

SC705: Advanced Statistics
Instructor: Natasha Sarkisian
Course notes: 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. Here is an example using NELS (National Educational Longitudinal Survey):

```
. tab bys12
```

-> 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. A few tools are available in Stata to examine the patterns of missing data; all are user-written packages that you'd need to download. The most comprehensive one is `mvpatterns`.

```
. net search mvpatterns
(contacting http://www.stata.com)

1 package found (Stata Journal and STB listed first)
-----
```

dm91 from <http://www.stata.com/stb/stb61>
 STB-61 dm91. Patterns of missing values / STB insert by Jeroen Weesie,
 Utrecht University, Netherlands / Support: J.Weesie@fss.uu.nl / After
 installation, see help mvpatterns

```
. mvpatterns female spoken
Variable      | type      obs   mv   variable label
-----+-----
female       | float    13703  119
spoken       | float    13762   60
```

Patterns of missing values

```
+-----+
| _pattern  _mv  _freq |
+-----+
|      ++    0  13645 |
|      .+    1   117 |
|      +.    1    58 |
|      ..    2     2 |
+-----+
```

Note: The maximum number of variables you can include is 80.

To examine whether the missingness is related to other variables in the dataset, you can generate dummy indicators for missing values and examine whether they are related to other variables. For example:

```
. ttest female, by(spokenm)

Two-sample t test with equal variances
-----+-----
      Group |      Obs      Mean      Std. Err.      Std. Dev.      [95% Conf. Interval]
-----+-----
          0 |  13645   .5135214   .004279   .4998355   .505134   .5219088
          1 |     58   .4310345   .0655936   .4995461   .2996855   .5623834
-----+-----
combined |  13703   .5131723   .00427   .4998447   .5048025   .5215421
-----+-----
      diff |           .082487   .0657708           -.0464328   .2114067
-----+-----
      diff = mean(0) - mean(1)                                t = 1.2542
Ho: diff = 0                                                  degrees of freedom = 13701

      Ha: diff < 0              Ha: diff != 0              Ha: diff > 0
Pr(T < t) = 0.8951             Pr(|T| > |t|) = 0.2098             Pr(T > t) = 0.1049
```

Methods of handling missing data

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.

2. Pairwise deletion

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.

3. Missing data indicators

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.

II. Deterministic imputation methods

1. Mean substitution

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.

2. Single regression imputation (a.k.a. conditional mean substitution)

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 13 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 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. It is also implemented as a default procedure in AMOS, a structural equation modeling package. Other software packages that use the raw maximum likelihood approach to handle incomplete data are the MIXED procedure in SAS and Michael Neale's MX -- a freeware structural equation modeling program; you can download it at <http://views.vcu.edu/mx>. So far, direct ML methods have not been implemented in Stata.

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 $\text{SQRT}[(1 - 1/m)/B + W]$, 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.

Software for multiple imputation is tracked on two websites:
www.stat.psu.edu/~jls/misoftwa.html and www.multiple-imputation.com.

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). A freeware program implementing the direct augmentation approach to multiple imputation, NORM, written by the person who developed this approach, Schaefer, is available at <http://www.stat.psu.edu/~jls/>. Another freeware program similar to NORM called Amelia may also be downloaded from <http://gking.harvard.edu/stats.shtml#amelia>. This can also be implemented through Proc MI as part of the SAS package. 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; two versions of it are available in Stata – a user-written module ICE and built-in command (in Stata 12 only), `mi impute chained`. 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.

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 be included in the imputation process; there are two approaches as to what to do with them afterwards. Until recently, most scholars suggested that after imputation, the imputed values of the dependent variable itself should be discarded (this is called MID—multiple imputation then deletion). The conventional wisdom was that the only exception to that rule is a situation when additional variables were used in imputing the 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 be preserved. However, recently scholars have been suggesting that in most cases, it is better to keep the imputed values of the dependent variable as well.
- If you have interactions or nonlinear relationships among the variables, you should create the variables for those before doing the imputations and use them in the imputation process.
- 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.
- 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, 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.
- 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.
- 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.

- When imputing multilevel datasets that are not longitudinal, we need to include information about clusters. The most common method 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 would want to include interactions of these cluster dummies with each of the independent variables that should have random slopes, although this can get very complicated very fast. (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.
- 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 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.

Learning to use MICE

You find the command using

```
. net search ice
```

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

[If you are using Stata 12, it also has an integrated module for ICE – the command for that is `mi impute chained`; the idea is the same although the syntax is somewhat different; see `help mi impute chained` in Stata for examples.]

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). Let's see an example, and then we'll examine a few options.

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

```

Multiple imputation for level 2:

```

. 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

```

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.

Some 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. 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).

Now that we imputed level 2 variables, we can merge them with our level 1 dataset 10 times:

```

. use "C:\Users\sarkisin\nels_imputed.dta", clear

. tab _mj
  imputation |
  number |      Freq.      Percent      Cum.
-----+-----
      0 |      1,013      16.67      16.67
      1 |      1,013      16.67      33.33
      2 |      1,013      16.67      50.00
      3 |      1,013      16.67      66.67
      4 |      1,013      16.67      83.33
      5 |      1,013      16.67     100.00
-----+-----
  Total |      6,078     100.00

. keep if _mj==1
(5065 observations deleted)

. merge 1:m sch_id using "C:\Users\sarkisin\Documents\Teaching Grad
Statistics\SC708\nels_science.dta"

  Result                                     # of obs.
-----+-----
not matched                                     0
matched                                     13,822  (_merge==3)
-----+-----

. tab _merge

      _merge |      Freq.      Percent      Cum.
-----+-----
  matched (3) |      13,822     100.00     100.00
-----+-----
      Total |      13,822     100.00

. drop _merge _mi _mj
. save "C:\Users\sarkisin\nels_imputed_1.dta"

```

As we have a group of dummies included in this imputation (race variables), we need to use passive and substitute options to deal with that.

passive(passivelist) allows the use of "passive" imputation of variables that depend on other variables, some of which are imputed. The syntax of passivelist is varname:exp [\varname:exp ...]. Notice the requirement to use "\" as a separator between items in passivelist, rather than the usual comma; the reason is that a comma may be a valid part of an expression.

When we do the imputation, we will impute race5 using mlogit command, and we want black, latino, asian, and native to be updated with new values based on the imputed values of race5.

This may be achieved by specifying `passive(black: race5==2\ latino: race5==3 \ asian: race5==4 \native: race5==5)`. It is also necessary to remove race5 from the list of predictors when variables other than race5 are being imputed – we want to use dummies in such imputations. This is done by using the substitute() option; in the present example, we would specify `substitute(race5: black latino asian native)`.

Note that all the race dummies (black latino asian native) should be specified in the mainvarlist; otherwise, they will not be recognized.

If we are using interactions, for example, between x1 and x2 (e.g., $x12=x1*x2$), we should also enter `passive(x12:x1*x2)`.

Since we are using a lot of dummies, we are likely to run into errors, especially in logit-based models. You can tell Stata to ignore those errors; 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. The syntax of eqdroplist is `varname1:varlist1 [, varname2:varlist2 ...]`.

For example, to run the following command, I need to use Stata SE rather than Stata IC. Stata SE is available at apps.bc.edu – you would want to specify larger amount of memory and larger matsize:

```
. set matsize 1200
. set mem 200m
. use "C:\Users\sarkisin\nels_imputed_1.dta", clear
xi: ice science female spoken pared race5 black latino asian native i.sch_id,
passive(black: race5==2\ latino: race5==3 \ asian: race5==4 \native: race5==5)
substitute(race5: black asian latino native) cmd(spoken: ologit)
saving("C:\Users\sarkisin\nels_science_set1.dta", replace) m(1) persist
```

I get the following errors:

```
Imputing
[Note: in regression for pared, dropping _Isch_ida45939 due to collinearity]
[persist option: ignoring error #506, not updating pared in cycle 1]
```

```
[Note: in regression for female, dropping _Isch_ida45222 due to collinearity]
[Perfect prediction detected: using auglogit to impute female]
[Note: in regression for race5, dropping _Isch_ida7658 _Isch_ida45222 due to
collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 1]
[Note: in regression for science, dropping _Isch_ida45939 _Isch_ida77219 due to
collinearity]
[persist option: ignoring error #506, not updating science
> in cycle 1]
.
[Note: in regression for pared, dropping _Isch_ida29757 _Is
> ch_ida77219 due to collinearity]
[persist option: ignoring error #506, not updating pared in
> cycle 2]
[Note: in regression for female, dropping _Isch_ida45222 du
> e to collinearity]
[Note: in regression for race5, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 2]
[Note: in regression for science, dropping _Isch_ida45939 due to collinearity]
[persist option: ignoring error #506, not updating science in cycle 2]
.
[Note: in regression for pared, dropping _Isch_ida77219 due to collinearity]
[persist option: ignoring error #506, not updating pared in cycle 3]
[Note: in regression for female, dropping _Isch_ida45222 due to collinearity]
[Note: in regression for race5, dropping _Isch_ida45939 due to collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 3]
[Note: in regression for science, dropping _Isch_ida45939 due to collinearity]
[persist option: ignoring error #506, not updating science in cycle 3]
.
[Note: in regression for pared, dropping _Isch_ida45939 due to collinearity]
[persist option: ignoring error #506, not updating pared in cycle 4]
[Note: in regression for female, dropping _Isch_ida45222 due to collinearity]
[Note: in regression for race5, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 4]
[Note: in regression for science, dropping _Isch_ida45939 due to collinearity]
[persist option: ignoring error #506, not updating science in cycle 4]
.
[Note: in regression for pared, dropping _Isch_ida45939 due to collinearity]
[persist option: ignoring error #506, not updating pared in cycle 5]
```

[Note: in regression for female, dropping _Isch_ida45222 due to collinearity]
[Note: in regression for race5, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 5]
[Note: in regression for science, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #506, not updating science in cycle 5]
.
[Note: in regression for pared, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #506, not updating pared in cycle 6]
[Note: in regression for female, dropping _Isch_ida45222 due to collinearity]
[Note: in regression for race5, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 6]
[Note: in regression for science, dropping _Isch_ida77219 due to collinearity]
[persist option: ignoring error #506, not updating science in cycle 6]
.
[Note: in regression for pared, dropping _Isch_ida77219 due to collinearity]
[persist option: ignoring error #506, not updating pared in cycle 7]
[Note: in regression for female, dropping _Isch_ida45222 due to collinearity]
[Note: in regression for race5, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 7]
[Note: in regression for science, dropping _Isch_ida77219 due to collinearity]
[persist option: ignoring error #506, not updating science in cycle 7]
.
[Note: in regression for pared, dropping _Isch_ida77219 due to collinearity]
[persist option: ignoring error #506, not updating pared in cycle 8]
[Note: in regression for female, dropping _Isch_ida45222 due to collinearity]
[Note: in regression for race5, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 8]
[Note: in regression for science, dropping _Isch_ida77219 due to collinearity]
[persist option: ignoring error #506, not updating science in cycle 8]
.
[Note: in regression for pared, dropping _Isch_ida45939 due to collinearity]
[persist option: ignoring error #506, not updating pared in cycle 9]
[Note: in regression for female, dropping _Isch_ida45222 due to collinearity]
[Note: in regression for race5, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 9]

```

[Note: in regression for science, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #506, not updating science in cycle 9]
.
[Note: in regression for pared, dropping _Isch_ida45222 due to collinearity]
[persist option: ignoring error #506, not updating pared in cycle 10]

[Note: in regression for female, dropping _Isch_ida45222 due to collinearity]
[Note: in regression for race5, dropping _Isch_ida45939 due to collinearity]
[persist option: ignoring error #908, not updating race5 in cycle 10]

[Note: in regression for science, dropping _Isch_ida77219 due to collinearity]

[persist option: ignoring error #506, not updating science in cycle 10]
.1
(note: file C:\Users\sarkisin\nels_science_set1.dta not found)
file C:\Users\sarkisin\nels_science_set1.dta saved

```

Note that I omitted level 2 variables because I am including dummies for schools which would be collinear with level 2 variables. Given the errors, I will then try omitting some dummies. I'll start with one:

```

xi: ice science female spoken pared race5 black latino asian native i.sch_id,
passive(black: race5==2\ latino: race5==3 \ asian: race5==4 \native: race5==5)
substitute(race5: black asian latino native) cmd(spoken: ologit)
saving("C:\Users\sarkisin\nels_science_set1.dta", replace) m(1) eqdrop(race5:
Isch_ids45939, pared: Isch_ids45939, science: Isch_ids45939, female: Isch_ids45939,
spoken: Isch_ids45939)

```

I would keep adjusting the imputation model until it runs without errors. Note that imputation is often a pretty slow process, especially if your dataset is large. If the number of level 2 units is very large and even after omitting a number of dummies, the model generates errors, you might be better off including level 2 variables instead of cluster dummies. For example, here we will include level 2 variables:

```

. xi: ice public pct_esl morale science female spoken pared race5 black latino asian
native, passive(black: race5==2\ latino: race5==3 \ asian: race5==4 \native: race5==5)
substitute(race5: black asian latino native) cmd(spoken: ologit)
saving("C:\Users\sarkisin\nels_science_imp1.dta", replace) m(1)

```

#missing	values	Freq.	Percent	Cum.
	0	13,089	94.70	94.70
	1	553	4.00	98.70
	2	27	0.20	98.89
	5	153	1.11	100.00
	Total	13,822	100.00	

Variable	Command	Prediction equation
public		[No missing data in estimation sample]
pct_esl		[No missing data in estimation sample]
morale		[No missing data in estimation sample]
science	regress	public pct_esl morale female spoken pared black latino asian native
female	logit	public pct_esl morale science spoken pared black

```

        |         | latino asian native
spoken | ologit | public pct_esl morale science female pared black
        |         | latino asian native
pared  | regress | public pct_esl morale science female spoken black
        |         | latino asian native
race5  | mlogit  | public pct_esl morale science female spoken pared
black  |         | [Passively imputed from race5==2]
latino |         | [Passively imputed from race5==3]
asian  |         | [Passively imputed from race5==4]
native |         | [Passively imputed from race5==5]
-----

```

```

Imputing .....1
file C:\Users\sarkisin\nels_science_imp1.dta saved

```

We would repeat this process 5 times using different imputed values of level 2 variables every time, e.g.:

```

. use "C:\Users\sarkisin\nels_imputed.dta", clear

. keep if _mj==2
(5065 observations deleted)

. merge 1:m sch_id using "C:\Users\sarkisin\Documents\Teaching Grad
Statistics\SC708\nels_science.dta"

```

Result	# of obs.
not matched	0
matched	13,822 (_merge==3)

```

. tab _merge

```

_merge	Freq.	Percent	Cum.
matched (3)	13,822	100.00	100.00
Total	13,822	100.00	

```

. drop _merge _mi _mj
. save "C:\Users\sarkisin\nels_imputed_2.dta"
. xi: ice public pct_esl morale science female spoken pared race5 black latino asian
native, passive(black: race5==2\ latino: race5==3 \ asian: race5==4 \native: race5==5)
substitute(race5: black asian latino native) cmd(spoken: ologit)
saving("C:\Users\sarkisin\nels_science_imp2.dta", replace) m(1)

```

#missing	values	Freq.	Percent	Cum.
0	0	13,089	94.70	94.70
1	1	553	4.00	98.70
2	2	27	0.20	98.89
5	5	153	1.11	100.00
Total		13,822	100.00	

Variable	Command	Prediction equation
public		[No missing data in estimation sample]
pct_esl		[No missing data in estimation sample]
morale		[No missing data in estimation sample]
science	regress	public pct_esl morale female spoken pared black latino asian native

```

female | logit   | public pct_esl morale science spoken pared black
        |          | latino asian native
spoken | ologit  | public pct_esl morale science female pared black
        |          | latino asian native
pared  | regress | public pct_esl morale science female spoken black
        |          | latino asian native
race5  | mlogit  | public pct_esl morale science female spoken pared
black  |          | [Passively imputed from race5==2]
latino |          | [Passively imputed from race5==3]
asian  |          | [Passively imputed from race5==4]
native |          | [Passively imputed from race5==5]
-----

```

```

Imputing .....1
file C:\Users\sarkisin\nels_science_imp2.dta saved

```

To obtain final estimates of the parameters of interest and their standard errors, one would fit a model in each imputation and carry out the appropriate post-MI averaging procedure on the results. In Stata, you can do that using `mi estimate` -- you would merge all the imputed datasets into a single file for that, and make sure each imputation is appropriately marked in the `_mj` variable. You would also have to do `mi import` to convert your data from MICE format into the format used by MI commands (see help `mi import`).

But we will use HLM: start with one dataset, then select Other Settings → Estimation Settings → Multiple imputation, then select up to 10 datasets. After that, estimate the model as you normally would. Note that it is better to do exploratory runs on a single imputation--that would be much faster. But the results could end up being different.

If you are using some other program that does not support multiple imputation estimation, you can estimate the model separately for each dataset and then combine the coefficients. Specifically, the coefficients would be simple averages of coefficients from separate regression models, and the standard errors can be calculated using Rubin's (1987) formula:

$$\sqrt{\frac{1}{M} \sum_k s_k^2 + \left(1 + \frac{1}{M}\right) \left(\frac{1}{M-1}\right) \sum_k (b_k - \bar{b})^2}$$

where b_k is the estimated regression coefficient in the imputed dataset k of the M imputed datasets, and s_k is its estimated standard error. \bar{b} is the average of coefficients across M imputations.

Essentially, we calculate the average squared standard error and then add to it the variance of coefficient estimates (multiplied by a correction factor $1 + 1/M$).

Methods for nonignorable missing data

All the methods of missing data handling considered above require that the data meet the MAR assumption. There are circumstances, however, when cases are considered missing due to non-ignorable causes. In such instances the investigator may want to consider the use of a selection model or a pattern-mixture model.

1. Selection models.

Social researchers have traditionally dealt with NMAR data by using selection models. In a selection model, you simultaneously model Y and the probability that Y is missing. These models are implemented in `heckman` and `heckprob` in Stata and we will discuss these times of models later on this semester.

2. Pattern mixture models.

An alternative to selection models is multiple imputation with pattern mixture. In this approach, you perform multiple imputations under a variety of assumptions about the missing data mechanism.

Pattern-mixture models categorize the different patterns of missing values in a dataset into a predictor variable, and this predictor variable is incorporated into the statistical model of interest. The investigator can then determine if the missing data pattern has an predictive power in the model, either by itself (a main effect) or in conjunction with another predictor (an interaction effect).

In ordinary multiple imputation, you assume that those people who report their weights are similar to those who don't. In a pattern-mixture model, you may assume that people who don't report their weights are an average of 20 pounds heavier. This is of course an arbitrary assumption; the idea of pattern mixture is to try out a variety of plausible assumptions and see how much they affect your results.

Although pattern mixture is more natural, flexible, and interpretable approach, it appears that social researchers more often use selection models – partly because of tradition, partly because they are easier to use. Pattern mixture models can be implemented in SAS using `PROC MI` or `PROC MIXED`, but still, this requires some custom programming. Also, if the number of missing data patterns and the number of variables with missing data are large relative to the number of cases in the analysis, the model may not converge due to insufficient data.