SOCY7708: Hierarchical Linear Modeling
Instructor: Natasha Sarkisian

Missing data

In most datasets, we will encounter the problem of item non-response -- for various reasons respondents often leave particular items blank on questionnaires or decline to give any response during interviews. Sometimes the portion of such missing data can be quite sizeable. This is a serious problem, and the more data points are missing in a dataset, the more likely it is that you will need to address the problem of incomplete cases. But it also becomes more likely that naïve methods of imputing or filling in values for the missing data points are most questionable because the proportion of valid data points relative to the total number of data points is small. We will briefly address these, still commonly used, naïve methods, and learn about more sophisticated techniques.

Types of missing data

The most appropriate way to handle missing or incomplete data will depend upon how data points became missing. Little and Rubin (1987) define three unique types of missing data mechanisms.

Missing Completely at Random (MCAR):
MCAR data exists when missing values are randomly distributed across all observations. In this case, observations with complete data are indistinguishable from those with incomplete data. That is, whether the data point on Y is missing is not at all related to the value of Y or to the values of any Xs in that dataset. E.g. if you are asking people their weight in a survey, some people might fail to respond for no good reason – i.e. their nonresponse is in no way related to what their actual weight is, and is also not related to anything else we might be measuring.

MCAR missing data often exists because investigators randomly assign research participants to complete only some portions of a survey instrument – GSS does that a lot, asking respondents various subsets of questions. MCAR can be confirmed by dividing respondents into those with and without missing data, then using t-tests of mean differences on income, age, gender, and other key variables to establish that the two groups do not differ significantly. But in real life, MCAR assumption is too stringent for most situations other than such random assignment.

Missing at Random (MAR):
MAR data exist when the observations with incomplete data differ from those with complete data, but the pattern of data missingness on Y can be predicted from other variables in the dataset (Xs) and beyond that bears no relationship to Y itself – i.e., whatever nonrandom processes existed in generating the missing data on Y can be explained by the rest of the variables in the dataset. MAR assumes that the actual variables where data are missing are not the cause of the incomplete data -- instead, the cause of the missing data is due to some other factor that we also measured. E.g., one sex may be less likely to disclose its weight.

MAR is much more common than MCAR. MAR data are assumed by most methods of dealing with missing data. It is often but not always tenable. Importantly, the more relevant and related predictors we can include in statistical models, the more likely it is that the MAR assumption will be met. Sometimes, if the data that we already have are not sufficient to make our data MAR, we can try to introduce external data as well – e.g., estimating income based on Census block data associated with the address of the respondent.

If we can assume that data are MAR, the best methods to deal with the missing data issue are multiple imputation and raw maximum likelihood methods. Together, MAR and MCAR are called ignorable missing data patterns, although that's not quite correct as sophisticated methods are still typically necessary to deal with them.

Not Missing at Random (NMAR or nonignorable):
The pattern of data missingness is non-random and it is not predictable from other variables in the dataset. NMAR data arise due to the data missingness pattern being explainable only by the very variable(s) on which the data are missing. E.g., heavy (or light) people may be less likely to disclose their weight. NMAR data are also sometimes described as having selection bias. NMAR data are difficult to deal with, but sometimes that's unavoidable; if the data are NMAR, we need to model the missing-data mechanism. Two approaches used for that are selection models and pattern mixture; however, we will not deal with them here.

Examining missing data

When examining missing data, the first thing is to make sure you know how the missing data were coded and take such codes into account when you do any recoding. It is also important to distinguish two main types of missing data – sometimes questions are not applicable and therefore not asked, but in other situations, questions are asked but not answered. It is very important to distinguish not applicable cases because those often would be cases that you might not want to include in the analyses or sometimes you might want to assign a certain value to them (e.g. if someone is not employed, their hours of work might be missing because that question was not relevant, but in fact we do know that it should be zero. Sometimes, however, datasets code some cases "not applicable" because a respondent has refused to answer some prior case – although coded not applicable, these cases are more likely to be an equivalent of "not answered" – i.e. truly missing data. "Don't know" is often a tough category – sometimes, on ordinal scales measuring opinions, you might be able to place them as the middle category, but in other situations, it becomes missing data.

For all examples here, we will use National Educational Longitudinal Study (NELS) (base year) for our example; it is available on course website.

```
-> tabulation of bys12
     sex of |
 respondent |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |      6,671       48.26       48.26
          2 |      7,032       50.88       99.14
          7 |          4        0.03       99.17
          8 |        115        0.83      100.00
------------+-----------------------------------
      Total |     13,822      100.00
```

```
. tab bys27a

 how well r |
understands |
     spoken |
    english |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |      2,715       19.64       19.64
          2 |        345        2.50       22.14
          3 |         94        0.68       22.82
          4 |         22        0.16       22.98
          8 |         60        0.43       23.41
          9 |     10,586       76.59      100.00
------------+-----------------------------------
      Total |     13,822      100.00

. gen female=(bys12==2) if bys12<7
(119 missing values generated)

. gen spoken=bys27a

. replace spoken=. if bys27a==8
(60 real changes made, 60 to missing)

. replace spoken=1 if bys27a==9
(10586 real changes made)

. tab female, m

     female |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |      6,671       48.26       48.26
          1 |      7,032       50.88       99.14
          . |        119        0.86      100.00
------------+-----------------------------------
      Total |     13,822      100.00

. tab spoken, m

     spoken |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |     13,301       96.23       96.23
          2 |        345        2.50       98.73
          3 |         94        0.68       99.41
          4 |         22        0.16       99.57
          . |         60        0.43      100.00
------------+-----------------------------------
      Total |     13,822      100.00
```

Once you differentiated between truly missing data and the results of skip patterns, you should examine patterns of missing data.

```
. misstable summarize id female spoken, all showzeros
                                                               Obs<.
                                                +------------------------------
               |                                | Unique
      Variable |     Obs=.     Obs>.     Obs<.  | values        Min         Max
  -------------+--------------------------------+------------------------------
            id |         0         0    13,822  |   >500     124902     9199197
        female |       119         0    13,703  |      2          0           1
        spoken |        60         0    13,762  |      4          1           4
  -----------------------------------------------------------------------------
```

```
. misstable patterns id female spoken, freq exok asis

   Missing-value patterns
     (1 means complete)

              |   Pattern
    Frequency |  1  2
  ------------+-------------
       13,645 |  1  1
              |
          117 |  0  1
           58 |  1  0
            2 |  0  0
  ------------+-------------
       13,822 |

  Variables are  (1) female  (2) spoken
```

asis specifies that the order of the variables in the table be the same as the order in which they are specified on the misstable command. (The default is to sort by the number of missing values.)

freq specifies that the table should report frequencies instead of percentages.

exok specifies that the extended missing values .a, .b, ..., .z should be treated as if they do not designate missing. This allows to treat “not applicable” or deliberately skipped cases as distinct from truly missing. You would investigate missing value patterns only after you examined skip patterns. Before examining missing value patterns, look at skip codes and make sure to assign these kinds of codes (.a .b .c etc. rather than a basic missing code .) to those cases that are truly not applicable because they are not really missing.

Methods of handling missing data

Methods of handling missing data

We will briefly address various naïve methods of dealing with missing data (that are no longer recommended), and learn to apply one of the more sophisticated techniques – multiple imputation by chained equations.

I. Available data approaches

1. Listwise (casewise) deletion

If an observation has missing data for any one variable used in a particular analysis, we can omit that observation from the analysis. This approach is the default method of handling incomplete data in Stata, as well as most other commonly-used statistical software.

There is no simple decision rule for whether to drop cases with missing values, or to impute values to replace missing values. Listwise deletion will produce unbiased results if the data are MCAR (but our sample will be smaller so the standard errors will be higher). When the data are MAR, listwise deletion produced biased results but they are actually less problematic than the results of many other common naïve methods of handling missing data. For instance, if the patterns of missing data on your independent variables are not related to the values of the dependent variables, listwise deletion will produce unbiased estimates.

Still, dropping cases with missing data can reduce our sample (and therefore also reduce the precision) substantially, and therefore we often want to avoid it. But when the number of cases with missing data is small (e.g., less than 5% in large samples), it is common simply to drop these cases from analysis.

<u>2. Pairwise deletion</u>
We can compute bivariate correlations or covariances for each pair of variables X and Y using all cases where neither X nor Y is missing – i.e., based upon the available pairwise data. To estimate means and variances of each of the variables, it uses all cases where that variable is not missing. We can then use these means and covariances in subsequent analyses.

Pairwise data deletion is available in a number of SAS and SPSS statistical procedures; Stata does not use it much and for a good reason – pairwise deletion produces biased results and shouldn't be used.

<u>3. Missing data indicators</u>
In this method, we would create a dummy variable equal to 1 in cases where X is missing, and 0 in cases where it is not. Then we would include both X and the dummy variable in the model predicting Y. This is method is very problematic and results in biased estimates.

<u>II. Deterministic imputation methods</u>

<u>1. Mean substitution</u>
The simplest imputation method is to use a variable's mean (or median) to fill in missing data values. This is only appropriate when the data are MCAR, and even then this method creates a spiked distribution at the mean in frequency distributions, lowers the correlations between the imputed variables and the other variables, and underestimates variance. Nevertheless, it is made available as an easy option in many SPSS procedures, and there is a procedure (STANDARD) available for that in SAS. Still, you should avoid using it.

A type of mean substitution where the mean is calculated in a subgroup of the non-missing values, rather than all of them, is also sometimes used; this technique also suffers from the same problems.

<u>2. Single regression imputation (a.k.a. conditional mean substitution)</u>
A better method than to impute using each variable's mean is to use regression analysis on cases without missing data, and then use those regression equations to predict values for the cases with missing data. This imputation technique is available in many statistical packages (for example, in Stata there is "impute" command). This technique still has the problem that all cases with the same values on the independent variables will be imputed with the same value on the missing variable, thus leading to an underestimate of variance; thus, the standard errors in your models will be lower than they should be.

3. Single random (stochastic) regression imputation

To improve upon the single regression imputation method, and especially to compensate for its tendency to lower the variance and therefore lead to an underestimation of standard errors, we can add uncertainty to the imputation of each variable so each imputed case would get a different value. This is done by adding a random value to the predicted result. This random value is usually the regression residual from a randomly selected case from the set of cases with no missing values. SPSS offers stochastic regression imputation – when doing regression imputation, SPSS by default adds the residual of a randomly picked case to each estimate. Impute command in Stata does not offer such an option, but one can use ice command we will learn soon to generate such imputations.

Single random regression imputation is better than regular regression imputation because it preserves the properties of the data both in terms of means and in terms of variation. Still, this residual is just a guess and it is likely that standard errors will be smaller than they should be. Another remaining problem, but it's a serious one, is that it uses imputed data as if they were real – it doesn't allow for the variation between different possible sets of imputed values. That's why we need to move beyond the traditional approaches to those that try to recognize the difference between real and imputed data.

4. Hot deck imputation

As opposed to regression imputation, hotdeck imputation is a nonparametric imputation technique (i.e., doesn't depend on estimating regression parameters). Hot deck imputation involves identifying the most similar case to the case with a missing value and substituting that most similar case's value for the missing value. We need to specify which variables are used to define such similarity – these variables should be related to the variable that's being imputed. Thus, a number of categorical variables are used to form groups, and then cases are randomly picked within those groups. For example:

Obs	Var 1	Var 2	Var 3	Var 4
1	4	1	2	3
2	5	4	2	5
3	3	4	2	.

Hot deck imputation examines the observations with complete records (obs 1 and 2) and substitutes the value of the most similar observation for the missing data point. Here, obs 2 is more similar to obs 3 than obs 1. New data matrix:

Obs	Var 1	Var 2	Var 3	Var 4
1	4	1	2	3
2	5	4	2	5
3	3	4	2	5

After doing this imputation, we analyze the data using the complete database. Stata offers a hot deck algorithm implemented in the hotdeck command. This procedure will tabulate the missing data patterns for the selected variables and will define a row of data with missing values in any of the variables as a `missing line' of data, similarly a `complete line' is one where all the variables contain data. The hotdeck procedure replaces the variables in the `missing lines' with the corresponding values in the `complete lines'. It does so within the groups specified by the "by" variables. Note that if a dataset contains many variables with missing values then it is possible that many of the rows of data will contain at least one missing value. The hotdeck procedure will not work very well in such circumstances. Also, hotdeck procedure assumes is that the missing data are MAR and that the probability of missing data can be fully accounted for by the categorical variables specified in the `by' option.

Hotdeck imputation allows imputing with real, existing values (so categorical variables remain categorical and continuous variables remain continuous). But it can be difficult to define "similarity." Also, once again this approach does not distinguish between real and imputed data and therefore will result in standard errors that are too low.

II. Maximum likelihood methods

The second group of methods we will consider are those based on maximum likelihood estimation. There are two types of techniques in this group.

1. Expectation Maximization (EM) approach:

EM approach is a technique uses ML algorithm to generate a covariance matrix and mean estimates given the available data, and then these estimates can be used in further analyses. All the estimates are obtained through an iterative procedure, and each iteration has two steps. First, in the expectation (E) step, we take estimates of the variances, covariances and means, perhaps from listwise deletion, use these estimates to obtain regression coefficients, and then fill in the missing data based on those coefficients. In the maximization (M) step, having filled in missing data, we use the complete data (using estimated values) to recalculate variances, covariances, and means. These are substituted back into the E step. The procedure iterates through these two steps until convergence is obtained (convergence occurs when the change of the parameter estimates from iteration to iteration becomes negligible). At that point we have maximum likelihood estimates of variances, covariances, and means, and we can use those to make the maximum likelihood estimates of the regression coefficients. Note that the actual imputed data are not generated in this process; only the parameter estimates.

The SPSS Missing Values Analysis (MVA) module uses the EM approach to missing data handling, and it's also available in SAS as SAS-MI; as far as I know, it is not available in Stata.

The strength of the approach is that it has well-known statistical properties and it generally outperforms available data methods and deterministic methods. The main disadvantage is that it adds no uncertainty component to the estimated data. Thus, it still underestimates the standard errors of the coefficients.

2. Direct ML methods

There are alternative maximum likelihood estimators that are better than the ones obtained by the EM algorithm; these involve direct ML method, also known as raw maximum likelihood method, or Full Information Maximum Likelihood estimation (FIML). This technique uses all available data to generate maximum likelihood-based statistics. Like EM algorithm, direct ML methods assume the missing values are MAR. Under an unrestricted mean and covariance structure, direct ML and EM return identical parameter estimate values. Unlike EM, however, direct ML can be employed in the context of fitting user-specified linear models. Direct ML methods are *model-based*, that is, implemented as part of a fitted statistical model. This produces standard errors and parameter estimates under the assumption that the fitted model is not false, so parameter estimates and standard errors are model-dependent. But it also makes it difficult to use variables that are not in the model in your imputation.

Direct ML has the advantage of convenience/ease of use and well-known statistical properties. Unlike EM, it also allows for the direct computation of appropriate standard errors and test statistics. Disadvantages include an assumption of joint multivariate normality of the variables used in the analysis and the lack of an imputed dataset produced by the analysis.

Direct ML method is implemented by the EM algorithm in the SPSS Missing Values option and MIXED procedure in SAS. It is also available in structural equation modeling packages, such as AMOS, LISREL, MPlus, as well as Stata SEM module.

III. Multiple imputation

An alternative to the maximum likelihood is called multiple imputation (MI). In multiple imputation we generate imputed values on the basis of existing data, but we do the imputation more than once, each time including a random component in our imputation. This allows us to prevent underestimating the standard errors of our regression coefficients. We do that by combining coefficient estimates from multiple datasets using special formulas for the standard errors. Specifically, the standard error of such estimates is equal to SQRT[(1 - 1/m)/B +W], where m is the number of replicates, B is the variance of the imputations, and W is the average of the estimated variance.

Multiple imputation is a sort of an approximation to Direct ML. In multiple imputation, we try a *few* plausible values of missing data. In maximum likelihood, we integrate over *all* possible data values, giving more weight to values that are more plausible. MI has the advantage of simplicity over Maximum Likelihood methods, making it particularly suitable for large datasets. The efficiency of MI is high even when the number of imputed datasets is low (3-10), although recent literature suggests that this depends on the amount of missing data--larger amount may necessitate more imputed datasets.

1. Multiple hot deck imputation
Multiple hot deck imputation combines the well-known statistical advantages of EM and direct ML with the ability of hot deck imputation to provide a raw data matrix. The primary difference between multiple hot deck imputation and regular hot deck imputation is that multiple imputation requires that we generate five to ten datasets with imputed values. We then analyze each database and summarize the results into one summary set of findings. Stata offers multiple hot deck imputation as a part of the same hotdeck command we discussed earlier.

2. Multiple Imputation under the Multivariate Normal Model

In this approach, multiple datasets are generated with methods somewhat similar to single random regression imputation, but with some important modifications. To impute the missing values, we use the information from available data to generate a distribution of plausible values for the missing data, and draw from that distribution at random multiple times to produce multiple datasets. The imputed value is affected by two sources of random variation:
(1) It is a random draw from a conditional distribution.
(2) The conditional distribution itself must be estimated, and the estimation contains some error.

Under this method, the model for the missing data given the observed is a fully specified joint model (e.g. multivariate normal). This is difficult to specify for a mixture of continuous and categorical data. Therefore this method assumes all data are normal. But luckily, tests suggest that this type of imputation is quite robust even when the simulation is based on an erroneous model (e.g., when normality is assumed even though the underlying data are not in fact normal). In Stata, it is available as a part of the MI package: see mi impute mvn. Since this method assumes that all variables are continuous (even though it is fairly robust to that violation), we will focus on learning another method, also implemented in Stata, that does not make that assumption.

3. Multiple Imputation by Chained Equations (MICE)
Chained equations method has a similar idea but adopts a different approach. No joint distribution is set up. Rather, we use a series of conditional distributions. E.g. if we have a continuous X, a count variable Y, and a binary Z, and some data for each are missing, we set up (1) a linear regression of X on Y and Z, (2) a Poisson regression of Y on X and Z, and (3) a logistic regression of Z on X and Y. We start by fitting (1) to the observed data, then simulate any missing X from that model. Then we fit (2) using observed Z and X (with missing values filled out by the simulations), and simulate any missing Y. Then we fit (3) using X and Y (with missing values filled by the simulations). We go through multiple iterations, fitting each model in turn, and updating the simulations with each iteration, waiting for the model to converge. We do that multiple times producing multiple datasets.

MICE is computationally simple to implement, and is available in Stata. The drawback is that the conditionals may not specify a unique joint distribution which can make the inferences problematic; but the simulation studies suggest it often works quite well, so it is increasingly used. Another problem for multilevel models is that there are special implementations of MICE for multilevel data, but these are not yet available in Stata, you would need to go to R for those, but hopefully that changes soon.

A few rules crucial for implementing any type of MI:

- All variables included in your models should be included in the imputation process; you can also add auxiliary variables that would improve predictions for some variables included in the model.
- Dependent variable should always be included in the imputation process, but there are different opinions on whether their imputed values should be subsequently discarded (this is called MID—multiple imputation then deletion) – for a while, the consensus was that MID was better than regular MI (i.e., that it is better to drop the imputed values of the dependent variable), but recent work suggests that the two approaches produce equivalent

results. What is clear is that it is definitely beneficial to keep imputed values of the dependent variable in a situation when additional variables were used in imputing this dependent variable—such supplementary variables enhance the imputation of the dependent variable but are not included in final data analysis models. In this case, imputed values of the dependent variable contain extra information and should definitely be preserved. (To better understand this situation, see: von Hippel, Paul T. 2007. Regression with Missing Ys: An Improved Strategy for Analyzing Multiply Imputed Data. *Sociological Methodology 37(1),* 83-117.)

- If you have interactions or nonlinear relationships among the variables, you should create the variables for those before doing the imputations – otherwise the imputed values will only reflect the linear relationships among the variables. There is some disagreement, however, whether those interactions and nonlinear terms should be imputed separately (this is known as "Just Another Variable" approach) or be a product of imputations of main terms.
- If you are using categorical variables as sets of dummies, you need to create single variables representing those sets before doing the imputation, even if you only plan to use them as dummies later on.
- It is helpful to transform skewed variables before imputing, then back-transform them for analyses.
- Only "soft" missing data (those denoted by just a dot) get imputed – "hard" missing data – e.g., .a, .b, etc. – do not get imputed.
- Another issue to consider when using multiple imputation is the number of datasets to impute. The larger the amount of missing data, the more datasets you'd need to avoid loss of efficiency. Typically, 5-10 can be sufficient, but with a lot of missing data, you might need up to 20. You can consult the following article for the info on that: Graham, J. W., A. E. Olchowski, and T. D. Gilreath. 2007. How many imputations are really needed? Some practical clarifications of multiple imputation theory. *Prevention Science 8(3)*: 206-213.
- If you want to do exploratory analyses, generate an extra imputed dataset specifically for that part of the analysis; then, for the final results, run your model on a number of other imputed datasets, not including the preliminary one.
- MICE is not designed specifically for multilevel data, so you will have to make adjustments to account for your data structure. There are versions of MICE for multilevel data in R – hopefully they will be implemented in Stata soon as well; see more details in the assigned readings.
- Multiple imputation of a longitudinal dataset should be done on a wide rather than long dataset (i.e., there should be separate variables for each year of data, rather than multiple observations within each case). That will allow you to use observations from one year to impute data in another year, which will improve the quality of prediction as well as account for the clustered nature of the data. To change between long and wide format of data in Stata, you can use reshape command (or mi reshape after imputation).
- For a longitudinal dataset, additional problems will be created by attrition—loss of cases over time for various reasons. It is possible to estimate models even if you have different number of time points for different cases, but oftentimes it is useful to conduct sensitivity analyses and check how the results would look like if those cases that are missing entirely

at certain time points are imputed (either under MAR assumption or using, for example, sensitivity testing approaches for NMAR; see below).

- When imputing multilevel datasets that are not longitudinal, we need to include information about clusters. The most common method available in Stata is to include dummy variables that serve as cluster indicators as separate variables in the imputation process. This method is fully appropriate if you are planning to only have random intercept but not random slopes. If you plan to have random slopes, you could try to include interactions of these cluster dummies with each of the independent variables that should have random slopes, although this will likely be too many variables, especially if you have multiple random slopes. (You might want to explore the issue of which random slopes you want to include using listwise deletion dataset in order to limit the number of interactions to include.) Another approach is to perform imputation separately for each cluster – but the clusters would have to be quite large for this approach to be feasible. Again, specialized R packages might be preferable.
- If you have missing data on level 2, do the multiple imputation separately for that level. To do that, aggregate all your level 1 variables (dependent and independent) by averaging them across level 2 units, and combine these aggregated variables with the level 2 variables into a separate level 2 dataset. Do the multiple imputation for that dataset. Then, for level 1, if you use cluster dummies, then just do level 1 imputation without any level 2 variables and then merge the two imputed datasets. If using level 2 predictors rather than cluster dummies, then combine each of the multiply imputed level 2 datasets with the level 1 dataset (before doing any imputation on level 1). After that, do the imputation for level 1 dataset using imputed level 2 variables -- you should generate one imputation per each merged dataset.

<u>Example for learning MICE</u>

School level variables:

```
. tab bysc30, m

  is this a |
     public |
     school |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |     11,342       82.06       82.06
          2 |      2,295       16.60       98.66
          . |        185        1.34      100.00
------------+-----------------------------------
      Total |     13,822      100.00

. gen public=bysc30==1 if bysc30<.
(185 missing values generated)

. tab public, m

     public |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |      2,295       16.60       16.60
          1 |     11,342       82.06       98.66
          . |        185        1.34      100.00
------------+-----------------------------------
      Total |     13,822      100.00
```

```
. tab bysc16e, m

     pct of |
students in |
 english as |
   2nd lang |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |     10,293       74.47       74.47
          1 |      1,111        8.04       82.51
          2 |        408        2.95       85.46
          3 |        278        2.01       87.47
          4 |        249        1.80       89.27
          5 |        176        1.27       90.54
          6 |        147        1.06       91.61
          7 |        190        1.37       92.98
          8 |         82        0.59       93.58
          9 |         65        0.47       94.05
         10 |        104        0.75       94.80
         11 |         47        0.34       95.14
         12 |         75        0.54       95.68
         13 |         46        0.33       96.01
         14 |          9        0.07       96.08
         15 |         34        0.25       96.32
         16 |         19        0.14       96.46
         17 |         16        0.12       96.58
         18 |         30        0.22       96.79
         19 |         25        0.18       96.98
         22 |         59        0.43       97.40
         24 |          9        0.07       97.47
         25 |         37        0.27       97.74
         32 |         18        0.13       97.87
         33 |          5        0.04       97.90
         34 |         18        0.13       98.03
         42 |         18        0.13       98.16
         43 |          5        0.04       98.20
       9998 |         64        0.46       98.66
          . |        185        1.34      100.00
------------+-----------------------------------
      Total |     13,822      100.00

. gen pct_esl=bysc16e
(185 missing values generated)

. replace pct_esl=. if bysc16e==9998
(64 real changes made, 64 to missing)

. tab pct_esl, m

    pct_esl |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |     10,293       74.47       74.47
          1 |      1,111        8.04       82.51
          2 |        408        2.95       85.46
          3 |        278        2.01       87.47
          4 |        249        1.80       89.27
          5 |        176        1.27       90.54
          6 |        147        1.06       91.61
          7 |        190        1.37       92.98
          8 |         82        0.59       93.58
          9 |         65        0.47       94.05
         10 |        104        0.75       94.80
         11 |         47        0.34       95.14
         12 |         75        0.54       95.68
         13 |         46        0.33       96.01
         14 |          9        0.07       96.08
         15 |         34        0.25       96.32
         16 |         19        0.14       96.46
         17 |         16        0.12       96.58
         18 |         30        0.22       96.79
         19 |         25        0.18       96.98
         22 |         59        0.43       97.40
         24 |          9        0.07       97.47
         25 |         37        0.27       97.74
         32 |         18        0.13       97.87
         33 |          5        0.04       97.90
         34 |         18        0.13       98.03
         42 |         18        0.13       98.16
         43 |          5        0.04       98.20
          . |        249        1.80      100.00
------------+-----------------------------------
      Total |     13,822      100.00

. tab  bysc47g

    teacher |
  morale is |
       high |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        247        1.81        1.81
          2 |        639        4.69        6.50
          3 |      2,198       16.12       22.61
          4 |      6,490       47.59       70.21
          5 |      4,032       29.57       99.77
          8 |         31        0.23      100.00
------------+-----------------------------------
      Total |     13,637      100.00

. gen morale=bysc47g
(185 missing values generated)

. replace morale=. if bysc47g==8
(31 real changes made, 31 to missing)

. tab morale, m

     morale |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        247        1.79        1.79
          2 |        639        4.62        6.41
          3 |      2,198       15.90       22.31
          4 |      6,490       46.95       69.27
          5 |      4,032       29.17       98.44
          . |        216        1.56      100.00
------------+-----------------------------------
      Total |     13,822      100.00
```

Individual level variables:

We already created female and spoken earlier so we will use them. Our dependent variable will be science standardized scores:

```
. sum  by2xsstd
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    by2xsstd |     13822    52.60967    13.75539      31.62      99.99

. tab  by2xsstd if  by2xsstd>=99.98

    science |
standardize |
    d score |      Freq.     Percent        Cum.
------------+-----------------------------------
      99.98 |         43        7.85        7.85
      99.99 |        505       92.15      100.00
------------+-----------------------------------
      Total |        548      100.00

. gen science= by2xsstd

. replace science=. if  by2xsstd>=99.98
(548 real changes made, 548 to missing)
```

We will also create a few additional independent variables for level 1:
Parents' education:

```
BYPARED Parent Qx Student Qx Label
01 1, 2 1 Did not finish high school
02 3, 4 2 High school grad or GED
03 5-10 3, 4 GT H.S. & LT 4 year degree
04 11 5 College graduate
05 12 6 M.A. or equivalent
06 13 7 Ph.D., M.D., other
07 - 8 Don't know
98 Missing
```

```
. tab bypared, m

    parents |
    highest |
  education |
      level |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |      1,562       11.30       11.30
          2 |      2,874       20.79       32.09
          3 |      5,561       40.23       72.33
          4 |      1,930       13.96       86.29
          5 |      1,126        8.15       94.44
          6 |        664        4.80       99.24
          7 |         93        0.67       99.91
         98 |         12        0.09      100.00
------------+-----------------------------------
      Total |     13,822      100.00

. recode bypared (1=10) (2=12) (3=14) (4=16) (5=18) (6=20) (7=.) (98=.), gen(pared)
(13822 differences between bypared and pared)

. tab pared, m

  RECODE of |
    bypared |
   (parents |
    highest |
  education |
     level) |      Freq.     Percent        Cum.
------------+-----------------------------------
         10 |      1,562       11.30       11.30
         12 |      2,874       20.79       32.09
         14 |      5,561       40.23       72.33
         16 |      1,930       13.96       86.29
         18 |      1,126        8.15       94.44
         20 |        664        4.80       99.24
          . |        105        0.76      100.00
------------+-----------------------------------
      Total |     13,822      100.00
```

Race:

```
1 950 6.9 API
2 1858 13.6 HISPANIC
3 1464 10.7 BLACK,NON-HISPANIC
4 8896 65.0 WHITE,NON-HISPANIC
5 514 3.8 AMERICAN INDIAN
{blank} 1093 .0 {NONR/NOT IN SAMPLE THIS WAVE}
6 25 .0 {MULTIPLE RESPNSE}
7 25 .0 {REFUSAL}
8 90 .0 {MISSING}

. tab bys31a, m

         rs |
race/ethnic |
 background |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        950        6.87        6.87
          2 |      1,858       13.44       20.32
          3 |      1,464       10.59       30.91
          4 |      8,896       64.36       95.27
          5 |        514        3.72       98.99
          6 |         25        0.18       99.17
          7 |         25        0.18       99.35
          8 |         90        0.65      100.00
------------+-----------------------------------
      Total |     13,822      100.00
. recode bys31a (4=1) (3=2) (2=3) (1=4) (5=5) (6/8=.), gen(race5)
(13308 differences between bys31a and race5)

. tab race5, m

  RECODE of |
 bys31a (rs |
race/ethnic |
background) |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |      8,896       64.36       64.36
          2 |      1,464       10.59       74.95
          3 |      1,858       13.44       88.40
          4 |        950        6.87       95.27
          5 |        514        3.72       98.99
          . |        140        1.01      100.00
------------+-----------------------------------
      Total |     13,822      100.00
```

```
. gen white=(race5==1) if race5<.
(140 missing values generated)

. gen black=(race5==2) if race5<.
(140 missing values generated)

. gen latino=(race5==3) if race5<.
(140 missing values generated)

. gen asian=(race5==4) if race5<.
(140 missing values generated)

. gen native=(race5==5) if race5<.
(140 missing values generated)
```

Creating a reduced dataset:

```
. keep  id sch_id sstratid public pct_esl morale  science female spoken pared race5
white black latino asian native
```

Aggregating level 1 variables and creating level 2 dataset:

```
. for var  science female spoken pared white black latino asian native: bysort sch_id:
egen Xm=mean(X)

->  bysort sch_id: egen sciencem=mean(science)
(229 missing values generated)

->  bysort sch_id: egen femalem=mean(female)
(10 missing values generated)

->  bysort sch_id: egen spokenm=mean(spoken)

->  bysort sch_id: egen paredm=mean(pared)

->  bysort sch_id: egen whitem=mean(white)
(34 missing values generated)

->  bysort sch_id: egen blackm=mean(black)
(34 missing values generated)

->  bysort sch_id: egen latinom=mean(latino)
(34 missing values generated)

->  bysort sch_id: egen asianm=mean(asian)
(34 missing values generated)

->  bysort sch_id: egen nativem=mean(native)
(34 missing values generated)

. bysort sch_id: gen case=_n

. save "C:\Users\sarkisin\nels_science.dta"
file C:\Users\nels_science.dta saved

. drop if case~=1
(12809 observations deleted)

. sum sch_id

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
```

```
      sch_id |      1013    46197.74    26628.62       1249      91991

. drop  id science- spoken case

. save "C:\Users\sarkisin\nels_science_lev2.dta"
file C:\Users\sarkisin\nels_science_lev2.dta saved
```

<u>Learning to use MICE</u>

MICE is available both within the mi module in Stata as well as a separate package. I will mostly demonstrate its use with the separate package but also introduce the mi module version (mi impute chained).

For a separate module, you find the command using

```
. net search mice
```

You need to install st0067_4 from http://www.stata-journal.com/software/sj9-3

Here's the syntax:

```
ice mainvarlist [if] [in] [weight] [, ice_major_options ice_less_used_options]

    ice_major_options              description
    ------------------------------------------------------------------------
    clear                          clears the original data from memory and
                                     loads the imputed dataset into memory
    dryrun                         reports the prediction equations -- no
                                     imputations are done
    eq(eqlist)                     defines customized prediction equations
    m(#)                           defines the number of imputations
    match[(varlist)]               prediction matching for each member of
                                     varlist
    passive(passivelist)           passive imputation
    saving(filename [, replace])   imputed and nonimputed variables are
                                     stored to filename
    ------------------------------------------------------------------------

ice_less_used_options          description
    ------------------------------------------------------------------------
    boot[(varlist)]                estimates regression coefficients for
                                     varlist in a bootstrap sample
    by(varlist)                    imputation within the levels implied by
                                     varlist
    cc(varlist)                    prevents imputation of missing data in
                                     observations in which varlist has a
                                     missing value
    cmd(cmdlist)                   defines regression command(s) to be used
                                     for imputation
    conditional(condlist)          conditional imputation
    cycles(#)                      determines number of cycles of regression
                                     switching
    dropmissing                    omits all observations not in the
                                     estimation sample from the output
    eqdrop(eqdroplist)             removes variables from prediction
                                     equations
    genmiss(string)                creates missingness indicator variable(s)
    id(newvar)                     creates newvar containing the original
                                     sort order of the data
    interval(intlist)              imputes interval-censored variables
    monotone                       assumes pattern of missingness is
```

```
                                monotone, and creates relevant
                                prediction equations
    noconstant                suppresses the regression constant
    nopp                      suppresses special treatment of perfect
                                prediction
    noshoweq                  suppresses presentation of prediction
                                equations
    noverbose                 suppresses messages showing the progress
                                of the imputations
    nowarning                 suppresses warning messages
    on(varlist)               imputes each member of mainvarlist
                                univariately
    orderasis                 enters the variables in the order given
    persist                   ignores errors when trying to impute
                                "difficult" variables and/or models
    restrict([varname] [if])  fits models on a specified subsample,
                                impute missing data for entire
                                estimation sample
    seed(#)                   sets random-number seed
    substitute(sublist)       substitutes dummy variables for multilevel
                                categorical variables
    trace(trace_filename)     monitors convergence of the imputation
                                algorithm
    ------------------------------------------------------------------------
```

Sets of imputed and nonimputed variables are stored to a new file specified using saving option. Option replace permits filename to be overwritten with new data. Any number of complete imputations may be created (defined by option m). We start with multiple imputation for level 2. Let's examine missing data:

```
. misstable summarize public pct_esl morale sciencem femalem spokenm paredm blackm
latinom asianm nativem, all showzeros
                                                          Obs<.
                                              +------------------------------
               |                              | Unique
      Variable |     Obs=.     Obs>.     Obs<.  | values        Min         Max
  -------------+------------------------------+------------------------------
        public |        17         0       996  |      2          0           1
       pct_esl |        22         0       991  |     28          0          43
        morale |        19         0       994  |      5          1           5
      sciencem |        21         0       992  |   >500   34.50857       71.11
       femalem |         2         0     1,011  |    125          0           1
       spokenm |         0         0     1,013  |     74          1           3
        paredm |         0         0     1,013  |    352         10          20
        blackm |         3         0     1,010  |     95          0           1
       latinom |         3         0     1,010  |    126          0           1
        asianm |         3         0     1,010  |     82          0    .9285714
       nativem |         3         0     1,010  |     57          0           1
  -----------------------------------------------------------------------------

. misstable patterns public pct_esl morale sciencem femalem spokenm paredm blackm
latinom asianm nativem, freq exok asis

             Missing-value patterns
               (1 means complete)

             |   Pattern
   Frequency |  1  2  3  4    5  6  7  8    9
  ------------+-------------------------------
         966 |  1  1  1  1    1  1  1  1    1
             |
          18 |  1  1  1  0    1  1  1  1    1
          14 |  0  0  0  1    1  1  1  1    1
           5 |  1  0  1  1    1  1  1  1    1
           3 |  0  0  0  0    1  1  1  1    1
           3 |  1  1  1  1    1  0  0  0    0
           2 |  1  1  0  1    1  1  1  1    1
           2 |  1  1  1  1    0  1  1  1    1
  ------------+------------------------------
       1,013 |

  Variables are  (1) public  (2) pct_esl  (3) morale  (4) sciencem  (5) femalem  (6)
blackm (7) latinom  (8) asianm  (9) nativem
```

Before we run actual multiple imputation using ICE, we will do a dry run – check that everything looks correct, without generating actual datasets:

```
. ice public pct_esl morale   sciencem femalem spokenm paredm blackm latinom asianm
nativem, saving("C:\Users\sarkisin\nels_imputed.dta", replace) m(5) cmd(morale:
ologit)
   #missing |
     values |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        966       95.36       95.36
          1 |         27        2.67       98.03
          3 |         14        1.38       99.41
          4 |          6        0.59      100.00
------------+-----------------------------------
      Total |      1,013      100.00

   Variable | Command | Prediction equation
------------+---------+-------------------------------------------------------
     public | logit   | pct_esl morale sciencem femalem spokenm paredm blackm
            |         | latinom asianm nativem
    pct_esl | regress | public morale sciencem femalem spokenm paredm blackm
            |         | latinom asianm nativem
     morale | ologit  | public pct_esl sciencem femalem spokenm paredm blackm
            |         | latinom asianm nativem
   sciencem | regress | public pct_esl morale femalem spokenm paredm blackm
            |         | latinom asianm nativem
    femalem | regress | public pct_esl morale sciencem spokenm paredm blackm
            |         | latinom asianm nativem
    spokenm |         | [No missing data in estimation sample]
     paredm |         | [No missing data in estimation sample]
     blackm | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | latinom asianm nativem
    latinom | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | blackm asianm nativem
     asianm | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | blackm latinom nativem
    nativem | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | blackm latinom asianm
------------------------------------------------------------------------------
End of dry run. No imputations were done, no files were created.
```

Note that I did not include whitem because it would create collinearity problems. Also note that in cmd option, we specified that we want to use ologit for morale. The default cmd for a variable is logit when there are two distinct values, mlogit when there are 3-5 and regress otherwise. Note that unless the dataset is large, use of the mlogit command may produce unstable estimates if the number of levels is too large.

This way, ologit is used to impute morale, but morale variable itself is used as if it were continuous when imputing other variables. If we want to break it up into dummies when imputing other variables, we can use o. prefix:

```
. ice public pct_esl o.morale sciencem femalem spokenm paredm blackm latinom asianm
nativem, saving("C:\Users\sarkisin\nels_imputed.dta", replace) m(5) dryrun

=> xi: ice public pct_esl morale i.morale sciencem femalem spokenm paredm blackm
latinom asianm nativem, cmd(morale:ologit) substitute(morale:i.morale)
>  saving(C:\Users\sarkisin\nels_imputed.dta, replace) m(5) dryrun

i.morale          _Imorale_1-5        (naturally coded; _Imorale_1 omitted)

   #missing |
     values |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        966       95.36       95.36
          1 |         27        2.67       98.03
          3 |         14        1.38       99.41
          4 |          6        0.59      100.00
------------+-----------------------------------
      Total |      1,013      100.00

   Variable | Command | Prediction equation
------------+---------+-------------------------------------------------------
     public | logit   | pct_esl _Imorale_2 _Imorale_3 _Imorale_4 _Imorale_5
            |         | sciencem femalem spokenm paredm blackm latinom asianm
            |         | nativem
    pct_esl | regress | public _Imorale_2 _Imorale_3 _Imorale_4 _Imorale_5
            |         | sciencem femalem spokenm paredm blackm latinom asianm
            |         | nativem
     morale | ologit  | public pct_esl sciencem femalem spokenm paredm blackm
            |         | latinom asianm nativem
 _Imorale_2 |         | [Passively imputed from (morale==2)]
 _Imorale_3 |         | [Passively imputed from (morale==3)]
 _Imorale_4 |         | [Passively imputed from (morale==4)]
 _Imorale_5 |         | [Passively imputed from (morale==5)]
   sciencem | regress | public pct_esl _Imorale_2 _Imorale_3 _Imorale_4
            |         | _Imorale_5 femalem spokenm paredm blackm latinom
            |         | asianm nativem
    femalem | regress | public pct_esl _Imorale_2 _Imorale_3 _Imorale_4
            |         | _Imorale_5 sciencem spokenm paredm blackm latinom
            |         | asianm nativem
    spokenm |         | [No missing data in estimation sample]
     paredm |         | [No missing data in estimation sample]
     blackm | regress | public pct_esl _Imorale_2 _Imorale_3 _Imorale_4
            |         | _Imorale_5 sciencem femalem spokenm paredm latinom
            |         | asianm nativem
    latinom | regress | public pct_esl _Imorale_2 _Imorale_3 _Imorale_4
            |         | _Imorale_5 sciencem femalem spokenm paredm blackm
            |         | asianm nativem
     asianm | regress | public pct_esl _Imorale_2 _Imorale_3 _Imorale_4
            |         | _Imorale_5 sciencem femalem spokenm paredm blackm
            |         | latinom nativem
    nativem | regress | public pct_esl _Imorale_2 _Imorale_3 _Imorale_4
            |         | _Imorale_5 sciencem femalem spokenm paredm blackm
            |         | latinom asianm
------------------------------------------------------------------------------
```

Similarly, m. can be used when the variable is nominal rather than ordinal; for nominal variables, always use m. rather than cmd option – otherwise, you would be treating your nominal variable as if it's continuous when imputing other variables. Specifying m. tells Stata that we want to use mlogit to impute that variable and that we want to use this nominal variable as a set of dummies when imputing other variables. Also, in general, if your dataset has a number of dummies instead (e.g., if we had separate dummies for black and Hispanic for race), it is a good idea to generate a multicategory variable first and use that variable rather than separate dummies in the imputation process—otherwise, your dummies will be imputed independently and therefore can end up overlapping.

Since our dry run looks good, we can generate imputed data:

```
. ice public pct_esl morale   sciencem femalem spokenm paredm blackm latinom asianm
nativem, saving("C:\Users\sarkisin\nels_imputed.dta", replace) m(5) cmd(morale:
ologit)

   #missing |
     values |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        966       95.36       95.36
          1 |         27        2.67       98.03
          3 |         14        1.38       99.41
          4 |          6        0.59      100.00
------------+-----------------------------------
      Total |      1,013      100.00

   Variable | Command | Prediction equation
------------+---------+------------------------------------------------------
     public | logit   | pct_esl morale sciencem femalem spokenm paredm blackm
            |         | latinom asianm nativem
    pct_esl | regress | public morale sciencem femalem spokenm paredm blackm
            |         | latinom asianm nativem
     morale | ologit  | public pct_esl sciencem femalem spokenm paredm blackm
            |         | latinom asianm nativem
   sciencem | regress | public pct_esl morale femalem spokenm paredm blackm
            |         | latinom asianm nativem
    femalem | regress | public pct_esl morale sciencem spokenm paredm blackm
            |         | latinom asianm nativem
    spokenm |         | [No missing data in estimation sample]
     paredm |         | [No missing data in estimation sample]
     blackm | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | latinom asianm nativem
    latinom | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | blackm asianm nativem
     asianm | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | blackm latinom nativem
    nativem | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | blackm latinom asianm
------------------------------------------------------------------------------

Imputing .
[Perfect prediction detected: using auglogit to impute public]
.........1
[Perfect prediction detected: using auglogit to impute public]
..........2
[Perfect prediction detected: using auglogit to impute public]
..........3
[Perfect prediction detected: using auglogit to impute public]
..........4.
[Perfect prediction detected: using auglogit to impute public]
```

```
.........5
(note: file C:\Users\sarkisin\nels_imputed.dta not found)
file C:\Users\sarkisin\nels_imputed.dta saved
```

Some additional options for ice (see help ice for more):

eq(eqlist) allows one to define customized prediction equations for any subset of variables. The default is that each variable in mainvarlist with any missing data is imputed from all the other variables in mainvarlist. The option allows great flexibility in the possible imputation schemes. The syntax of eqlist is varname1:varlist1 [,varname2:varlist2 ...], where each varname# (or varlist#) is a member (or subset) of mainvarlist. It is important to ensure that each equation is sensible. ice places no restrictions except to check that all variables mentioned are indeed in mainvarlist.

```
. ice public pct_esl morale   sciencem femalem spokenm paredm blackm latinom asianm
nativem, saving(imputed.dta, replace) m(5) cmd(morale: ologit) eq(pct_esl:sciencem
femalem spokenm) dryrun

   #missing |
     values |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        966       95.36       95.36
          1 |         27        2.67       98.03
          3 |         14        1.38       99.41
          4 |          3        0.30       99.70
          5 |          3        0.30      100.00
------------+-----------------------------------
      Total |      1,013      100.00

   Variable | Command | Prediction equation
------------+---------+-------------------------------------------------------
     public | logit   | pct_esl morale sciencem femalem spokenm paredm whitem
            |         | blackm latinom asianm nativem
    pct_esl | regress | sciencem femalem spokenm
     morale | ologit  | public pct_esl sciencem femalem spokenm paredm whitem
            |         | blackm latinom asianm nativem
   sciencem | regress | public pct_esl morale femalem spokenm paredm whitem
            |         | blackm latinom asianm nativem
    femalem | regress | public pct_esl morale sciencem spokenm paredm whitem
            |         | blackm latinom asianm nativem
    spokenm |         | [No missing data in estimation sample]
     paredm |         | [No missing data in estimation sample]
     whitem | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | blackm latinom asianm nativem
     blackm | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | whitem latinom asianm nativem
    latinom | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | whitem blackm asianm nativem
     asianm | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | whitem blackm latinom nativem
    nativem | regress | public pct_esl morale sciencem femalem spokenm paredm
            |         | whitem blackm latinom asianm
------------------------------------------------------------------------------
End of dry run. No imputations were done, no files were created.
```

Note that I used dryrun option to just check whether the equations work.

Eqdrop option: This option specifies which variables to omit from certain equations. You should omit as little as possible in order to make imputation run. The syntax of eqdroplist is varname1:varlist1 [, varname2:varlist2 ...]. For example:

```
. ice public pct_esl morale   sciencem femalem spokenm paredm blackm latinom asianm
nativem, saving(imputed.dta, replace) m(5) cmd(morale: ologit) eqdrop(pct_esl: paredm
blackm latinom asianm nativem) dryrun
```

One more useful option, genmiss(name), creates an indicator variable for the missingness of data in any variable in mainvarlist for which at least one value has been imputed. The indicator variable is set to missing for observations excluded by if, in, etc. The indicator variable for xvar is named namexvar (of course, you get to specify the actual name).

Passive option: If we are using interactions, for example, between x1 and x2 (e.g., x12=x1*x2), we should use passive option, passive(x12:x1*x2). For example:

```
. sum science

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    sciencem |       992    50.56493    5.597011   34.50857      71.11

. gen sciencem_m=sciencem-r(mean)
(21 missing values generated)

. sum female

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     femalem |      1011    .5072957    .1963534          0          1

. gen femalem_m=female-r(mean)
(2 missing values generated)

. gen scifemm= sciencem_m* femalem_m
(23 missing values generated)

. ice public pct_esl morale   science_m female_m spokenm paredm blackm latinom asianm
native scifemm, saving("C:\Users\sarkisin\nels_imputed.dta", replace) m(5) cmd(morale:
ologit) dryrun passive(scifemm: science_m*female_m)
```

However, some people recently have been arguing that interactions and nonlinear terms should not be imputed passively and instead should be imputed separately – this is known as “Just Another Variable” approach.

Persist option: If we are using a lot of dummies, we are likely to run into errors when running imputation, especially in logit-based models. You can tell Stata to ignore those errors by specifying “persist” option; however, the resulting imputation would be problematic. You might want to use it to see which variables keep creating errors and then exclude those variables from corresponding equations using eqdrop option.

Conditional option: This option specifies criteria for inclusion/exclusion of cases during imputation; it is useful when your variable should only have values for a subset for the whole sample. For example:

```
. ice public pct_esl morale   sciencem femalem spokenm paredm blackm latinom asianm
nativem, saving("C:\Users\sarkisin\nels_imputed.dta", replace) m(5) cmd(morale:
ologit) dryrun conditional(sciencem: public==1 \ morale: public==1)
```

Interval option: This option can be used to limit the range of a variable during imputation (because it should not be top or bottom coded afterwards!):

```
gen pct_eslll=pct_esl
gen pct_eslul=pct_esl
replace pct_eslll=0 if pct_esl==.
replace pct_eslul=100 if pct_esl==.
ice public pct_esl morale   sciencem femalem spokenm paredm blackm latinom asianm
nativem, saving("C:\Users\sarkisin\nels_imputed.dta", replace) m(5) cmd(morale:
ologit) dryrun seed(1234) interval(pct_esl: pct_eslll pct_eslul)
```

Seed option: This option allows to make a fixed starting point for random number generation, so that every time you run that command, you get exactly the same imputed datasets rather than different ones every time. For example:

```
ice public pct_esl morale   sciencem femalem spokenm paredm blackm latinom asianm
nativem, saving("C:\Users\sarkisin\nels_imputed.dta", replace) m(5) cmd(morale:
ologit) dryrun seed(1234)
```

If your dataset is large and your model includes many variables, you might run into problems running imputation in Intercooled Stata (Stata IC). In that case, try Stata SE, it is available at apps.bc.edu – you would want to specify large matsize, e.g.:

```
. set matsize 1200
```

Note that imputation is often a pretty slow process, especially if your dataset is large. And you often have to keep adjusting the imputation model until it runs without errors.

Stata built-in module

To do the same thing with the Stata built-in module, we need to first specify the style of imputed data storage and register variables. The styles are specified with mi set command and to better understand them, we look at the example from Stata documentation: type help mi_styles to see four styles: wide, flong, mlong, and flongsep.
Original dataset, with two variables – a and b, as well as a third variable that is equal to their sum c=a+b – and two cases (observations), with one missing value:

```
+-------------------+
| a       b       c |
|-------------------|
| 1       2       3 |
| 4       .       . |
+-------------------+
```

We are doing two imputations (M=2), which generate different guesses for the missing values of b for case #2. We have two imputed values for b for that case, namely, 4.5 and 5.5. The resulting datasets are m=0 (original data), m=1, and m=2.

```
                 +-------------------+
m=0:             | a       b       c |
                 |-------------------|
                 | 1       2       3 |
                 | 4       .       . |
                 +-------------------+

                 +-------------------+
m=1:             | a       b       c |
                 |-------------------|
                 | 1       2       3 |
                 | 4     4.5     8.5 |
                 +-------------------+

                 +-------------------+
m=2:             | a       b       c |
                 |-------------------|
                 | 1       2       3 |
                 | 4     5.5     9.5 |
                 +-------------------+
```

In mi terminology in Stata module, a is a regular variable, b is an imputed variable, and c is a passive variable.

We can store the resulting data differently.

1. Style wide

```
     +------------------------------------------------+
     | a   b   c   _1_b   _2_b   _1_c   _2_c   _mi_miss |
     |------------------------------------------------|
  1. | 1   2   3      2      2      3      3          0 |
  2. | 4   .   .    4.5    5.5    8.5    9.5          1 |
     +------------------------------------------------+
```

Here, _mi_miss contains 0 for complete observations and 1 for incomplete observations.

Style flong

```
+-------------------------------------------+
     | a     b     c   _mi_miss   _mi_m   _mi_id |
     |-------------------------------------------|
  1. | 1     2     3          0       0        1 |
  2. | 4     .     .          1       0        2 |
     |-------------------------------------------|
  3. | 1     2     3          .       1        1 |
  4. | 4   4.5   8.5          .       1        2 |
     |-------------------------------------------|
  5. | 1     2     3          .       2        1 |
  6. | 4   5.5   9.5          .       2        2 |
     +-------------------------------------------+
```

In addition to _mi_miss (which only has values in m=0, the original dataset), we also have _mi_m which records m (imputation number), _mi_id which records an arbitrarily coded observation-identification variable that allows to link observations across imputations (not to be used as a regular ID because mi program can arbitrarily change those numbers). . It is 1 and 2 in m=0, and then repeats in m=1 and m=2.

Style mlong

```
             +------------------------------------------+
             | a     b     c   _mi_miss   _mi_m   _mi_id |
             |------------------------------------------|
          1. | 1     2     3          0       0        1 |
          2. | 4     .     .          1       0        2 |
          3. | 4   4.5   8.5          .       1        2 |
          4. | 4   5.5   9.5          .       2        2 |
             +------------------------------------------+
```

Style flongsep

In this style, the data from m=0, m=1 and m=2 are stored in three separate datases, e.g.

    . mi convert flongsep example, clear
    (files example.dta _1_example.dta _2_example.dta created)

Here, example.dta looks like this:

```
             +-------------------------------+
             | a   b   c   _mi_miss   _mi_id |
             |-------------------------------|
          1. | 1   2   3          0        1 |
          2. | 4   .   .          1        2 |
             +-------------------------------+
```

_1_example.dta looks like this:

```
             +------------------------+
             | a     b     c   _mi_id |
             |------------------------|
          1. | 1     2     3        1 |
          2. | 4   4.5   8.5        2 |
             +------------------------+
```

_2_example.dta:

```
             +------------------------+
             | a     b     c   _mi_id |
             |------------------------|
          1. | 1     2     3        1 |
          2. | 4   5.5   9.5        2 |
             +------------------------+
```

Note that to change style, we used mi convert; another example:
mi convert flong, clear

Let’s set the style and register variables; I prefer flong so I will use that:

```
. mi set flong
. mi register imputed public pct_esl morale   sciencem femalem blackm latinom asianm
native
. mi register regular spokenm paredm
. mi impute chained (logit) public (ologit) morale (reg) pct_esl sciencem femalem
blackm latinom asianm native = spokenm paredm, add(5) augment

Conditional models:
            femalem: regress femalem blackm latinom asianm nativem i.public i.morale
sciencem pct_esl spokenm paredm
             blackm: regress blackm femalem latinom asianm nativem i.public i.morale
sciencem pct_esl spokenm paredm
            latinom: regress latinom femalem blackm asianm nativem i.public i.morale
sciencem pct_esl spokenm paredm
```

```
            asianm: regress asianm femalem blackm latinom nativem i.public i.morale
sciencem pct_esl spokenm paredm
           nativem: regress nativem femalem blackm latinom asianm i.public i.morale
sciencem pct_esl spokenm paredm
            public: logit public femalem blackm latinom asianm nativem i.morale
sciencem pct_esl spokenm paredm , augment
            morale: ologit morale femalem blackm latinom asianm nativem i.public
sciencem pct_esl spokenm paredm , augment
          sciencem: regress sciencem femalem blackm latinom asianm nativem i.public
i.morale pct_esl spokenm paredm
           pct_esl: regress pct_esl femalem blackm latinom asianm nativem i.public
i.morale sciencem spokenm paredm

Performing chained iterations ...

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

Initialization: monotone                     Iterations =       50
                                                burn-in =       10

            public: logistic regression
            morale: ordered logistic regression
           pct_esl: linear regression
          sciencem: linear regression
           femalem: linear regression
            blackm: linear regression
           latinom: linear regression
            asianm: linear regression
           nativem: linear regression

------------------------------------------------------------------
                   |               Observations per m
                   |----------------------------------------------
          Variable |   Complete   Incomplete   Imputed |     Total
-------------------+-----------------------------------+----------
            public |        996           17        17 |      1013
            morale |        994           19        19 |      1013
           pct_esl |        991           22        22 |      1013
          sciencem |        992           21        21 |      1013
           femalem |       1011            2         2 |      1013
            blackm |       1010            3         3 |      1013
           latinom |       1010            3         3 |      1013
            asianm |       1010            3         3 |      1013
           nativem |       1010            3         3 |      1013
------------------------------------------------------------------
(complete + incomplete = total; imputed is the minimum across m
 of the number of filled-in observations.)
```

Note that if any of your variables that are used but not imputed (variables after =) should be sets of dummies, use i. before these variables' names in the command.

<u>Some helpful options of mi impute include:</u>
dryrun – same as for ice; specify equations but not run the actual imputation
rseed(#) – specify random-number seed in order to create imputations in a replicable fashion
by(varlist) – impute separately on each group formed by varlist
conditional(if condition)
include – specify what to include in the imputation equation
omit – to omit certain variables from some equations