Vanderbilt University
Leadership, Policy and Organizations
Class Number 9953
Spring 2021

Practicum: Dealing with Missing Data Using Multiple Imputation

The Missing Data Problem

"The problem with missing data is that it's missing"–Nathaniel Beck

When dealing with a dataset with severe missing data, we are living in the land of bad options. We want all of our approaches to guarantee first that our estimates are unbiased, and second, that we can use as much of the actual data as possible.

All of the techniques described below are evaluated based first on whether they introduce bias into the estimates and second on whether they provide more efficiency. The goal of techniques for missing data is not to literally replace the data that isn't there, but to come up with the least bad way to use that data that is there to estimate the parameters of interest.

There are three types of missing data:

1. **Missing Completely at Random (MCAR)**: In this case, the data are missing in a pattern that is independent of other variables in the dataset. For instance, a question on a survey might have only been asked of a random subset of respondents. Let Z be a variable with missing data, and X be a vector of always observed variables. If R_z is an indicator variable which 1 in if Z is missing and 0 otherwise, then the MCAR assumption can be formalized as:

$$Pr(R_z = 1|X, Z) = Pr(R_Z = 1)$$

2. **Missing at Random (MAR)** In this case, the data are missing in such a way that they are dependent on at least some of the other available data, including the data missing from the variable itself. Importantly, the missing data are dependent on the *observed* values of the other data. The relationship between the missing data and the available data is not deterministic. Using the same notation as before, the MAR assumption can be formalized as:

$$Pr(R_z = 1|X, Z) = Pr(R_z = 1|X)$$

   It's important to note that the missing information in Zcan depend on$X$, but not on Z alone, after you have conditioned on X. This actually shares some characteristics with the conditional independence assumption used in causal inference.

   The "at random" part of missing at random does not mean *completely* random. It means that the indicator variable for the missing data is a random variable whose conditional mean is governed by other data in the dataset.

3. **Not Missing at Random (NMAR)** In this case, the data are missing as a function of the underlying variable which is missing data. For instance, if respondents consistently refused to report their own income above a certain income level. Data not missing at random are dependent on the values of the unobserved data. This is also called "non-ignorable" missingness.

   Formally, if we are missing data for a variable Z for all $Z > A$,

$$Pr(R_z = 1|Z > A) = 1$$

If we have MCAR data, it's just a subsample of the sample, and there's no really problem. If we have NMAR data, there's nothing to be done.

Most of the time, we're dealing with data that's MAR—we know it's not missing completely at random, but we're not stuck with data that has a perfect relationship with values of the underlying variable, and can be reasonably predicted from other observed data.

The following is a list of commonly used options for missing data and reasons not to use any of them except multiple imputation (and its variants).

Casewise Deletion (Complete-Case Analysis)

You should be familiar with casewise deletion at this point. This is the default option in Stata and other similar statistical programs. Casewise deletion is acceptable only when the data are MCAR. Essentially, the analyst is completing the analysis on a random sample of a random sample, which is not problematic. Other names for this are listwise deletion or complete case analysis.

When do we know this is true? Generally, only when the sample design results in missing data patterns. For instance, if a portion of a questionnaire was applied only to a random subset of the respondents, then casewise deletion based on missing data is appropriate.

In any circumstance where the missing data were not generated intentionally, you can not safely assume MCAR, and therefore can not use casewise deletion. In any case where MCAR is not exactly established, casewise deletion results in both bias and a loss of efficiency. Bias results because the estimates are derived from a non-random sample, loss of efficiency because casewise deletion by design throws out a large amount of data. You should only use casewise deletion in cases when a very small amount of data are missing, and you are quite uncertain about how to model the missing data (these circumstances are vanishingly rare in practice).

However, even though this is an *inefficient* method, it does not appear to bias results much. Allison writes:

> Somewhat surprisingly, listwise deletion is very robust to violations of MCAR (or even MAR) for predictor variables in a regression analysis. Specifically, so long as missingnenss on the predictors does not depend on the dependent variable, listwise delation will yield approimately unbiased estimates of regression coefficients

(Little, 1992). And this holds for virtually any kind of regression–linear, logistic, Poisson, Cox, etc. (Allison 2008 p. 75)

Weighted Casewise Deletion

Weighted casewise deletion attempts to make up for the loss of data from casewise deletion by re-weighting the sample to account for the patterns of missing data. Each complete case is assigned a new weight after deletion is complete, resulting in a weighted sample that is supposed to represent the overall population under study.

The same conditions must apply for weighted casewise deletion to be appropriate as must apply when casewise deletion is appropriate. Otherwise, the bias caused will be even worse, since the technique assumes a subset of a random sample as the basis for re-weighting.

Surprisingly, this is *exactly* what NCES does with many of its longitudinal datasets. The "panel weights" applied to various combinations of panels simply re-weight the data in order to reflect the new pattern of missing data.

Mean Imputation

Mean imputation is never a good idea. Also called "mean-plugging", mean imputation involves replacing missing data for the variable x_{i1} with the mean of that variable, $\bar{x_1}$. As you can probably sense intuitively, this drastically reduces that estimates of the variance of x_1, since we're artificially constraining the variance. Thus, all variance estimates will be attenuated. As Little and Rubin say "This method cannot be recommended". (p. 62)

Mean Imputation with a Dummy

There are multiple proposed methods for mean plugging that involve including a dummy variable for whether or not the individual variable has been mean-plugged. The idea is to condition on imputation, to make up for the fact that we've imputed that particular value. Conditioning on imputation does not help with the bias associated with mean plugging, although just by virtue of variance inflation it does reduce the problem of overconfidence slightly. This method is also not recommended under any circumstances.

Conditional Mean Imputation

Conditional mean imputation involves taking a model for the missing data, which we can state in general form as $E(x_1|x_2, x_3, \ldots x_p)$. This is a little better than unconditional mean imputation, but not much. Essentially this procedure ignores the standard error of the regression, which means that predictions are based only on the systematic relationship between x_1 and the other

variables. You can add in an error term –this is called stochastic regression imputation. Also called Buck's method, this is a "not-too-bad" approach. However, we can do better than this.

Hotdecking, cold decking

In hot decking, missing data are replaced by values from other, similar units in the sample. The term comes from the days of computer cards, when the analyst would grab a card from another unit from the "hot deck" and put it back into the cold deck for reanalysis. As you might imagine, the key to hotdecking is deciding which units are "similar" to the unit that's missing data. The properties of these methods are not well established, but theoretical and empirical investigations indicate that estimates from data which have been imputed using hotdeck methods are unbiased only when the MCAR assumption holds. Hot decking is also quite commonly used in NCES adminstrative surveys.

Cold-decking means taking data from an outside data set (usually a previous survey) and carrying that data into the current

Logical Imputation

In certain limited situations we may logically infer the value of a missing response based on patterns of other responses. For instance, if an individual has indicated that they are 31, we may logically infer that they do not receive Medicare part D benefits.

Multiple Imputation

After their review of the above methods, Little and Rubin conclude:

> Imputations should generally be:
>
> - Conditioned on observed variables, to reduce bias due to nonresponse, improve precision, and preserve association between missing and observed variables;
> - Multivariate, to preserve associations between missing variables;
> - Draws from the predictive distribution rather than means, to provide valid estimates of a wide range of estimands. (Little and Rubin (2002) p. 72)

As it turns out, multiple imputation provides the only method that meets all of these goals.

We'll talk about two algorithms for multiple imputation: Multiple imputation via chained equations (mice) and data augmentation (DA).

Multiple Imputation via Chained Equations (MICE)

In multiple imputation via chained equations, we use separate models for each missing variable, then use the newly imputed data for that variable to influence the estimates for the next variable.

Say we had p variables in our dataset, each of which was missing some data. First, we would choose values for the missing data $x_1^0 \dots x_p^0$, usually at random. Then we would iterate across several cycles denoted by t, with new values drawn from the conditional distribution for each variable:

$$
\begin{aligned}
x_1^{(t+1)} &= p(x_1 | x_2^{(t)}, x_3^{(t)}, \dots x_p^{(t)}) \\
x_2^{(t+1)} &= p(x_2 | x_1^{(t+1)}, x_3^{(t)}, \dots x_p^{(t)}) \\
x_3^{(t+1)} &= p(x_3 | x_1^{(t+1)}, x_2^{(t+1)}, \dots x_p^{(t)}) \\
x_p^{(t+1)} &= p(x_p | x_1^{(t+1)}, x_2^{(t+1)}, \dots x_{(p-1)}^{(t)})
\end{aligned}
$$

The steps for MICE are as follows:

1. Create a model for each missing variable.
2. Run an algorithm that creates estimates for each of the models, which will iterate across all of the variables.
3. Take a random draw for the missing variable from the predictions from the above models.
4. Take the draw and use it to fill in the missing data point–this is called imputation based on the predictive distribution.
5. Repeat the above steps. Theoretical and empirical results indicate that between 2-10 repetitions are appropriate, depending on the complexity of the models and the amount of missing data. This process is what is known as multiple imputation. In typical usage we impute 5 different datasets.
6. Run estimates an each of the separate imputed datasets. These are then combined.

Multiple Imputation via Data Augmentation

The DA algorithm is used by most software for multiple imputation. It works like this: we fist come up with a starting point for the coefficients for the "main" regression. We then use the variance covariance matrix from these parameters to obtain coefficients from regressions in which the dependent variable is the variable missing data– going from the first to the last variable missing any data. We then come up with a prediction for a given variable based on the other data. To this prediction, some random noise is added. This new data is added is added to the completed data set to come up with a new variance covariance matrix. From this This is iterated, until the models converge.

Imputation

Based on the results of either of the above procedures, we can take a draw from the conditional distribution of each x_p and plug it into the dataset. For each missing data point, we plug in a different draw, resulting in multiple sets of data.

The steps for multiple imputation via data augmentation are as follows (Allison, 2002):

1. Select start values for your parameters. Stata does this using the EM algorithm.
2. Use the current values of the means and covariances to obtain estimates of regression coefficients for equations in which each variable with missing data is regressed on all observed variables.
3. Use the regression estimates to generate predicted values for all missing values. To each predicted value, add a random draw based on the distribution of that variable.
4. With the newly completed dataset, recalculate the variance-covariance matrix.
5. With the new variance-covariance matrix, take a random draw from the possible values for the means and covariances.
6. Iterate, starting at step 2.

Combining Results from Imputed Datasets

The analysis is conducted as normal for each imputed dataset. The question is then how to combine the results? Combining parameter estimates is easy: it's the mean of the estimates from all of the datasets.

Variance estimates are a little harder to combine, although it's really not too bad. The key is to reflect the within-imputation variance from each run of the model with the between imputation variance across all models. Take D to be the count of imputed datasets, and V_d is the variance estimate from dataset d. If θ is the estimate we're interested in, and $\widehat{\theta}_d$ is our estimate from dataset d, Rubin's variance estimate is:

$$Var(\theta|X_{obs}) \approx \frac{1}{D}\sum_{i=1}^{D} V_d + \frac{1}{D-1}\sum_{i=1}^{D}(\widehat{\theta}_d - \bar{\widehat{\theta}})^2 = \bar{V} + B$$

Where $\bar{V}$ is the average of all variance estimates and B is the between-imputation variance.

Luckily, we usually don't have to do this by hand, but it's good to know how to do so, since many times "canned" missing data programs might get stuck on a particular model we're using.

Using Stata's missing data commands

Stata has a very strong set of imputation commands. The first thing you'll want to do is to run some basic descriptives to understand the extent of your missing data problem. The most frequent question I get is: how much is too much missing data? There's no one answer to this question. If you have more than 20% of observations missing for more than 50% of your analysis variables, you have a fairly serious problem. You can run all of the missing data routines described here, but it just won't help, because you don't have enough data. The next thing to do is to run the `mvpatterns` command. The output of `mvpatterns` looks like this:

```
Patterns of missing values

  +-------------------------------------------------------------------+
  |                                              _pattern   _mv   _freq |
  |-------------------------------------------------------------------|
  | ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++     0   11638 |
  | .++++++++++++++++++++++++++++++++++++++++++++++++++++++++     1    1727 |
  | +.........++.....................................    50     477 |
  | +................................................    52     171 |
  | ..........++.....................................    51     117 |
  |-------------------------------------------------------------------|
  | +++.++++++++++++++++++++++++++++++++++++++++++++++++++++     1      39 |
  | ++..+++.............++++++++++++++++++++++++.........+    23      31 |
  | ++..................................................+    50      24 |
  | ++..++.............++++++++.........++++++++.........+    32      11 |
  | .+..++.............++++++++.........++++++++.........+    33       6 |
  |-------------------------------------------------------------------|
  | .+..................................................+    51       5 |
  | .+..+++.............++++++++++++++++++++++++.........+    24       4 |
  | .++.++++++++++++++++++++++++++++++++++++++++++++++++++     2       3 |
  | ++..+.+.....................+++++++++++++++.........+    32       3 |
  +-------------------------------------------------------------------+
```

This tells you how missing data patterns may be related across variables.

With that, you're set to begin running the `mi` set of commands. The first thing to do is to tell Stata that you're going to create a "long" dataset, stacking each imputation, one under the other:

```
mi set mlong
```

You next to "register" the data, telling Stata which variables with missing data are going to predicted by which variables without missing data.

With that, you're ready to run the `mi impute` command. The `mi impute` command has a variety of ways of using the existing data to predict the missing data. We're going to use `mi mvn` which makes a broad assumption of multivariate normaility in order to use some advanced MCMC techniques.

Using mi impute chained

The `mi impute chained` command works by specifying a model for each of the missing variables that you want to be imputed. It then runs an iterated series of predictions of each of the missing variables. You'll need to specify several options.

The models to be used for different groups of variables

The number of imputations to be added `add`. Use 5 as the default.

The number of "burn in" iterations to go through. 100 is recommended.

Customizations of the default prediction equations.

Using mi impute mvn

The `mi impute mvn` command works in two steps. In the first step, it uses the expectation-maximization algorithm to find some reasonable starting points. In the second step, it uses a Data Augmentation routine to fill in the missing data in the iterative manner described above.

There are several options that you need to specify before using `mi mvn`:

The number of imputations to be added, add().

The "prior" to be used. The prior provides a probabilisitc description of where you expect the missing data points to be. I recommend a uniform prior.

The number of "burn in" iterations that the DA algorithm should run before you start using the results. Something like 1,000 should work well.

The number of "burn between" iterations that should be used between each dataset included in the multiply imputed results.

How the mcmc chains should be initialized: use the em algorithm.

You can also use save some information about the convergence of the mcmc run.

Once you have your multiply imputed dataset ready to go, you can then use the `mi estimate :` prefix before commands. This will use all of your imputed datasets in estimation, and return results that are averaged across all of the imputed datasets. In the do file, note that I have included a code "chunk" that allows the results from mi estimate and svy to be outputted to the estout command.

In the output from `mi impute mvn` you'll first see the results of the em algorithm:

```
. mi impute mvn /*Assuming mvn pattern, can use multiple methods to predict*/
> `race´ `pared´ byses1
> =
> f1psepln byincome
>
> /*Nonmissing data*/
>
>
>
> ,  add(5)  /*Number of imputations: Use 5 to start*/
> alldots
> noisily
> prior(ridge, df(0.5))
> burnin(2000)
```

```
> burnbetween(500)
> initmcmc(em,  iter(2000) tol(1e-6))
> savewlf(wlf, replace)
> force
> ;

Performing EM optimization:
note: 849 observations omitted from EM estimation because of all imputation variables missing

Iteration 0:   Observed log posterior =  68051.922
Iteration 1:   Observed log posterior =  69071.266

Expectation-maximization estimation      Number obs           =     13407
                                         Number missing       =         0
                                         Number patterns      =         1
Prior: ridge, df=.5                      Obs per pattern: min =     13407
                                                          avg =     13407
                                                          max =     13407

Observed log posterior =  69071.266 at iteration 1

------------------------------------------------------------------------------
             |     amind      asian      black   hispanic  multira~l     byses1
-------------+----------------------------------------------------------------
Coef         |
    f1psepln | -.0026674    .024547   .0157179  -.0159283  -.0031649   .1104196
    byincome | -.0014871   -.010512  -.0289848  -.0222282   .0005692   .2116825
       _cons |  .0338002   .0844891   .3222665   .4119973   .0540025  -2.350626
-------------+----------------------------------------------------------------
Sigma        |
       amind |  .0081838  -.0008288  -.0012732  -.0013977   -.000378  -.0001191
       asian | -.0008288    .086905  -.0140171  -.0141417  -.0043064   .0024234
       black | -.0012732  -.0140171   .1057976  -.0207833   -.005605  -.0010089
    hispanic | -.0013977  -.0141417  -.0207833   .1149542  -.0061889  -.0267954
 multiracial |  -.000378  -.0043064   -.005605  -.0061889   .0430571   .0011872
      byses1 | -.0001191   .0024234  -.0010089  -.0267954   .0011872   .2611111
------------------------------------------------------------------------------
```

Next, Stata runs through the "burn in" period for the mcmc algorithm:

```
Performing MCMC data augmentation:
  burn-in 2000 .........10.........20.........30.........40.........50.........60.........70.........80.........90.....
> ......110.........120.........130.........140.........150.........160.........170.........180.........190.........200..
> 10.........220.........230.........240.........250.........260.........270.........280.........290.........300........
> ....320.........330.........340.........350.........360.........370.........380.........390.........400.........410....
> .........430.........440.........450.........460.........470.........480.........490.........500.........510.........52
> ..530.........540.........550.........560.........570.........580.........590.........600.........610.........620......
> .......640.........650.........660.........670.........680.........690.........700.........710.........720.........730.
> 740.........750.........760.........770.........780.........790.........800.........810.........820.........830........
> .....850.........860.........870.........880.........890.........900.........910.........920.........930.........940...
> 0.........960.........970.........980.........990.........1000.........1010.........1020.........1030.........1040.....
> .........1060.........1070.........1080.........1090.........1100.........1110.........1120.........1130.........1140..
> 150.........1160.........1170.........1180.........1190.........1200.........1210.........1220.........1230.........124
> ..1250.........1260.........1270.........1280.........1290.........1300.........1310.........1320.........1330........
> .....1350.........1360.........1370.........1380.........1390.........1400.........1410.........1420.........1430.....
> ........1450.........1460.........1470.........1480.........1490.........1500.........1510.........1520.........1530...
> 40.........1550.........1560.........1570.........1580.........1590.........1600.........1610.........1620.........1630
> .1640.........1650.........1660.........1670.........1680.........1690.........1700.........1710.........1720.........1
> ....1740.........1750.........1760.........1770.........1780.........1790.........1800.........1810.........1820.......
> .......1840.........1850.........1860.........1870.........1880.........1890.........1900.........1910.........1920....
> 0.........1940.........1950.........1960.........1970.........1980.........1990.........2000 done
```

Last, it imputes the data, iterating the number of times specified in the option burnbetween:

```
imputing m=1 through m=5 ..... done
```

Last, it gives you a summary of the imputation procedure:

```
Multivariate imputation                Imputations =        5
```

```
Multivariate normal regression                added =        5
Imputed: m=1 through m=5                    updated =        0

Prior: ridge, df=.5                      Iterations =     4000
                                            burn-in =     2000
                                            between =      500

               |              Observations per m
               |----------------------------------------------
      Variable |   complete   incomplete   imputed |     total
---------------+-----------------------------------+----------
         amind |      13407          849       849 |     14256
         asian |      13407          849       849 |     14256
         black |      13407          849       849 |     14256
      hispanic |      13407          849       849 |     14256
   multiracial |      13407          849       849 |     14256
        byses1 |      13407          849       849 |     14256
-------------------------------------------------------------
(complete + incomplete = total; imputed is the minimum across m
 of the number of filled in observations.)
```

A Final Warning

Multiple imputation is always the LAST STEP in an analysis. Your design for analysis should be in place, with all models set up and ready to go. In general, it is far too time consuming to deal with multiple imputation when you're conducting preliminary analyses.

Remember, multiple imputation DOES NOT give you a value for any given missing data point. You should never report individual data points from a multiply imputed dataset as actual data. Instead, multiple imputation allows you to estimate results that reflect the actual, observed data without either losing information or biasing your results.