Models In Epidemiology And Biostatistics
Gordon Hilton Fick

Session 17 : The Basics of Missing Data

Introduction [adapted from http://www.ats.ucla.edu/stat/Stata/modules/missing.html]

We will now explore the basics of missing data in Stata, focusing on numeric missing data. It will describe how to indicate missing data in your raw data files, as well as how missing data are handled in Stata logical commands and assignment statements.

We will illustrate some of the missing data properties in Stata using data from a reaction time study with eight subjects indicated by the variable id , and the subjects reaction times were measured at three time points (trial1 trial2 trial3). The input data file is shown below.

```
input id trial1 trial2 trial3

    1 1.5 1.4 1.6

    2 1.5 .   1.9

    3 .   2.0 1.6

    4 .   .   2.2

    5 1.9 2.1 2

    6 1.8 2.0 1.9

    7 .    .  .

    end

list
```

You might notice that some of the reaction times are coded using a single . as is the case for subject 2. The person measuring time for that trial did not measure the response time properly, therefore the data for the second trial is missing.

```
     +-------------------------------+
     | id   trial1   trial2   trial3 |
     |-------------------------------|
  1. |  1      1.5      1.4      1.6 |
  2. |  2      1.5        .      1.9 |
  3. |  3        .        2      1.6 |
  4. |  4        .        .      2.2 |
  5. |  5      1.9      2.1        2 |
     |-------------------------------|
  6. |  6      1.8        2      1.9 |
  7. |  7        .        .        . |
     +-------------------------------+
```

How Stata handles missing data in Stata procedures

As a general rule, Stata commands that perform computations of any type handle missing data by omitting the missing values. However, the way that missing values are omitted is not always consistent across commands, so let's take a look at some examples.

First, let's summarize our reaction time variables and see how Stata handles the missing values.

```
summarize trial1 trial2 trial3
```

As you see in the output below, summarize computed means using 4 observations for trial1 and trial2 and 6 observations for trial3. In short, the summarize command performed the computations on all the available data.

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      trial1 |         4       1.675    .2061553        1.5        1.9
      trial2 |         4       1.875    .3201562        1.4        2.1
      trial3 |         6    1.866667     .233809        1.6        2.2
```

A second example, shows how the tabulation or tab1 command handles missing data. Like summarize, tab1 uses just available data. Note that the percentages are computed based on the total number of non-missing cases.

```
tab1 trial1 trial2 trial3

    -> tabulation of trial1

     trial1 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.5 |          2       50.00       50.00
        1.8 |          1       25.00       75.00
        1.9 |          1       25.00      100.00
------------+-----------------------------------
      Total |          4      100.00

    -> tabulation of trial2

     trial2 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.4 |          1       25.00       25.00
          2 |          2       50.00       75.00
        2.1 |          1       25.00      100.00
------------+-----------------------------------
      Total |          4      100.00

    -> tabulation of trial3

     trial3 |      Freq.     Percent        Cum.
------------+-----------------------------------
        1.6 |          2       33.33       33.33
        1.9 |          2       33.33       66.67
          2 |          1       16.67       83.33
        2.2 |          1       16.67      100.00
------------+-----------------------------------
      Total |          6      100.00
```

It is possible that you might want the percentages to be computed out of the total number of observations, and the percentage missing for each variable shown in the table. This can be achieved by including the missing option after the tabulation. command,

```
tab1 trial1 trial2 trial3, m

    -> tabulation of trial1

         trial1 |      Freq.     Percent        Cum.
    ------------+-----------------------------------
            1.5 |          2       28.57       28.57
            1.8 |          1       14.29       42.86
            1.9 |          1       14.29       57.14
              . |          3       42.86      100.00
    ------------+-----------------------------------
          Total |          7      100.00

    -> tabulation of trial2

         trial2 |      Freq.     Percent        Cum.
    ------------+-----------------------------------
            1.4 |          1       14.29       14.29
              2 |          2       28.57       42.86
            2.1 |          1       14.29       57.14
              . |          3       42.86      100.00
    ------------+-----------------------------------
          Total |          7      100.00

    -> tabulation of trial3

         trial3 |      Freq.     Percent        Cum.
    ------------+-----------------------------------
            1.6 |          2       28.57       28.57
            1.9 |          2       28.57       57.14
              2 |          1       14.29       71.43
            2.2 |          1       14.29       85.71
              . |          1       14.29      100.00
    ------------+-----------------------------------
          Total |          7      100.00
```

Let's look at how the correlate command handles missing data. We would expect that it would perform the computations based on the available data, and omit the missing values. Here is an example command.

```
corr trial1 trial2 trial3
```

The output is show below. Note how the missing values were excluded. Stata will perform listwise deletion and only display correlation for observations that have non-missing values on all variables listed.

```
corr trial1 trial2 trial3
    (obs=3)

    | trial1 trial2 trial3
    -------------+---------------------------
    trial1 | 1.0000
    trial2 | 0.9939 1.0000
    trial3 | 1.0000 0.9939 1.0000
```

Stata also allows for pairwise deletion. Correlations are displayed for the observations that have non-missing values for each pair of variables. This can done using the pwcorr command. We use the obs option to display the number of observation used for each pair, as you can see, they differ depending on the amount of missing.

```
pwcorr trial1 trial2 trial3, obs

             |   trial1   trial2   trial3
-------------+---------------------------
      trial1 |   1.0000
             |        3
             |
      trial2 |   0.9939   1.0000
             |        3        4
             |
      trial3 |   0.7001   0.6439   1.0000
             |        4        4        6
```

Summary of how missing values are handled in Stata procedures

summarize : For each variable, the number of non-missing values are used.

tabulation : By default, missing values are excluded and percentages are based on the number of non-missing values. If you use the missing option on the tab command, the percentages are based on the total number of observations (non-missing and missing) and the percentage of missing values are reported in the table.

corr : By default, correlations are computed based on the number of pairs with non-missing data (pairwise deletion of missing data). The pwcorr command can be used to request that correlations be computed only for observations that have non-missing data for all variables listed after the pwcorr command (listwise deletion of missing data).

regress, logit, ologit, mlogit [all modeling commands] If any of the variables listed after the modeling command are missing, the observations missing that value(s) are excluded from the analysis (i.e., listwise deletion of missing data).

For other procedures, see the Stata manual for information on how missing data are handled.

Missing values in assignment statements

It is important to understand how missing values are handled in assignment statements. Consider the example shown below.

```
gen sum1 = trial1 + trial2 + trial3
```

The list command below illustrates how missing values are handled in assignment statements. The variable sum1 is based on the variables trial1 trial2 and trial3. If any of those variables were missing, the value for sum1 was set to missing. Therefore sum1 is missing for observations 2, 3 and 4, as is the case for observation 7.

```
list
```

```
     +-------------------------------------+
     | id   trial1   trial2   trial3   sum1 |
     |-------------------------------------|
  1. |  1      1.5      1.4      1.6    4.5 |
  2. |  2      1.5        .      1.9      . |
  3. |  3        .        2      1.6      . |
  4. |  4        .        .      2.2      . |
  5. |  5      1.9      2.1        2      6 |
     |-------------------------------------|
  6. |  6      1.8        2      1.9    5.7 |
  7. |  7        .        .        .      . |
     +-------------------------------------+
```

As a general rule, computations involving missing values yield missing values. For example,

```
2 + 2 yields 4
2 + . yields .
2 / 2 yields 1
. / 2 yields .
2 * 3 yields 6
2 * . yields .
```

whenever you add, subtract, multiply, divide, etc., values that involve missing data, the result is missing.

In our reaction time experiment, the total reaction time sum1 is missing for four out of seven cases. We could try totaling the data for the non-missing trials by using the rowtotal function as shown in the example below.

```
egen sum2 = rowtotal(trial1 trial2 trial3)

list
```

The results below show that sum2 now contains the sum of the non-missing trials.

```
     +---------------------------------------------+
     | id   trial1   trial2   trial3   sum1   sum2 |
     |---------------------------------------------|
  1. |  1      1.5      1.4      1.6    4.5    4.5 |
  2. |  2      1.5        .      1.9      .    3.4 |
  3. |  3        .        2      1.6      .    3.6 |
  4. |  4        .        .      2.2      .    2.2 |
  5. |  5      1.9      2.1        2      6      6 |
     |---------------------------------------------|
  6. |  6      1.8        2      1.9    5.7    5.7 |
  7. |  7        .        .        .      .      0 |
     +---------------------------------------------+
```

Note that the rowtotal function treats missing as a zero value. When summing several variables it may not be reasonable to treat missing as zero if an observations is missing on all variables to be summed. The rowtotal function with the missing option will return a missing value if an observation is missing on all variables.

```
egen sum3 = rowtotal(trial1 trial2 trial3) , missing
```

	id	trial1	trial2	trial3	sum1	sum2	sum3
1.	1	1.5	1.4	1.6	4.5	4.5	4.5
2.	2	1.5	.	1.9	.	3.4	3.4
3.	3	.	2	1.6	.	3.6	3.6
4.	4	.	.	2.2	.	2.2	2.2
5.	5	1.9	2.1	2	6	6	6
6.	6	1.8	2	1.9	5.7	5.7	5.7
7.	7	.	.	.	.	0	.

Other statements work similarly. For example, observed what happened when we try to create an average variable without using a function (as in the example below). If any of the variables trial1, trial2 or trial3 are missing, the value for avg1 are set to missing.

```
gen avg1 = (trial1 + trial2 + trial3)/3
```

Alternatively, the rowmean function averages the data for the non-missing trials in the same way as the rowtotal function.

```
egen avg2 = rowmean(trial1 trial2 trial3)
```

Note: Had there been large number of trials, say 50 trials, then it would be annoying to have to type avg=rowmean(trial1 trial2 trial3 trial4 ...). Here is a shortcut you could use in this kind of situation:

```
egen avg3 = rowmean(trial1 - trial3)

list
```

	id	trial1	trial2	trial3	avg1	avg2	avg3
1.	1	1.5	1.4	1.6	1.5	1.5	1.5
2.	2	1.5	.	1.9	.	1.7	1.7
3.	3	.	2	1.6	.	1.8	1.8
4.	4	.	.	2.2	.	2.2	2.2
5.	5	1.9	2.1	2	2	2	2
6.	6	1.8	2	1.9	1.9	1.9	1.9
7.	7	.	.	.	.	.	.

Finally, you can use the rowmiss and rownomiss functions to determine the number of missing and the number of non-missing values, respectively, in a list of variables. This is illustrated below.

```
egen miss = rowmiss(trial1 - trial3)

egen nomiss = rownonmiss(trial1 - trial3)

list
```

For variable nomiss, observations 1, 5 and 6 had three valid values, observations 2 and 3 had two valid values, observation 4 had only one valid value and observation 7 had no valid values. The variable miss shows the opposite, it provides a count of the number of missing values.

```
     +---------------------------------------------------+
     | id   trial1   trial2   trial3   miss   nomiss |
     |---------------------------------------------------|
  1. |  1      1.5      1.4      1.6      0        3 |
  2. |  2      1.5        .      1.9      1        2 |
  3. |  3        .        2      1.6      1        2 |
  4. |  4        .        .      2.2      2        1 |
  5. |  5      1.9      2.1        2      0        3 |
     |---------------------------------------------------|
  6. |  6      1.8        2      1.9      0        3 |
  7. |  7        .        .        .      3        0 |
     +---------------------------------------------------+
```

Missing values in logical statements

It is important to understand how missing values are handled in logical statements. For example, say that you want to create a 0/1 variable for trial1 that is 1 if it is 1.5 or less, and 0 if it is over 1.5. We show this below (incorrectly, as you will see).

```
gen newvar1 =(trial2 <1.5)

list trial2 newvar1
```

It appears that something went wrong with our newly created variable newvar1! The observations with missing values for trial2 were assigned a zero for newvar1.

```
     +------------------+
     | trial2   newvar1 |
     |------------------|
  1. |    1.4         1 |
  2. |      .         0 |
  3. |      2         0 |
  4. |      .         0 |
  5. |    2.1         0 |
     |------------------|
  6. |      2         0 |
  7. |      .         0 |
     +------------------+
```

Let's explore why this happened by looking at the frequency table of trial2.

As you can see in the output, missing values are at the listed after the highest value 2.1 This is because Stata treats a missing value as the largest possible value (e.g., positive infinity) and that value is greater than 2.1, so then the values for newvar1 become 0.

```
tab trial2, missing

      trial2 |      Freq.     Percent        Cum.
-------------+-----------------------------------
         1.4 |          1       14.29       14.29
           2 |          2       28.57       42.86
         2.1 |          1       14.29       57.14
           . |          3       42.86      100.00
-------------+-----------------------------------
       Total |          7      100.00
```

Now that we understand how Stata treats missing values, we will explicitly exclude missing values to make sure they are treated properly, as shown below.

```
gen newvar2 =(trial2 <1.5) if trial2 !=.

list trial2 newvar1 newvar2
```

As you can see in the Stata output below, the new variable newvar2 has missing values for observations that are also missing for trial2.

```
     +----------------------------+
     | trial2   newvar1   newvar2 |
     |----------------------------|
  1. |    1.4         1         1 |
  2. |      .         0         . |
  3. |      2         0         0 |
  4. |      .         0         . |
  5. |    2.1         0         0 |
     |----------------------------|
  6. |      2         0         0 |
  7. |      .         0         . |
     +----------------------------+
```

Missing values in logical statements

When creating or recoding variables that involve missing values, always pay attention to whether the variable includes missing values.

Missing values are handled quite differently in R. For example, in R, missing values are represented by the symbol NA (not available). Impossible values (e.g., dividing by zero) are represented by the symbol NaN (not a number). There are many online tutorials on missing values in R. This is a big topic with many subtle differences between software. The above material in this session is, in places, quite specific to Stata and can be very different in R.

Old Approaches to Missing Data: [adapted from CK Anders 'Applied Missing Data Analysis']

Listwise and pairwise deletion might seem like a sensible if pragmatic way to analyses with missing data. Deletion methods have serious limitations that preclude their use in most situations. Most importantly, these approaches make assumptions about the available data and can produce distorted parameter estimates. There is little to recommend these techniques unless the proportion of missing data is trivially small.

Listwise Deletion:
Listwise deletion (also known as complete-case analysis) discards the data for any case that has one or more missing values. Relative to pairwise deletion, listwise deletion also has the advantage of producing a common set of cases for all analyses. In most situations, the disadvantages of listwise deletion far outweigh its advantages. The primary problem with listwise deletion is that it requires assumptions about the available data and can produce distorted parameter estimates when these assumptions do not hold.

As an example, consider a study where the participants in the lower half of the IQ distribution have missing job performance ratings. By virtue of this selection process, listwise deletion discards the entire lower half of the IQ distribution. Because IQ scores and job performance ratings are presumably associated, listwise deletion also excludes cases from the lower tail of the job performance distribution (i.e., the cases with low IQ scores). Not surprisingly, the remaining cases are unrepresentative of the hypothetically complete data set. They have higher scores on both variables. Consequently, with listwise deletion, estimates of the mean are too high. In addition, the restriction of range that results from discarding the lower tails of the distributions makes the estimates of the variability of the data too small and would understimate the correlation between these two variables. Bias aside, listwise deletion is potentially very wasteful, particularly when the discarded cases have data on a large number of variables. Deleting the incomplete data records can produce a dramatic reduction in the total sample size, the magnitude of which increases as the missing data rate or number of variables increases. For example, consider a data set with 10 variables, each of which has 2% of its observations missing in a completely random fashion. Although the proportion of missing data on any single variable is relatively small, listwise deletion could eliminate [an expected percentage of]18% of the data records. With 20 variables, the expected percentage of complete cases drops to about 67%.

Pairwise Deletion
Pairwise deletion (also known as available-case analysis) attempts to mitigate the loss of data by eliminating cases on an analysis-by-analysis basis. It is common to find published research articles that report varying sample sizes across a set of analyses. Using as much of the data as possible might sound like a good idea. Alas, the issues discussed with listwise deletion afflict pairwise deletion as well. Further, pairwise deletion also has a number of unique problems. Using different subsets of participants poses subtle problems with many analyses. For example, one might have two logistic regression analyses that, on the basis of their model formulations, would appear to be nested but if the fits are based on different sets of participants, the models are not in fact nested.

Single Imputation
So-called single imputation methods attempt to fill in the missing data with the actual data prior to analysis. By single imputation we mean approaches generate a single replacement value for each missing data point. This is in contrast to multiple imputation, which creates several copies of the data set and imputes each copy with different plausible estimates of the missing values. Imputation then gives us a complete [but not real] data set. Imputation then enables the use of data that deletion

approaches would otherwise discard. Single imputation techniques have potentially serious drawbacks. Bias in the estimates remain [for the most part] an issue. In addition, single imputation techniques attenuate standard errors. Analyzing a single imputed data set effectively treats the filled-in values as real data, so single imputation techniques result in analyses that underestimate standard errors. Multiple imputation can often address this problem.

Imputation is the process of replacing missing data with substituted values.

im·pute [origin: 16th century]
1. To relate (something, usually something bad) to a particular cause or source; place the fault or responsibility for: imputed the rocket failure to a faulty gasket; kindly imputed my clumsiness to inexperience. See Synonyms at attribute.
2. To assign as a characteristic; credit: the gracefulness so often imputed to cats.

1. to attribute or ascribe (something dishonest or dishonourable, esp a criminal offence) to a person
2. to attribute to a source or cause: I impute your success to nepotism.
3. (Commerce) commerce to give (a notional value) to goods or services when the real value is unknown

Arithmetic Mean Imputation
Arithmetic mean imputation (also referred to as mean substitution and unconditional mean imputation) takes the seemingly appealing tack of filling in the missing values with the arithmetic mean of the available cases. The idea of replacing missing values with the mean is an old one that methodologists often attribute to Wilks (1932). In essentially all settings, this approach severely distorts the resulting parameter estimates. Mean imputation will attenuate the standard errors. This approach has been studied at length [from the 70's onwards]. In fact, many of these studies suggest that mean imputation is possibly the worst missing data handling method available. Consequently, in no situation is mean imputation defensible, and you should absolutely avoid this approach.

Regression Imputation
Regression imputation (also known as conditional mean imputation) replaces missing values with predicted scores from a regression equation. Like arithmetic mean imputation, regression imputation has a long history that dates back many years (Buck, 1960). The basic idea behind this approach is intuitively appealing: use information from the complete variables to fill in the incomplete variables. The first step of this imputation process is to estimate a set of regression equations. A complete-case analysis usually generates these estimates. The second step is to generate 'predicted' values for the incomplete variables. These predicted scores fill in the missing values and produce a complete data set. This approach has been studied at length as well [1970's onwards]. [Different] biases abound. For a time it was thought that 'corrections' could be made to these techniques to repair the bias issues. With the advent of much more secure [Multiple Imputation] methods in recent times, regression imputation is not recommended.

Stochastic Regression Imputation
Stochastic regression imputation also uses regression equations to predict the incomplete variables from the complete variables, but there is an extra step of simulating a value using the regression equation. This step can take the form of adding an error term [in the case of linear regression] or creating a [0/1] outcome using a threshold [as in logistic regression]. This method addresses some of the problems discussed above but the estimation of standard errors remains problematic.

Hot Deck Imputation

Hot-deck imputation is a collection of techniques that impute the missing values with scores from "similar" respondents. Statisticians at the US Census Bureau originally developed the hot-deck to deal with missing data in public-use data sets, and the procedure has a long history in survey applications (Scheuren, 2005). This procedure has received a good deal of attention in the survey literature. Methodologists have proposed several variations of hot-deck imputation. The basic premise is to impute missing values with the scores of other respondents. In its simplest incarnation, a random draw from the observed data replaces each missing value. The more typical application of hot-deck imputation replaces each missing value with a random draw from a subsample of respondents that scored similarly on a set of matching variables. For example, consider a general population survey in which some respondents refuse to disclose their income. The hot-deck procedure classifies respondents into cells based on demographic characteristics such as gender, age, race, and marital status. It then replaces the missing values with a random draw from the income distribution of respondents that shared the same constellation of demographic characteristics as the individual with missing data. Note that the background variables need not be categorical, and some hot-deck algorithms match individuals on continuous variables (e.g., nearest neighbour hot deck). Hot-deck imputation generally preserves the univariate distributions of the data and does not attenuate the variability of the filled-in data to the same extent as other imputation methods. Like other single imputation procedures, hot deck underestimates standard errors, although researchers have proposed corrective procedures (e.g., the jackknife) for estimating standard errors.

Similar Response Pattern Imputation

Similar response pattern imputation is a technique to impute each missing value with the score from another individual (i.e., a "donor" case) who has a similar score profile on a set of matching variables. The similar response pattern approach closely resembles a variant of hot-deck imputation known as nearest neighbour hot deck. Similar response pattern imputation does not necessarily produce a complete data set, and there are a number of nuances to implementing this approach. For example, identifying a set of matching variables with complete data may be a significant obstacle in and of itself. Although it is possible for incomplete variables to serve as matching variables, imputation will fail if there are no donor cases with complete data on the matching variables. Consequently, using incomplete variables as matching variables can reduce the pool of donors to the point where imputation becomes impossible, at least for a subset of cases. Similar response pattern imputation has no known theoretical foundation, so it is difficult to predict the procedure's performance.

Averaging The Available Items

Researchers in many disciplines use multiple-item questionnaires to measure complex con- structs. For example, psychologists routinely use several questionnaire items to measure depression, each of which taps into a different depressive symptom (e.g., sadness, lack of energy,sleep difficulties, feelings of hopelessness). Rather than analyzing the individual item responses, researchers typically compute a scale score by summing or averaging the items that measure a common theme. The resulting scale score reflects each respondent's overall standing on the construct of interest (e.g., a higher numeric value indicates more depressive symptoms). It is often the case that respondents answer some, but not all, of the items on a questionnaire. Rather than discard the incomplete questionnaires, researchers frequently compute scale scores by averaging the available items. For example, if a respondent answered 8 out of 10 items on a depression questionnaire, his scale score would be the average of those 8 items. The missing data literature sometimes describes this procedure as person mean imputation, but researchers in other disciplines sometimes refer to it as a prorated scale score. It may not be immediately obvious, but averaging the available items is equivalent to imputing the missing values with the mean of a respondent's complete items—thus the name "person mean imputation." Very few

empirical studies have the properties of examined person mean imputation. Averaging the available items is probably the most common approach for dealing with item-level missing data on questionnaires. Test manuals often give instructions for computing prorated scale scores with missing data, yet they may not offer cautionary statements about the biases that can result from this procedure.

Last Observation Carried Forward

Last observation carried forward is a missing data technique that is specific to longitudinal designs. As its name implies, the procedure imputes missing repeated measures variables with the observation that immediately precedes dropout. For example, if a participant drops out after the fifth week of an 8-week study, his week five score fills in the remaining waves of data. This strategy has been applied to the cases that permanently drop out as well as to the cases with intermittent missing data. Researchers still routinely use this method in medical studies and clinical trials. This technique effectively assumes that scores do not change after the last observed measurement or during the intermittent period where scores are missing. Research has shown that this imputation scheme can attenuate or exaggerate group differences at the end of a study. The direction and the magnitude of the bias are difficult to predict and depend on specific characteristics of the data and last observation carried forward will produce distorted parameter estimates. Despite its frequent use in medical studies and clinical trials, a growing number of empirical studies suggest that this approach is poor strategy for dealing with longitudinal missing data (Cook et al., 2004; Liu & Gould, 2002; Mallinckrodt et al., 2001; Molenberghs et al., 2004; Shao & Zhong, 2004). These same studies generally recommend the newer methods like multiple imputation.

Researchers have been studying missing data for decades and have proposed dozens of techniques to address the problem. Many of these approaches have enjoyed widespread use, while others are now little more than a historical footnote. The above methods are a few of the more common “traditional” missing data handling methods that you are likely to encounter in published research articles. They are all obsolete.

Multiple Imputation Using Chained Equations

Multiple Imputation Using Chained Equations is now regarded as THE approach for analyzing incomplete data.

This procedure:
1) replaces missing values with multiple sets of simulated values to complete the data,
2) applies standard analyses to each completed dataset,
and 3) adjusts the obtained parameter estimates for missing-data uncertainty

The objective is not to predict missing values as closely as possible to the true ones but to handle missing data in a way resulting in defensible statistical inferences.

This procedure has been available with Stata since Release 12.

The process is far from elementary and involves several steps each involving a number of decisions that impact on the form of the analysis.

Within Stata, one can see what is involved by:
help mi

This procedure is receiving many current [as of 2016] statistical reviews and many of these reviews are fraught with challenges, debate and, yes, controversy.

Perhaps the most challenging matter relates to the proportion of missing data relative to the actual data. There are many proposals for such percentages but none that seem definitive or completely detailed.