5,     ///
    saving("mi", replace) cmd(dm_grp_i: ologit) m(1)               ///
    passive(dm1:(dm_grp_i==1) \ dm2:(dm_grp_i==2) \                ///
    dm3:(dm_grp_i==3) \ dm4:(dm_grp_i==4) \ dm5:(dm_grp_i==5))     ///
     sub(dm_grp_i: dm1-dm5) seed(999)
```

Note that dichotomous variables (such as `pgen`) do not need this complicated treatment, provided that the two values that they take are 0 and 1. Only multicategory variables need to be treated in this way.

Note also that I've not shown the code for converting to and from normal scores in the above example. When we impute data for real, we will have to do that, but for and example to illustrate how categorical variables work that would be an unnecessary complication.

7 Imputation with interactions

Suppose that we want to include the interaction between gender and HAQ in our imputations. Note that this would only matter if we wanted the imputation equations to include the interaction: we could calculate interactions between imputed variables in subsequent analysis as normal. However, it may be that we think that the association between DAS and HAQ differs between men and women: if we include the interaction between gender and HAQ in our imputation models, we should achieve better imputations.

Including an interaction term is similar to including a categorical predictor: we need to

1. Create our own variables containing the interaction terms
2. Tell ice not to impute these indicators directly, but calculate them from pgen and ovmean

We do not need to use the `sub` option in this case, since we do not need to remove any variables from the imputation models. The code to perform these imputations is given in Listing 6.

Listing 6 Allowing for interactions in an imputation

```
restore
gen genhaq_i = pgen_i*ovmean_i
preserve

ice age_i disdur_i ovmean_i dascore_i pgen_i dm_grp_i dm1-dm5  ///
    genhaq_i, saving("mi", replace) cmd(dm_grp_i: ologit) m(5) ///
    sub(dm_grp_i: dm1-dm5) passive(dm1:(dm_grp_i==1) \         ///
    dm2:(dm_grp_i==2) \ dm3:(dm_grp_i==3) \ dm4:(dm_grp_i==4)  ///
    \ dm5:(dm_grp_i==5) \ genhaq: pgen_i*ovmean_i) seed(999)
```

8 Restricting Predictors in the imputation

So far, we have used every available predictor to impute every possible variable. This is generally sensible: adding superfluous variables to our model

does not have any detrimental effect on our predicted values (it can have detrimental effects on our coefficients, which is why we avoid it in general, but in this instance we are not concerned with the coefficients). However, there is an `eq` option that allows us to restrict the variables used in a particular imputation.

For example, entering `eq(ovmean_i: age_i dascore_i)` would tell ice to use only the variables `age_i` and `dascore_i` when imputing ovmean.

9 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) 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 complicated (see Section 7). Far easier would be to perform the imputations completely separately in the treated and control arms. The way to do this is illustrated in Listing 7.

I have set the number of imputations to 5, since this is the dataset that will be analysed, and analysis can only be performed if there are more than 1 imputation. In fact, 5 is the absolute bare minimum that would be considered acceptable: if at all possible, 10-20 should be used.

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. 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.

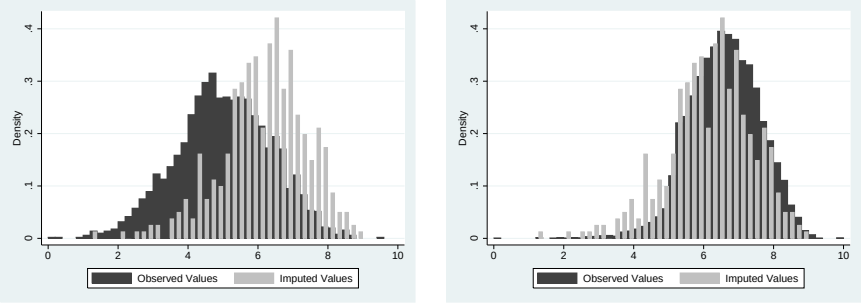
(Bear in mind that the distibutions of observed and imputed data do not need to be the same. For example, it may be that older subjects are less likely to complete a HAQ. Then the missing HAQs are likely to be higher than the observed HAQs. However, the associations between missingness and each of

Listing 7 Imputing in treated and untreated separately

```
restore
preserve
keep if treated == 0
nscore age_i disdur_i ovmean_i dascore_i, gen(nscore)
ice nscore1-nscore4 pgen_i dm_grp_i dm1-dm5 genhaq_i,            ///
    saving("miu", replace) cmd(dm_grp_i: ologit) m(5)            ///
    sub(dm_grp_i: dm1-dm5) passive(dm1:(dm_grp_i==1) \           ///
    dm2:(dm_grp_i==2) \ dm3:(dm_grp_i==3) \ dm4:(dm_grp_i==4) \  ///
    dm5:(dm_grp_i==5) \ genhaq: pgen_i*nscore3) seed(999)
use miu, clear
invnscore age_i disdur_i ovmean_i dascore_i
save, replace

restore
preserve
keep if treated == 1
nscore age_i disdur_i ovmean_i dascore_i, gen(nscore)
ice nscore1-nscore4 pgen_i dm_grp_i dm1-dm5 genhaq_i,            ///
    saving("mit", replace) cmd(dm_grp_i: ologit) m(5)            ///
    sub(dm_grp_i: dm1-dm5) passive(dm1:(dm_grp_i==1) \           ///
    dm2:(dm_grp_i==2) \ dm3:(dm_grp_i==3) \ dm4:(dm_grp_i==4) \  ///
    dm5:(dm_grp_i==5) \ genhaq: pgen_i*nscore3) seed(999)
use mit, clear
invnscore age_i disdur_i ovmean_i dascore_i

append using miu
tw histogram dascore if treated == 0, width(0.2) color(gs4) || ///
   histogram dascore_i if dascore == . & treated == 0, gap(50) ///
             color(gs12) width(0.2)                            ///
             legend(label(1 "Observed Values")                 ///
                    label(2 "Imputed Values"))
graph export dascore3c.eps, replace
tw histogram dascore if treated == 1, width(0.2) color(gs4) || ///
   histogram dascore_i if dascore == . & treated == 1, gap(50) ///
             color(gs12) width(0.2)                            ///
             legend(label(1 "Observed Values")                 ///
                    label(2 "Imputed Values"))
graph export dascore3d.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

the variables is only minor in this instance, so the distributions of imputed and observed data should be similar.)

10 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"[3]) to obtain your final estimate. Fortunately, as part of the ice package, there is a program `mim` which does exactly that. You simply precede whichever regression command you wanted use with `mim:` (provided there are at least 2 imputations, which is why we used `m(5)` in the `ice` command earlier). So, to obtain a propensity score from our imputed data, we would enter the command

```
drop _merge
xi: mim: logistic treated age_i disdur_i ovmean_i dascore_i  ///
                  i.pgen_i i.dm_grp_i
```

Note that the `xi` command has to come *before* the `mim` command. Also, `mim` merges datasets as part of its analysis, so that if you have a variable called `_merge` in your data, `mim` will fail, so rename (or drop) the variable before using `mim`[2].

One word of warning: I have found that predict does not work as expected after `mim: logistic`. It seems unable to produce predicted probabilities, and outputs the linear predictor from the logistic regression equation instead. If you wish to calculate propensity scores, you therefore need to calculate the probability yourself:

```
predict lpi, xb
gen pi = exp(lpi)/(1+exp(lpi))
```

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 ovmean dascore i.pgen i.dm_grp ///
             if _mj == 1
predict pc if _mj == 1
corr pc pi
```

If you enter the above commands, you will see that for the subjects with complete data, the propensity scores are almost identical (r = 0.9964) 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 8

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 dont need to lose such a substantial number is a definite bonus.

One minor point: ice creates variables called `_mi` and `_mj`, which represent the observation number and the imputation number respectively. If you

[2]There happened to be a `_merge` variable in the dataset I was working on for this example, and since I got caught out, I thought I'd mention the problem

Listing 8 Subjects with complete and missing data

```
. gen miss = pc == . if _mj == 1
(68120 missing values generated)

. tab treated miss, ro

+----------------+
| Key            |
|----------------|
|   frequency    |
| row percentage |
+----------------+

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

change these variables, mim may well produce gibberish.

It may be tempting to impute a single dataset, so that you dont need to worry about `mim`. 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.

11 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. 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 [3]. 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.

Basically, the variance of the parameter you are interested is

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

Where W is the mean variance of the imputations and B is the variance between imputations. So, if we had complete data, the variance would be W, 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.

Both of these parameters are produced, but not displayed, by `mim`. To display them, install the `mfracmiss` package from my homepage:

1. type `net from http://personalpages.manchester.ac.uk/staff/mark.lunt`
2. click on the blue `mfracmiss`
3. click on `click here to install.`

Now rerun the `mim: logistic` command above, then type `mfracmiss`, and you will get the output in Listing 9:

Listing 9 Missing information due to missing data

```
. mfracmiss
------------------------------------------------------------------------
  Variable   |  Increase in Variance        Missing Information
-------------+----------------------------------------------------------
       age_i |             0.0357                    0.0350
    disdur_i |             0.0072                    0.0071
    ovmean_i |             0.0553                    0.0537
   dascore_i |             0.0192                    0.0190
  _Ipgen_i_1 |             0.0175                    0.0173
_Idm_grp_i_2 |             0.0090                    0.0089
_Idm_grp_i_3 |             0.0358                    0.0351
_Idm_grp_i_4 |             0.0012                    0.0012
_Idm_grp_i_5 |             0.0043                    0.0043
_Idm_grp_i_6 |             0.0124                    0.0123
       _cons |             0.0231                    0.0228
------------------------------------------------------------------------
```

There are a few surprises here. First, there was no missing data for `pgen`, yet there is 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 considerable 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.

On the other hand, the missing information about HAQ is 5.4%, far less than the proportion of missing data. This is also because of the strong correlation between HAQ and age (and HAQ and DAS): although the data is missing, we can make a good guess as to what it should be from the values of the other variables, and therefore make good imputations. If it was unrelated to any other variable in the dataset, the proportion of missing information would be the same as the proportion of missing data.

This simple example shows how important it is to look at missing information, rather than missing data. The missing information may be more than the missing data (age, pgen) or less (HAQ), depending on the exact structure of your data. Inference may be possible even if a considerable amount of data is missing, provided that there is either plenty of information about the missing data in the dataset, or that the data in question has little effect on the parameter of interest (it is a weak confounder of the variable of primary interest, for example). On the other hand, you may have complete data on the outcome and exposure of interest, yet still have missing information on relative risk, since there is missing information about confounding.

12 Imputation Options

Here is a recap of all of the options to ice we have used, and what they do.

cmd Defines the type of regression command to be used in imputing a particular variable.

m The number of imputations to perform. More iterations give greater precision, but the law of diminishing returns applies. Five is generally enough (more may be needed if there is a substantial amount of missing data), but see the discussion in [4].

eq Allows the choice of variables from which to a impute a particular variable passive Identifies variables that can are functions of one or more other variables being imputed (indicators for a categorical variable or interaction effects).

sub Allows the replacement of one variable by one or more other variables in the imputation models. Used to replace a categorical variable with its indicators.

seed Sets the seed for the random number generator, so that the same values will be imputed if the analysis is reapeated.

There are a few other options to ice that may be useful:

cycles If there are multiple missing variables, they all need to be imputed. The imputation equations are first calculated on the complete-case data alone, then recalculated using the observed and imputed values. As the imputed values change, the imputation equations may change, leading to further changes in the imputed values. Eventually, this process should converge. The cycles option says how many iterations of this process should be performed before producing your final imputations. The default value is 10.

There are two sources of variation in the imputations: uncertainty about the coefficients in the imputation equations, and random variation in the population. Both are generally assumed to be normally distributed. If either of these assumptions are untrue, the imputed data may be unreliable. However, we have seen that we can avoid problems with a non-normal distribution in the population using `nscore`. There are also two options to ice to help with this problem:

boot If the regression coefficients are not normally distributed, we can use the `boot` option to get reasonable imputations.

match Rather than draw imputations from a theoretical (normal) distribution, only observed values of the variable are used. Thus the imputed values are drawn from the observed distribution. This is much slower than the default method, and the imputations may not be as reliable: see [4]. I would recommend the normal scored method outlined above in place of this option.

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

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