Multiple Imputation

Bill Rising
StataCorp LP

2010 Italian Stata Users Group Meeting
Bologna, Italia
November 11, 2010

Contents

1 Introduction **1**
- 1.1 Goals 1
- 1.2 MI Background 1

2 MI Basics **3**
- 2.1 A Simple Example 3

3 MI In More Depth **14**
- 3.1 Setup 14
- 3.2 Imputation 15
- 3.3 Estimation and Postestimation in MI 20
- 3.4 Data Management in MI 23

4 Conclusion **26**
- 4.1 Conclusion 26

1 Introduction

1.1 Goals

Goals

- Learn the mechanics of basic multiple imputation using Stata
- Learn about some of the extras `mi` has to offer

1.2 MI Background

Why Impute?

- Missing values cause observations to be omitted from analyses
- Omitted observations mean lost power
- Would like to regain some of the information from the non-missing variables in those observations

---

Past Methods

- Hot deck—picking a fixed value from another observation with the same covariates
  - Not necessarliy deterministic if there were many observations with the same covariate pattern
- Mean imputation—replacing with a mean
- Regression imputation—replacing with a single fitted value
- These methods all suffer from too little variation
  - Replaced missing values single good estimates

Multiple Imputation

- Draw many guesses at the missing values
  - Use the Bayes posterior predictive distribution of the missing values based (typically) on some sort of non-informative prior
  - Allows accounting for variation due to not being observed
- Estimate the model on each of the imputed datasets
- Pool the estimates using rules which account for variation from each dataset (within) and variation from the imputation (between)
- Originally developed by Rubin

What Missingness?

- For MI to work, the missingness must be unrelated to the data
  - Missing completely at random (MCAR)—missingness is completely independent the data generating process
    - Think of having complete data and randomly omitting some values
  - Missing at random (MAR)—given the observed data, the probability an observation is missing is unrelated to its value
- MCAR is often hard to believe; MAR is often easy to believe
  - MCAR means missingness causes no bias, just loss of power
  - MAR means missingness cause bias and loss of power
- These are sometimes called "ignorable", which really means "not non-ignorable"
  - "Non-ignorable" missing value processes depend on the values of the very missing values

---

When to Use Multiple Imputation

- Works best when casewise deletion would drop many of the observations
- Also works best when there is some correlation between the variables with missing data and the variables which are complete
- Remember that using multiple imputation means fitting a valid imputation model—so it requires the same care as fitting any other kind of model

---

2 MI Basics

2.1 A Simple Example

A Simple Example

- We'll run through a simple example first, using the control panel for `mi`
- This will get us familiar with the steps involved in multiple imputation without getting lost in the details
  - There are many details when doing MI in a careful fashion—this lesson is more about mechanics than technique
- Be aware that this example is intentionally very simple-minded

---

Modeling Energy Usage

- Open up the `autometric` dataset

```
. use autometric

(auto data with liters per 100km)
```

- This is simply the auto dataset with a gas mileage variable `lp100km` for liters of gas used per 100km
  - Used for physics' sake
- Try this model:

```
. regress lp100km weight displacement ///
.  gear_ratio length foreign

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  5,    68) =   46.46
       Model |  560.704497     5  112.140899           Prob > F      =  0.0000
    Residual |  164.146432    68  2.41391812           R-squared     =  0.7735
-------------+------------------------------           Adj R-squared =  0.7569
       Total |  724.850929    73  9.92946478           Root MSE      =  1.5537

------------------------------------------------------------------------------
     lp100km |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |   .0028799   .0008972     3.21   0.002     .0010896    .0046703
displacement |   .0028702   .0051781     0.55   0.581    -.0074625    .0132029
  gear_ratio |  -.6059845   .8019463    -0.76   0.452    -2.206243    .9942745
      length |   .0243027    .025469     0.95   0.343    -.0265199    .0751254
     foreign |   1.807019   .5657431     3.19   0.002     .6780958    2.935941
       _cons |  -.4167049   3.851043    -0.11   0.914    -8.101341    7.267931
------------------------------------------------------------------------------
```

- Store the estimates for comparison later

```
. estimates store complete
```

Modeling with Missing Observations

- Now open up the automiss dataset

```
. use automiss

(auto data with missing values and liters per 100km)
```

- This is the same dataset with some missing values for weight, displacement and length

```
. codebook

-------------------------------------------------------------------------------
make                                                             Make and Model
-------------------------------------------------------------------------------

                  type:  string (str17)

         unique values:  74                       missing "":  0/74

              examples:  "Cad. Deville"
                         "Dodge Magnum"
                         "Merc. XR-7"
                         "Pont. Catalina"

               warning:  variable has embedded blanks

-------------------------------------------------------------------------------
price                                                                     Price
-------------------------------------------------------------------------------

                  type:  numeric (int)

                 range:  [3291,15906]                 units:  1
         unique values:  74                       missing .:  0/74

                  mean:   6165.26
              std. dev:    2949.5

           percentiles:        10%       25%       50%       75%       90%
                              3895      4195    5006.5      6342     11385

-------------------------------------------------------------------------------
lp100km                                              Liters per 100 kilometers
-------------------------------------------------------------------------------

                  type:  numeric (float)

                 range:  [5.7,20.4]                   units:  .1
         unique values:  55                       missing .:  0/74

                  mean:    12.123
              std. dev:   3.15111

           percentiles:        10%       25%       50%       75%       90%
                               8.2       9.7      11.9      13.7      16.8

-------------------------------------------------------------------------------
rep78                                                       Repair Record 1978
-------------------------------------------------------------------------------
```

```
                  type:  numeric (byte)

                 range:  [1,5]                        units:  1
         unique values:  5                        missing .:  5/74

            tabulation:  Freq.  Value
                             2  1
                             8  2
                            30  3
                            18  4
                            11  5
                             5  .

--------------------------------------------------------------------------------
headroom                                                          Headroom (in.)
--------------------------------------------------------------------------------

                  type:  numeric (float)

                 range:  [1.5,5]                      units:  .1
         unique values:  8                        missing .:  0/74

            tabulation:  Freq.  Value
                             4  1.5
                            13  2
                            14  2.5
                            13  3
                            15  3.5
                            10  4
                             4  4.5
                             1  5

--------------------------------------------------------------------------------
trunk                                                      Trunk space (cu. ft.)
--------------------------------------------------------------------------------

                  type:  numeric (byte)

                 range:  [5,23]                       units:  1
         unique values:  18                       missing .:  0/74

                  mean:   13.7568
              std. dev:    4.2774

           percentiles:        10%       25%       50%       75%       90%
                                 8        10        14        17        20

--------------------------------------------------------------------------------
weight                                                             Weight (lbs.)
--------------------------------------------------------------------------------

                  type:  numeric (int)

                 range:  [1760,4840]                  units:  10
         unique values:  51                       missing .:  20/74

                  mean:   3029.44
              std. dev:   798.429

           percentiles:        10%       25%       50%       75%       90%
```

```
                        2050      2240      3190      3670      4080

------------------------------------------------------------------------------
length                                                            Length (in.)
------------------------------------------------------------------------------

                  type:  numeric (int)

                 range:  [147,233]                    units:  1
         unique values:  42                       missing .:  11/74

                  mean:   189.063
              std. dev:   21.5892

           percentiles:        10%       25%       50%       75%       90%
                               161       172       193       204       220

------------------------------------------------------------------------------
turn                                                         Turn Circle (ft.)
------------------------------------------------------------------------------

                  type:  numeric (byte)

                 range:  [31,51]                      units:  1
         unique values:  18                       missing .:  0/74

                  mean:   39.6486
              std. dev:   4.39935

           percentiles:        10%       25%       50%       75%       90%
                                34        36        40        43        45

------------------------------------------------------------------------------
displacement                                             Displacement (cu. in.)
------------------------------------------------------------------------------

                  type:  numeric (int)

                 range:  [79,425]                     units:  1
         unique values:  26                       missing .:  17/74

                  mean:   206.895
              std. dev:   95.0826

           percentiles:        10%       25%       50%       75%       90%
                                97       121       225       258       350

------------------------------------------------------------------------------
gear_ratio                                                          Gear Ratio
------------------------------------------------------------------------------

                  type:  numeric (float)

                 range:  [2.19,3.89]                  units:  .01
         unique values:  36                       missing .:  0/74

                  mean:   3.01486
              std. dev:   .456287

           percentiles:        10%       25%       50%       75%       90%
```

```
                          2.43      2.73     2.955      3.37      3.72

-------------------------------------------------------------------------------
foreign                                                                Car type
-------------------------------------------------------------------------------

                 type:  numeric (byte)
                label:  origin

                range:  [0,1]                        units:  1
        unique values:  2                        missing .:  0/74

           tabulation:  Freq.   Numeric  Label
                           52         0  Domestic
                           22         1  Foreign
```

- We still would like to predict gasoline usage (`lp100km`)

```
. regress lp100km weight displacement ///
.   gear_ratio length foreign

      Source |       SS       df       MS              Number of obs =      36
-------------+------------------------------           F(  5,    30) =   32.52
       Model |  365.765507     5  73.1531013           Prob > F      =  0.0000
    Residual |  67.4944682    30  2.24981561           R-squared     =  0.8442
-------------+------------------------------           Adj R-squared =  0.8183
       Total |  433.259975    35  12.3788564           Root MSE      =  1.4999

------------------------------------------------------------------------------
     lp100km |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |   .0020555   .0012717     1.62   0.116    -.0005416    .0046526
displacement |   .0003472   .0065069     0.05   0.958    -.0129417     .013636
  gear_ratio |  -1.593898   1.129048    -1.41   0.168    -3.899721    .7119245
      length |   .0529692   .0376232     1.41   0.169    -.0238676     .129806
     foreign |     1.6176   .8411427     1.92   0.064    -.1002423    3.335443
       _cons |    .376183   5.090549     0.07   0.942    -10.02011    10.77247
------------------------------------------------------------------------------
```

  - Note that the number of complete observations has dropped considerably, and that no coefficient is significant

- Store these estimates for comparison

```
. estimates store withmissing
```

Using the `mi` Control Panel

- The simplest way to learn about running an MI analysis is to use the control panel for `mi`
  - Either go to **Statistics** > **Multiple Imputation**, or
  - Type `db mi`
- The control panel is structured to match steps in MI
- Even if not used for commands, it is useful for memory cues

---

Start: A Careful Check of Missing Values

- From the status bar, we see that the data are not set up yet
- Click on the **Examine** button
- On the subpane *Tabulate missing values*, click on the **Go** –> button
  - In the submenu, select **Report pattern** and **Report frequencies**
  - Click **OK**
- The command issued is new in Stata 11: `misstable`

```
. misstable patterns, frequency

   Missing-value patterns
     (1 means complete)

              |   Pattern
    Frequency |  1  2  3  4
  ------------+-------------
           34 |  1  1  1  1
              |
           12 |  1  1  1  0
            9 |  1  1  0  1
            5 |  1  0  1  1
            4 |  1  1  0  0
            3 |  1  0  0  1
            2 |  0  1  1  1
            2 |  1  0  1  0
            1 |  0  0  1  0
            1 |  0  1  0  1
            1 |  0  1  1  0
  ------------+-------------
           74 |

  Variables are  (1) rep78  (2) length  (3) displacement  (4) weight
```

Something about `misstable`

- If you like, play around with other output for `misstable`
- Why the detail?
  - If the missingness is nested, it makes it easier to impute variables of different kinds, as we'll see
- Of course, nested missing values is rare in real-life data, unless the missingness is due to dropouts in studies
  - So the missingness for earlier observations is nested in that of later observations

---

Setting Up the Dataset for MI

- Click the **Setup** button
- Click the < *Choose style* > combo box
- Four different styles are available—choose *Marginal long*
  - All styles will work; for what we are doing the style is not important—we'll come back to this when looking at details
- Click the **Submit** button

```
. mi set mlong
```

Things to Note

- Stata created three variables in this case: `_mi_miss`, `_mi_m`, and `_mi_id`
  - These are for tracking the imputed datasets—more later

Registering Variables I

- Stata needs to watch over variables when imputing:
  - **Imputed** variables are variables for which we want to impute values
  - **Passive** variables are variables derived from imputed variables
  - **Regular** variables are variables which have no missing values, and which
- In general, it helps to register all variables
  - This is not necessary, however

Registering Variables II

- Register `weight`, `displacement`, and `length` as imputed

```
. mi register imputed ///
.  weight length displacement

(38 m=0 obs. now marked as incomplete)
```

- We can register the others (other than `rep78`) as regular

```
. mi register regular ///
.  price headroom trunk turn ///
.  gear_ratio foreign lp100km
```

- We have no passive variables

Imputing in the Control Panel I

- To do imputations in the Control Panel,
  - Select the **Reset # of imputations** radio button
  - Type in some number of imputatations—here use 20

```
. mi set M=20

(20 imputations added; M = 20)
```

- Now click the **Impute** button
- Here we would like to use multivariate normal regression
  - Select the *Multivariate normal regression* choice
  - Click the **Go** –> button

Imputing in the Control Panel II

- In the submenu, put `weight length displacement` in the *Imputed variables* field
- Put `price headroom trunk turn gear_ratio lp100km` in the *Independent variables* field
- Check the **Replace imputed...** checkbox
- Enter a random seed, say 3593
- Click the **OK** button

```
. mi impute mvn ///
.  weight length displacement = ///
.  price headroom trunk turn gear_ratio lp100km, ///
.  replace rseed(3593)

Performing EM optimization:
  observed log likelihood = -703.46284 at iteration 21

Performing MCMC data augmentation ...

Multivariate imputation                 Imputations =       20
Multivariate normal regression                added =        0
Imputed: m=1 through m=20                   updated =       20

Prior: uniform                           Iterations =     2000
                                            burn-in =      100
                                            between =      100

               |              Observations per m
               |----------------------------------------------
      Variable |   complete   incomplete   imputed |     total
---------------+-----------------------------------+----------
        weight |         54           20        20 |        74
        length |         63           11        11 |        74
  displacement |         57           17        17 |        74
--------------------------------------------------------------
(complete + incomplete = total; imputed is the minimum across m
 of the number of filled in observations.)
```

What Happened?

- Stata imputed 20 different datasets
- It treated the above variables (all of them) as being jointly multivariate normal
- It then used the Monte Carlo Markov Chain (MCMC) data augementation methods to pick values from the posterior predictive distribution for the multivariate normal
- Try looking at how `mi` sees the dataset

```
. mi describe

  Style:  mlong

  Obs.:   complete           36
          incomplete         38  (M = 20 imputations)
          ---------------------
          total              74

  Vars.:  imputed:  3; weight(20) length(11) displacement(17)

          passive: 0

          regular: 7; price headroom trunk turn gear_ratio foreign lp100km

          system:  3; _mi_m _mi_id _mi_miss

         (there are 3 unregistered variables; make rep78 _est_withmissing)
```

- Stata only imputed values for observations with system-missing values
  - Non-system missing values are considered to be structural missing values which should not be imputed
  - System missing are called *soft*; structural missing values are called *hard* missing values

---

Estimating a model

- Estimating a model is now as simple a estimating a typical Stata model
- We could select the –> *Linear regression* and click the **Go** –> button to use another dialog
- Since we know Stata's `regress` command, we'll just put our regression command from earlier in the *Estimation command* field, and click **Submit**

```
. mi estimate: regress lp100km ///
. weight displacement gear_ratio length foreign

Multiple-imputation estimates                    Imputations     =         20
Linear regression                                Number of obs   =         74
                                                 Average RVI     =     0.1706
                                                 Complete DF     =         68
DF adjustment:   Small sample                    DF:     min     =      33.47
                                                         avg     =      47.27
                                                         max     =      61.28
Model F test:       Equal FMI                    F(   5,   63.9) =      35.80
Within VCE type:          OLS                    Prob > F        =     0.0000

------------------------------------------------------------------------------
     lp100km |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
```

```
      weight |   .0027651   .0011953     2.31   0.027      .000336    .0051943
displacement |   .0011164   .0054778     0.20   0.839    -.0099087    .0121416
  gear_ratio |  -.6147166   .9248199    -0.66   0.509    -2.464591    1.235157
      length |   .0329809   .0357376     0.92   0.363    -.0396889    .1056506
     foreign |   1.689288   .6080325     2.78   0.007     .4735638    2.905013
       _cons |  -1.309322   4.368096    -0.30   0.766    -10.08922    7.470574
------------------------------------------------------------------------------
```

What Happened?

- Stata estimated the model for all 20 imputed datasets
- The results were then combined
  - The results are weighted according to how much variation there is between imputations *vs.* how much variation there is according to the linear model
  - These rules are often called Rubin's rules
- We can see how much variation there is with

```
. mi estimate, vartable nocitable

Multiple-imputation estimates                     Imputations     =         20

Variance information
------------------------------------------------------------------------------
             |        Imputation variance                            Relative
             |    Within   Between     Total       RVI       FMI   efficiency
-------------+----------------------------------------------------------------
      weight |   9.4e-07   4.7e-07   1.4e-06   .527225    .35328      .982643
displacement |   .000024   6.1e-06    .00003   .272386   .217839      .989225
  gear_ratio |   .791364   .060884   .855292   .080782   .075288       .99625
      length |   .000828   .000428   .001277   .543305   .360331      .982302
     foreign |   .347025   .021598   .369704    .06535   .061713      .996924
       _cons |   15.4553    3.4523   19.0803   .234541   .193043       .99044
------------------------------------------------------------------------------
Note: FMIs are based on Rubin's large-sample degrees of freedom.
```

  - The relative efficiencies are estimates of efficiency relative to an infinite number of imputations

---

Estimates Table and `mi`

- Now store these results

```
. estimates store mi
```

- We would now like to make an estimates table

```
estimates table complete withmissing mi, b(%9.6f) se(%9.6f)
```

- This fails, because `mi` is being protective—it does not want to post `e(b)` or `e(V)`—and these are needed by `estimates table`

Getting Estimates Table to Work

- To force `mi estimate` to give us the (dangerous) values, add an `post` option to the prefix command

```
. mi estimate, post: regress lp100km ///
. weight displacement gear_ratio length foreign

Multiple-imputation estimates                    Imputations     =         20
Linear regression                                Number of obs   =         74
                                                 Average RVI     =     0.1706
                                                 Complete DF     =         68
DF adjustment:   Small sample                    DF:     min     =      33.47
                                                         avg     =      47.27
                                                         max     =      61.28
Model F test:       Equal FMI                    F(   5,   63.9) =      35.80
Within VCE type:          OLS                    Prob > F        =     0.0000

------------------------------------------------------------------------------
     lp100km |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |   .0027651   .0011953     2.31   0.027      .000336    .0051943
displacement |   .0011164   .0054778     0.20   0.839    -.0099087    .0121416
  gear_ratio |  -.6147166   .9248199    -0.66   0.509    -2.464591    1.235157
      length |   .0329809   .0357376     0.92   0.363    -.0396889    .1056506
     foreign |   1.689288   .6080325     2.78   0.007     .4735638    2.905013
       _cons |  -1.309322   4.368096    -0.30   0.766    -10.08922    7.470574
------------------------------------------------------------------------------
```

- And then re-store the estimates

```
. estimates store mi
```

- Now for the table

```
. estimates table complete withmissing mi, ///
.  b(%9.6f) se(%9.6f)

--------------------------------------------------
    Variable | complete    withmis~g      mi
-------------+------------------------------------
      weight |  0.002880    0.002056    0.002765
             |  0.000897    0.001272    0.001195
displacement |  0.002870    0.000347    0.001116
             |  0.005178    0.006507    0.005478
  gear_ratio | -0.605984   -1.593898   -0.614717
             |  0.801946    1.129048    0.924820
      length |  0.024303    0.052969    0.032981
             |  0.025469    0.037623    0.035738
     foreign |  1.807019    1.617600    1.689288
             |  0.565743    0.841143    0.608032
       _cons | -0.416705    0.376183   -1.309322
             |  3.851043    5.090549    4.368096
--------------------------------------------------
                                      legend: b/se
```

- Phew!

How Did We Do?

- Note that the standard errors are in the order

  complete $<$ imputed $<$ missing

- If we check significance, we'll see that the imputed dataset did better, also

```
. estimates table complete withmissing mi, ///
.  b(%9.6f) star

------------------------------------------------------------
    Variable |   complete     withmissing        mi
-------------+----------------------------------------------
      weight |  0.002880**     0.002056       0.002765*
displacement |  0.002870       0.000347       0.001116
  gear_ratio | -0.605984      -1.593898      -0.614717
      length |  0.024303       0.052969       0.032981
     foreign |  1.807019**     1.617600       1.689288**
       _cons | -0.416705       0.376183      -1.309322
------------------------------------------------------------
                   legend: * p<0.05; ** p<0.01; *** p<0.001
```

- Here, things seem to have worked just fine

General Flow:

- Inspect the data: `misstable`
- `mi set` the data
- Register variables: `mi register`
- Impute values
  - Choose an Imputation model
    - This should be done with the same care as choosing the model of interest
  - Compute passive variables using `mi passive:`
    - Here is where passive variables should be generated using `mi`
- Estimate using `mi estimate:`

---

3 MI In More Depth

3.1 Setup

Some Notation and Terminology

- The *original data* are just what you would think—the original dataset
- An *imputation* is an entire dataset where missing values have been filled with imputed values
  - Our example had 20 imputations
- The imputation datasets are $m = 0, 1, 2 \ldots, M$, where M is the number of imputations
  - $m = 0$ is the original dataset
- *Complete* observations are observations without any imputed values
  - *imputed* observations have at least one imputed value

Data Styles

- mi understands 4 data styles:
  - Full, long, and separate (flongsep stores a full copy of each imputation dataset from 0 through M
    - Needed only for very large datasets or imported datasets
  - Full, long (flong), keeps a full copy of each imputation dataset stacked up in one Stata dataset
    - Once again best for imported datasets
  - Marginal long (mlong) keeps just imputed observations
    - Useful when manipulating variables
  - Wide (wide) stores copies of imputed variables
    - Useful when manipulating observations

Looking at Our Example

- Our dataset is in mlong form
  - _mi_id marks the observation in the original data for which this is the imputed observation
  - _mi_miss marks the observations in the original dataset which have missing values in the imputed variables
  - _mi_m holds the imputation dataset number
- These can be used for looking at the imputed values
- We could have just as easily specified that the data should be wide

---

Which Data Style?

- The mi convert command can convert between forms of datasets, so the type of dataset is not immediately critical
- It is best, if the dataset can fit in memory, to use either mlong or wide form
- Here, we can convert to wide form

```
. mi convert wide, clear
```

- Note that we now have prefixed variables corresponding to each imputation and imputed variable
- Either wide or mlong is preferable to the other styles

3.2 Imputation

Univariate Imputation Methods

- Univariate Methods
  - linear regression (mi impute regress)—for continuous variables
  - predictive mean matching (mi impute pmm)—for continuous variables when normal errors in linear regression are suspect
  - logistic regression (mi impute logit)—for 0/1 variables
  - multinomial logistic regression (mi impute mlogit)—for nominal variables
  - ordinal logistic regression (mi impute ologit)—for ordinal variables
- The general syntax is mi impute *method model*
  - The model is specified just like the corresponding estimation command

---

An Example

- First allow rep78 to be imputed by registering it:

```
. mi register imputed rep78
```

- An example with our present dataset:

```
. mi impute ologit rep78 ///
.  headroom trunk turn gear_ratio foreign, ///
.  replace rseed(30103)

Univariate imputation                   Imputations =       20
Ordered logistic regression                   added =        0
Imputed: m=1 through m=20                   updated =       20

               |              Observations per m
               |----------------------------------------------
      Variable |   complete   incomplete   imputed |     total
---------------+-----------------------------------+----------
         rep78 |         69            5         5 |        74
--------------------------------------------------------------
(complete + incomplete = total; imputed is the minimum across m
 of the number of filled in observations.)
```

- We can investigate this
  - We make the data mlong

```
. mi convert mlong, clear
```

  - We see where it is missing

```
. list _mi_id if _mi_m==0 & missing(rep78)

     +--------+
     | _mi_id |
     |--------|
  3. |      3 |
 10. |     10 |
 53. |     53 |
 57. |     57 |
 63. |     63 |
     +--------+
```

  - Look at the various imputed values

```
. tab _mi_id rep78 ///
.  if _mi_id & inlist(_mi_id,3,10,53,57,63)

           |                   Repair Record 1978
    _mi_id |         1          2          3          4          5 |     Total
-----------+-------------------------------------------------------+----------
         3 |         0          1         10          7          2 |        20
        10 |         2          4          8          3          3 |        20
        53 |         1          0          2         13          4 |        20
        57 |         2          6         11          1          0 |        20
        63 |         1          5         13          1          0 |        20
-----------+-------------------------------------------------------+----------
     Total |         6         16         44         25          9 |       100
```

Were These Imputations Proper?

- These imputations were done sequentially after imputing the missing values for `weight`, `length`, and `displacement`
- We may use these imputed values in two situations:
  - We think that `rep78` is independent of the other imputed variables, or
  - We will not use `rep78` together with the other imputed variables in any model building
    * In our case, this is something reasonable, because `rep78` and `lp100km` would likely not be involved in each other's modeling

---

Multivariate Methods

- Multivariate normal regression (`mi impute mvn`)—for continuous data when there is no structure to missing values
  - `mi impute mvn` *imputed varlist* = *indepvarlist*
- Sequential univariate imputation (`mi impute monotone`) when missing observations are nested
  - Models are specified in order from least to most missing values

---

Example of Monotone Imputation I

- Save what we've done, if you like; other wise clear out

```
. clear
```

- Open up a specially constructed dataset

```
. use automono

(auto data with monotone missing values and lp100km)
```

- Take a look at it using`misstable` (method 1)

```
. misstable nested

     1.  foreign(4) -> rep78(5) -> weight(21)
```

  - This works because the missing observations are truly nested
- Using `misstable` (method 2)

```
. misstable patterns, freq bypat

   Missing-value patterns
     (1 means complete)

              |   Pattern
    Frequency |  1  2  3
  ------------+-------------
           53 |  1  1  1
              |
  1:          |
           16 |  1  1  0
  2:          |
            1 |  1  0  0
```

```
3:          |
          4 |  0  0  0
------------+-------------
         74 |

Variables are  (1) foreign  (2) rep78  (3) weight
```

Example of Monotone Imputation II

- We can use monotone imputation, starting with weight, then foreign, then rep78
- We'll do this via a do-file to save some typing

```
. do automono

. * start by using automono again
.
. use automono
(auto data with monotone missing values and lp100km)


.
. mi set mlong

. mi register imputed rep78 weight foreign
(21 m=0 obs. now marked as incomplete)

. mi register regular price headroom trunk length-gear_ratio lp100km


.
. mi impute monotone (pmm) weight (logit) foreign (ologit) rep78 ///
>   = lp100km trunk length turn displacement gear_ratio, ///
>   add(20) rseed(3443)

Conditional models:
       foreign: logit foreign lp100km trunk length turn displacement
                 gear_ratio
         rep78: ologit rep78 i.foreign lp100km trunk length turn displacement
                 gear_ratio
        weight: pmm weight i.rep78 i.foreign lp100km trunk length turn
                 displacement gear_ratio

Multivariate imputation                Imputations =       20
Monotone method                              added =       20
Imputed: m=1 through m=20                  updated =        0

       foreign: logistic regression
         rep78: ordered logistic regression
        weight: predictive mean matching

              |              Observations per m
              |----------------------------------------------
      Variable |   complete   incomplete   imputed |     total
---------------+-----------------------------------+----------
       foreign |         70            4         4 |        74
         rep78 |         69            5         5 |        74
        weight |         53           21        21 |        74
--------------------------------------------------------------
(complete + incomplete = total; imputed is the minimum across m
 of the number of filled in observations.)
```

```
.
. * saving the results for later
. mi estimate, saving(automono_fit): ///
>   regress lp100km weight displacement gear_ratio length foreign

Multiple-imputation estimates                     Imputations     =         20
Linear regression                                 Number of obs   =         74
                                                  Average RVI     =     0.1366
                                                  Complete DF     =         68
DF adjustment:   Small sample                     DF:     min     =      37.13
                                                          avg     =      49.08
                                                          max     =      60.27
Model F test:       Equal FMI                     F(   5,   64.4) =      38.03
Within VCE type:          OLS                     Prob > F        =     0.0000

------------------------------------------------------------------------------
     lp100km |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |   .0030823   .0013547     2.28   0.029     .0003378    .0058269
displacement |   .0007368   .0064655     0.11   0.910    -.0122319    .0137056
  gear_ratio |   -.581057   .8499161    -0.68   0.497    -2.280985    1.118871
      length |   .0252698   .0328691     0.77   0.447     -.041158    .0916977
     foreign |    1.72751   .6428406     2.69   0.010     .4355583    3.019461
       _cons |  -.8242788   4.226009    -0.20   0.846    -9.292324    7.643767
------------------------------------------------------------------------------

.
.
end of do-file
```

Notes on Monotone Imputation

- `mi` is smart enough to learn from the method to know which imputed variables are categorical
- `monotone` includes each imputed variable for imputing the following variables

---

Notes on `pmm`

- `pmm` was used for `weight` to illustrate its use
  - We could have used linear regression
- `pmm` chooses values from the posterior predictive distribution
- It then picks the observed value corresponding to the closed predicted value
  - The number of neighbors from which the value is chosen can be changed from 1 using the `knn` option

---

Note on Multivariate Normal Imputation

- This is the method which is commonly used if there is no pattern to the missing values
- It assumes a multivariate normal distribution for the imputed variables, however

---

What if a Mixture of Models is Needed?

- It is possible to try multivariate normal methods and round values
- If the missing patterns are nearly monotone, it is possible to delete values to force monotonicity, and then run the imputations
- It is possible to use the chained equation methods
  - This is a user-written command: `ice`
  - It complements Stata's capabilities, and should be considered as a possibility

---

Other Notes on Imputation

- If you think there will be interaction terms involving a complete variable in the dataset, separate imputations should be run on the levels
- For example: if `foreign` were complete, we would need

  `mi impute` ... `if foreign==0`

  `mi impute` ... `if foreign==1`

  as two separate imputations

3.3 Estimation and Postestimation in MI

What Estimation Commands Work?

- Many of Stata's estimation commands work as above, with

  `mi estimate` *estimation command* ...

- To see which commands work, look at `help mi estimation`
- This is a large subset of the estimation commands
  - These include some of the special-form datasets, such as survival time datasets and complex survey datasets
  - These dataset structures must be set up before using the`mi` version of the command—see `help mi XXXset`

What about User-Written Estimation Commands?

- User-written estimation commands can be used via

  `mi estimate, cmdok:` ...

- Warning: this is something that must be done carefully, because the user-written command must understand how to combine the results from the various imputations

---

What Postestimation Commands Work?

- Because MI estimation is a much different beast than typical estimation, the suite of postestimation commands is different: try `help mi postestimation`
- `mi test` corresponds to `test`, but only for testing coeffecients being zero
- `mi testtransform` is used for testing linear and non-linear hypotheses

An Example of `mi test`

- Look at the last model

```
. mi estimate

Multiple-imputation estimates                    Imputations     =         20
Linear regression                                Number of obs   =         74
                                                 Average RVI     =     0.1366
                                                 Complete DF     =         68
DF adjustment:   Small sample                    DF:     min     =      37.13
                                                         avg     =      49.08
                                                         max     =      60.27
Model F test:       Equal FMI                    F(   5,   64.4) =      38.03
Within VCE type:          OLS                    Prob > F        =     0.0000

------------------------------------------------------------------------------
     lp100km |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |   .0030823   .0013547     2.28   0.029      .0003378    .0058269
displacement |   .0007368   .0064655     0.11   0.910     -.0122319    .0137056
  gear_ratio |   -.581057   .8499161    -0.68   0.497     -2.280985    1.118871
      length |   .0252698   .0328691     0.77   0.447      -.041158    .0916977
     foreign |    1.72751   .6428406     2.69   0.010      .4355583    3.019461
       _cons |  -.8242788   4.226009    -0.20   0.846     -9.292324    7.643767
------------------------------------------------------------------------------
```

- If we would like to test if the coefficients of `displacement`, `gear_ratio`, and `length` are simultaneously zero, we would use

```
. mi test displacement gear_ratio length

note: assuming equal fractions of missing information

 ( 1)  displacement = 0
 ( 2)  gear_ratio = 0
 ( 3)  length = 0

       F(  3,  62.9) =    0.34
            Prob > F =    0.7952
```

An Example of a Linear Test I

- When running a test on a non-MI model, we would use, for example
  `test weight == displacement`
- We could estimate at linear combinations using
  `lincom weight - displacement`

- These don't work directly after `mi estimate`, because more information is needed from the fitting of the model
  - This is different than typical tests which just need the model, coefficients and the variance-covariance matrix

---

An Example of a Linear Test II

- We would need to re-estimate the model, but we saved the estimation results in `automono_fit.ster` above
  - Stata added the `ster` extension
- So, now we estimate the difference to run the test (like in lincom)

```
. mi estimate (lintest: _b[weight] - _b[displacement]) using automono_fit

Multiple-imputation estimates                     Imputations     =         20
Linear regression                                 Number of obs   =         74
                                                  Average RVI     =     0.1366
                                                  Complete DF     =         68
DF adjustment:   Small sample                     DF:     min     =      37.13
                                                          avg     =      49.08
                                                          max     =      60.27
Model F test:       Equal FMI                     F(   5,   64.4) =      38.03
Within VCE type:          OLS                     Prob > F        =     0.0000

------------------------------------------------------------------------------
     lp100km |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |   .0030823   .0013547     2.28   0.029     .0003378    .0058269
displacement |   .0007368   .0064655     0.11   0.910    -.0122319    .0137056
  gear_ratio |   -.581057   .8499161    -0.68   0.497    -2.280985    1.118871
      length |   .0252698   .0328691     0.77   0.447     -.041158    .0916977
     foreign |    1.72751   .6428406     2.69   0.010     .4355583    3.019461
       _cons |  -.8242788   4.226009    -0.20   0.846    -9.292324    7.643767
------------------------------------------------------------------------------

Transformations                                   Average RVI     =     0.2290
                                                  Complete DF     =         68
DF adjustment:   Small sample                     DF:     min     =      48.96
                                                          avg     =      48.96
Within VCE type:          OLS                             max     =      48.96

      lintest: _b[weight] - _b[displacement]

------------------------------------------------------------------------------
     lp100km |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lintest |   .0023455   .0074491     0.31   0.754    -.0126244    .0173155
------------------------------------------------------------------------------
```

and then can run the test, if we like

```
. mi testtransform lintest

note: assuming equal fractions of missing information

      lintest: _b[weight] - _b[displacement]

 ( 1)  lintest = 0
```

```
         F(  1,  49.0) =    0.10
              Prob > F =    0.7542
```

- Non-linear tests work the same way

3.4 Data Management in MI

Data Management in MI

- Data management in MI requires more care than typical data management
  - Passive variables need to be kept consistent
  - Simple manipulations of the dataset have more ramificiations
  - Reproducibility can be made more difficult
- Good News: Stata has tools for making the data management simpler—but it still requires care

---

Working with Internal Datasets

- If you are working with your own internal datasets, where you have access to the whole dataset, it is best to do all your data mangement **before** doing imputation
- You will want to separate your overall data management from your imputation and creation of passive variables,
  - Keep them as 2 separate do-files so the separation is clear
- Every time you redo any data management, redo the imputation
- This will allow reproducibility to work properly

Importing Datasets

- Stata has several tools for working with imported datasets
  - `mi import flong`, `mi import flongsep`, and `mi import wide` correspond to the different `mi` data types
  - `mi import ice` imports MI datasets created by `ice`
  - `mi import nhanes1` imports MI datasets from NCHS
- The datasets must be Stata datasets before they can be imported
  - `ice` naturally makes Stata datasets
- We'll not cover this here

Basic Tools for Data Management

- `mi describe` shows the `mi` structure of the dataset

```
. mi describe

  Style:  mlong
          last mi update 02nov2010 10:48:59, 1 second ago

  Obs.:   complete           53
          incomplete         21  (M = 20 imputations)
          ---------------------
          total              74

  Vars.:  imputed:  3; rep78(5) weight(21) foreign(4)

          passive: 0

          regular: 8; price headroom trunk length turn displacement gear_ratio
                   lp100km

          system:  3; _mi_m _mi_id _mi_miss

         (there is one unregistered variable; make)
```

  - `mi query` gives a much shorter description

```
. mi query

data mi set mlong, M = 20
last mi update 02nov2010 10:48:59, 1 second ago
```

- `mi varying` can be used to spot mistakes—they look for how constant variables are across the imputed datasets

```
. mi varying

           Possible problem   variable names
  -----------------------------------------------------------------------------
          imputed nonvarying:  (none)
          passive nonvarying:  (none)
        unregistered varying:  (none)
  -----------------------------------------------------------------------------
```

  - Constant imputed variables have not yet been imputed,
  - Constant passive variables' imputed bases have not been imputed
  - Varying unregistered variables change across imputations, and yet are supposedly not imputed or passive
    - Such variables are called "super varying" and should not occur often

- Of course, `mi varying` can work only if variables are registered

Specialized Tools for Data Management

- Stata has a few tools for working on the repeating datasets
  - `mi xeq:` *command* runs *command* on each of the imputed datasets
  - There are several commands which have been extended to work with `mi`: look at `help mi`
  - `mi update` is used to check consistency and update changes

Example of `mi update`

- Try dropping an observation—recall that `rep78` is missing in observation 3

```
. drop in 3

(1 observation deleted)
```

- Dropping the 3rd observation means that some imputed observations will no longer be needed

```
. mi update

(system variable _mi_id updated due to changed number of obs.)
(20 m>0 marginal obs. dropped due to dropped obs. in m=0)
```

- `mi update` is run after every `mi` command—so you need to run it only after manipulating data directly
- Be careful—`mi update` keeps track of observations and other bookkeeping, but it cannot keep track of needed changes in passive variables or changes to observations which could affect imputations
  - Hence it should really only be used when manipulating imported data

---

A Fancy Tool

- There is a pair of commands which allows temporarily stepping out of the `mi` realm while doing data management
- Suppose you have saved your `mi` dataset in `doremi`, and you would like to manipulate it
  - `use doremi`
`mi extract 0`
...
`mi replace0 using doremi, id(`*keyvarlist*`)`
will allow changes to the main dataset to be propagated through the imputed datasets

---

Example of the Fancy Tool

- Open the `automono2` dataset

```
. use automono2, clear

(auto data with monotone missing values and lp100km)
```

- Extract the non-imputed dataset

```
. mi extract 0
```

- Change some data (output omitted)

```
. replace rep78 = 2 if make=="AMC Spirit"
. drop if price>=14000
```

- Rebuild

```
. mi replace0 using automono2, id(make)

(system variable _mi_id updated due to changed number of obs.)
(imputed variables updated in 20 obs. in m>0 in order to match m=0 data)
```

- Peek

```
. list make _mi_m rep78 if make=="AMC Spirit"

     +----------------------------+
     | make         _mi_m   rep78 |
     |----------------------------|
  3. | AMC Spirit       0       2 |
 73. | AMC Spirit       1       2 |
 93. | AMC Spirit       2       2 |
113. | AMC Spirit       3       2 |
133. | AMC Spirit       4       2 |
     |----------------------------|
153. | AMC Spirit       5       2 |
173. | AMC Spirit       6       2 |
193. | AMC Spirit       7       2 |
213. | AMC Spirit       8       2 |
233. | AMC Spirit       9       2 |
     |----------------------------|
253. | AMC Spirit      10       2 |
273. | AMC Spirit      11       2 |
293. | AMC Spirit      12       2 |
313. | AMC Spirit      13       2 |
333. | AMC Spirit      14       2 |
     |----------------------------|
353. | AMC Spirit      15       2 |
373. | AMC Spirit      16       2 |
393. | AMC Spirit      17       2 |
413. | AMC Spirit      18       2 |
433. | AMC Spirit      19       2 |
     |----------------------------|
453. | AMC Spirit      20       2 |
     +----------------------------+
```

- This works fine—but take care if you are working on an internal dataset which could change

---

Data Management in General

- Use `mi` versions of data management commands if they are available
- Use the `mi xeq:` prefix command
- Try checking `mi update` for consistency
- Take great care

---

4 Conclusion

4.1 Conclusion

Conclusion

- Multiple Imputation allows gaining back some information lost by non-response
- Stata has tools built for working with a wide variety of estimation commands and specialized data structures—all of these are `mi` commands

---

Index

M

`mi`, 3–26
    control panel, 7
    data management tools, 23–26
    data styles, 15
    `estimates table`, 12, 13
    estimation, 11, 12
    imputation methods
        multivariate, 17–20
        univariate, 15–17
    imputed datasets, 14
    imputing, 10, 11
    postestimation, 20–23
    registering variables, 9
    workflow, 14
missing at random (MAR), 2
missing completely at random (MCAR), 2
missing value structure, *see* `misstable`
`misstable`, 8
multiple imputation
    concept, 1, 2
    implementation, *see* `mi`