PharmaSUG 2017 - Paper SP01

Multiple Imputation: A Statistical Programming Story

Chris Smith, Cytel Inc., Cambridge, MA
Scott Kosten, DataCeutics Inc., Boyertown, PA

ABSTRACT

Multiple imputation (MI) is a technique for handling missing data. MI is becoming an increasingly popular method for sensitivity analyses in order to assess the impact of missing data. The statistical theory behind MI is a very intense and evolving field of research for statisticians. It is important, as statistical programmers, to understand the technique in order to collaborate with statisticians on the recommended MI method. In SAS/STAT® software, MI is done using the MI and MIANALYZE procedures in conjunction with other standard analysis procedures (e.g. FREQ, GENMOD or MIXED procedures). We will describe the 3-step process in order to perform MI analyses. Our goal is to remove some of the mystery behind these procedures and to address typical misunderstandings of the MI process. We will also illustrate how multiply imputed data can be represented using the ADaM standards and principals through an example-driven discussion. Lastly, we will do a run-time simulation in order to determine how the number of imputations influences the MI process. SAS® 9.4 M2 and SAS/STAT 13.2 software were used in the examples presented, but we will call out any version dependencies throughout the text. This paper is written to all levels of SAS users. While we present a statistical programmer's perspective, an introductory level understanding about statistics including p-values, hypothesis testing, confidence intervals, mixed models, and regression is beneficial.

INTRODUCTION

Most procedures in the SAS/STAT software only analyze complete cases (i.e. records that have all non-missing values for covariates and dependent variables, end up contributing to the statistical model). Over the course of our careers, we have seen many types of imputation done on missing data in order to perform sensitivity analyses. Examples for continuous data are baseline observation carried forward, last observation carried forward, and worst observation carried forward. For dichotomous endpoints, such as success/failure, some imputation possibilities are missing values treated as failure and missing values treated as success. These approaches are somewhat straightforward to implement from a programming perspective. An alternative to these single value imputation methods is multiple imputation (MI), but it is not as widely known amongst the statistical programming community nor is it as easy to implement as the other imputation techniques. This paper will provide an applied approach and bring clarity to the MI process. We will also show how data for a MI analysis can be represented using ADaM standards. Lastly, we will do a run-time comparison utilizing various numbers of imputations.

THE MI PROCESS

Implementing the MI process requires three steps, and unfortunately there is not a single SAS software procedure to execute the entire process. We will describe the 3-step process and the procedures used at each stage. Statistical considerations and missing data assumptions are beyond the scope of this paper. For more information on those topics, see Berglund and Heeringa (2014, Chapters 1 and 2).

STEP 1: IMPUTATION STEP

First, each missing value is imputed based on statistical modeling, and this process is repeated several times. Later, we will discuss the various methods, but for now we just need to be aware it is accomplished using the MI procedure. The output of interest from PROC MI is a data set containing multiple repetitions of the original data set, along with the newly imputed values. The repetitions are indexed with a variable named _IMPUTATION_. Let us show what this looks like. Table 1 contains subject data in a diabetes clinical trial with variables of weight and HbA1c at baseline, week 24, and week 48.

SUBJID	WEIGHT	BASE	HBA1C24	HBA1C48
100-101	60.7	6.5	6.3	6.4
100-102	57.7	7.9	7.5	**?**
100-103	80.7	7.0	**?**	8.4

Table 1. Incomplete Data

After running PROC MI, we see multiple sets of information as shown in Table 2.

SUBJID	WEIGHT	BASE	HBA1C24	HBA1C48	_IMPUTATION_
100-101	60.7	6.5	6.3	6.4	1
100-102	57.7	7.9	7.5	**7.3**	1
100-103	80.7	7.0	**8.5**	8.4	1
100-101	60.7	6.5	6.3	6.4	2
100-102	57.7	7.9	7.5	**8.0**	2
100-103	80.7	7.0	**7.1**	8.4	2
…	…	...	…	…	3

Table 2. Completed Sets

Note that subject 100-101 did not have any missing information, so the variable values are the same across each _IMPUTATION_. However, for the other two subjects, the missing values are imputed with various values. A graphical look at the HbA1c values over time for 100-103 is shown in Figure 1.

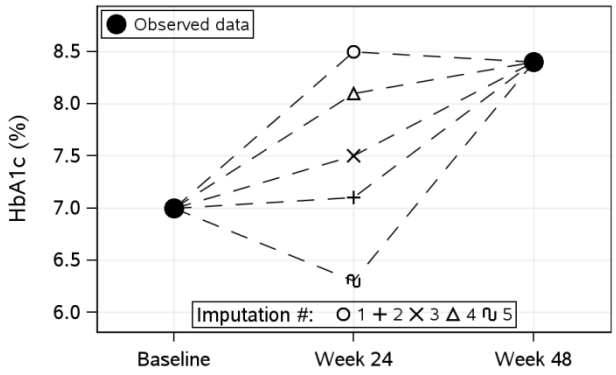

Figure 1. Imputed Values for Subject 100-103 at Week 24

By plotting the newly imputed values at week 24, we see the process introduces more variability. We will handle the added variability in step 3 of the MI process, but first we need to analyze each MI repetition. Also note that in general, the more variables included in the imputation model, the better our model will impute the data. We are not limited to the dependent variables and covariates used in the analysis step. Based on the recommendations of Berglund and Heeringa, we should use more variables than are in the analysis model (Berglund and Heeringa, 2014, pp. 16-17).

STEP 2: ANALYSIS STEP

Next, analysis is done using any SAS statistical procedure the same way we analyze non-imputed data. This includes, for example, the FREQ, MEANS, MIXED, and GENMOD procedures. However, we need to analyze each MI repetition separately. This is done by adding a BY statement with the _IMPUTATION_ variable to the relevant procedure. We will provide an example later in the paper.

STEP 3: POOLING STEP

Lastly, we need to combine all the results obtained in step 2. The MIANALYZE procedure combines the results from every MI repetition and provides valid statistical inferences (SAS, 2014, p. 5160).

Regardless of the method used to analyze the data in step 2, PROC MIANALYZE combines the information to obtain one result. Thus, we account for the variability originally introduced in step 1.

COMMON MISUNDERSTANDINGS

We have seen a few misunderstandings about the MI process. Some frequent questions are:

- Do we simply analyze the last repetition of imputed results? Do we average them before analysis?
- I performed an MI analysis using PROC MI. Do I need to use PROC MIANALYZE?
- How many imputations do we need?

Hopefully, it is clear that all MI repetitions are used in the 3-step MI process and also that PROC MIANALYZE is required to complete the final pooling step. So, we might question a programmer's comfort level with the overall MI process if they used PROC MI without PROC MIANALYZE. The 3-step MI process could be split up, with the imputation step being created within an analysis data set request and the analysis and pooling steps handled in an output table request. As for the number of imputations, we will address this later.

MISSING DATA PATTERNS

Before we get into the various MI methods, we will discuss the different types of missing data patterns. The pattern can be checked using the following code:

```
PROC MI DATA=adeff NIMPUTE=0;
   VAR weight base hba1c24 hba1c48;
RUN;
```

- NIMPUTE= requests the number of imputations. Here, we choose 0 since we are only interested in the missing data pattern and not generating any imputed values for missing data.

The PROC MI code above will reveal one of two patterns, monotone or arbitrary, shown in Output 1 and Output 2, respectively.

Group	WEIGHT	BASE	HBA1C24	HBA1C48
1	X	X	X	X
2	X	X	X	.
3	X	X	.	.
4	X	.	.	.

Output 1. Abbreviated PROC MI Output Revealing a Monotone Missing Data Pattern

Above, an "X" represents observed data and a "." represents missing data. The ordering of variables is important in defining a monotone missing data pattern. Once a missing value is encountered, then all subsequent variables in the ordered list must also be missing. In the example above, had we listed variable HBA1C24 prior to variable WEIGHT, we would have an arbitrary missing data pattern rather than a monotone missing data pattern. Typically, demographic and baseline variables are listed first, which are usually not missing, followed by the chronologically ordered, dependent variables. This type of pattern might arise in clinical trial data if subjects are lost-to-follow-up or discontinue from the study. While it may be rare for clinical data to fall exactly into a monotonic pattern, it does allow for more choices in how to impute the missing values. From a programming perspective, it still involves the 3-step process. If the data does not have a monotone pattern, then it is classified as an arbitrary missing data pattern, which is more common due to a subject missing a single visit without discontinuing the study.

Group	WEIGHT	BASE	HBA1C24	HBA1C48
1	X	X	X	X
2	X	X	X	.
3	X	X	.	X
4	.	X	X	.

Output 2. Abbreviated PROC MI Output Revealing an Arbitrary Missing Data Pattern

Subjects in Group 1 are complete cases with no missing data. Groups 2 and 3 contain subjects with missing HbA1c at one study visit, while Group 4 subjects have missing weight and HbA1c at week 48.

METHOD RECOMMENDATIONS

Choosing the recommended MI method requires knowledge of the missing data pattern. It is also dependent on the type of data we are imputing. We will not go behind the scenes and dive into statistical theory and the inner workings of these methods. However, as statistical programmers, it is important to be aware of the different MI methods in order to contribute to the conversation with our statistician colleagues. Table 3 shows various MI methods available in SAS/STAT 13.2 software.

Missing Data Pattern	Imputed Variable Type	Method	PROC MI Statement
Monotone	Continuous	Linear regression	MONOTONE REG
		Predictive mean matching	MONOTONE REGPMM
		Propensity score	MONOTONE PROPENSITY
	Binary/ordinal	Logistic regression	MONOTONE LOGISTIC
	Nominal	Discriminant function	MONOTONE DISCRIM
Arbitrary	Continuous	With continuous covariates:	
		MCMC monotone method	MCMC IMPUTE=MONOTONE
		MCMC full-data imputation	MCMC IMPUTE=FULL
		With mixed covariates:	
		FCS regression	FCS REG
		FCS predictive mean matching	FCS REGPMM
	Binary/ordinal	FCS logistic regression	FCS LOGISTIC
	Nominal	FCS discriminant function	FCS DISCRIM

Table 3. SAS PROC MI Imputation Methods. Adapted from Multiple Imputation of Missing Data Using SAS® (p. 18), by P. Berglund and S. Heeringa, 2014, Cary, NC: SAS Press. Copyright 2014, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. Reproduced with permission of SAS Institute Inc., Cary, NC.

As we can see, there are a number of methods available, and the table above nicely organizes them based on missing data patterns and variable types to be imputed. The Fully Conditional Specification (FCS) method is a newer approach that was experimental in SAS 9.3 software and available starting in SAS/STAT 12.1 software. The default method, if none is specified, is the Markov chain Monte Carlo (MCMC) method with full-data imputation (SAS, 2014, pp. 5038-5039, 5051).

A TALE OF TWO IMPUTATION METHODS

As mentioned above, prior to SAS/STAT 12.1 software the FCS method was not available. In order to impute missing values of the continuous type with an arbitrary missing data pattern and an imputation model that contains mixed covariates, we would need to break the imputation step into two parts. The first part imputes just enough missing values to create a monotone missing data pattern. We accomplish this using the MCMC method with IMPUTE=MONOTONE option and only continuous covariates in the imputation model. A rudimentary way to control for classification variables is to add them to a BY statement within PROC MI. This would allow different imputation models for each treatment group. The second part utilizes the MONOTONE statement, in a second call to PROC MI, to impute the remaining missing data.

Let us return to our previous example for subjects in a diabetes clinical trial with variables of weight and HbA1c at baseline, week 24, and week 48, but now we also want to account for treatment group and sex in our imputation and analysis models. Output 3 shows us the missing data pattern for the diabetes trial, which does not fit the monotone pattern. So, we cannot use the MONOTONE statement to impute the missing values.

Group	TRTP	SEX	WEIGHT	BASE	HBA1C24	HBA1C48
1	X	X	X	X	X	X
2	X	X	X	X	X	.
3	X	X	X	X	.	X
4	X	x	X	.	X	.

Output 3. Missing Data Pattern with TRTP and SEX Included.

Suppose we are asked to create five MI repetitions. As can be seen in the pattern of missing values, we need to impute missing values for HbA1c at baseline, week 24, and week 48. Since TRTP and SEX are classification variables, they cannot be included in the initial MCMC step directly. However, we can add them into the BY statement of the PROC MI code so that separate imputation models are performed for each combination of treatment group and sex. The first part is to impute just enough values to convert the missing data pattern to monotone:

```
PROC MI DATA=adeff NIMPUTE=5 OUT=mi_monotone;
   BY trtp sex;
   MCMC IMPUTE=MONOTONE;
   VAR weight base hba1c24 hba1c48;
RUN;
```

- MCMC statement with IMPUTE=MONOTONE option imputes just enough data to obtain a monotone missing data pattern.

We now have a data set named MI_MONOTONE with five MI repetitions of the data that is indexed with the _IMPUTATION_ variable. We now perform a separate PROC MI step using the MONOTONE statement to impute the remaining missing values. For this example, suppose the statistician wants to do a regression model that uses all "previous" variables to impute the next (e.g., BASE is imputed using SEX, TRTP, and WEIGHT; HBA1C24 is imputed using SEX, TRTP, WEIGHT, and BASE; HBA1C48 is imputed using SEX, TRTP, WEIGHT, BASE, and HBA1C24). This can be done with the following code:

```
PROC MI DATA=mi_monotone NIMPUTE=1 OUT=mi_complete;
   BY _imputation_;
   CLASS sex trtp;
   VAR sex trtp weight base hba1c24 hba1c48;
   MONOTONE REG(base = sex trtp weight);
   MONOTONE REG(hba1c24 = sex trtp weight base)
   MONOTONE REG(hba1c48 = sex trtp weight base hba1c24);
RUN;
```

- NIMPUTE= requests the number of imputations. Here, NIMPUTE=1 since we already have five repetitions of the data in MI_MONOTONE; we only need to impute the remaining missing values.
- MONOTONE statement dictates how the variables with a monotone missing data pattern are imputed. Here, we are using a regression method to impute the missing values [REG (<imputed variable> = <effects>)] for continuous variables BASE, HBA1C24 and HBA1C48.

Note that since we only have missing values for BASE, HBA1C24, and HBA1C48, we only need to specify imputation models for these three variables. The other continuous variable weight and two classification variables are always present on all observations, so it is not necessary to specify an imputation model for them. In this scenario, the above three MONOTONE statements could be replaced with a single MONOTONE statement with no options specified (i.e., "MONOTONE;"). The SAS/STAT 13.2 User's Guide for the MONOTONE statement reveals:

> *When a MONOTONE statement is used without specifying any methods, the regression method is used for all imputed continuous variables and the discriminant function method is used for all imputed classification variables. In this case, for each imputed continuous variable,* ***all preceding variables*** *in the VAR statement are used as the covariates, and for each imputed classification variable, all preceding continuous variables in the VAR statement are used as the covariates. (SAS, 2014, p. 5059)*
>
> *Excerpted from "SAS/STAT® 13.2 User's Guide" published by SAS Institute Inc. Copyright © 2014 SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.*

This is a nice default option to have, but if a more complex model is needed (e.g. inclusion of interaction terms in the imputation model [e.g. sex*trtp]) or more transparency in the programming is desired, then the three explicit MONOTONE statements may be more appropriate. We would then go on to complete the second and third steps in the overall MI process. These are the analysis and pooling steps.

This two part process was a necessity prior to the inclusion of the FCS method in SAS/STAT 12.1 software. However, this two part imputation process can be replaced using the FCS statement using the following code:

```
PROC MI DATA=adeff NIMPUTE=5 OUT=mi_complete;
   CLASS sex trtp;
   VAR sex trtp weight base hba1c24 hba1c48;
   FCS REG(base = sex trtp weight);
   FCS REG(hba1c24 = sex trtp weight base)
   FCS REG(hba1c48 = sex trtp weight base hba1c24);
RUN;
```

- FCS statement dictates how the variables with an arbitrary missing data pattern are imputed. Here, we are using a regression method to impute the missing values for continuous variables BASE, HBA1C24 and HBA1C48.

The syntax is similar to the MONOTONE statement. However, the FCS statement with no options specified (i.e., "FCS;") will not use the same imputation model as the MONOTONE statement with no options specified. The SAS/STAT 13.2 User's Guide for the FCS statement states:

> *When an FCS statement is used without specifying any methods, the regression method is used for all imputed continuous variables and the discriminant function method is used for all imputed classification variables. In this case, for each imputed continuous variable,* ***all other variables*** *in the VAR statement are used as the covariates, and for each imputed classification variable, all other continuous variables in the VAR statement are used as the covariates. (SAS, 2014, p. 5045)*
>
> *Excerpted from "SAS/STAT® 13.2 User's Guide" published by SAS Institute Inc. Copyright © 2014 SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.*
>
> *Published by SAS Institute Inc. Copyright © 2014 SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.*

So, for example, if we were to use the FCS with no options specified, HBA1C24 and HBA1C 48 would be used as covariates to impute BASE. This is not possible with the MONOTONE statement.

MCMC AND MONOTONE VERSUS FCS

There are likely to be small differences in parameter estimates and standard errors between the two part process and the one-step FCS due to the random nature of how the data are drawn for the imputations. As a result, it is possible for inferences to be affected (e.g., a p-value just below 0.05 using one method and a p-value just over 0.05 using the other method). However, this is also true for different runs of the same method using a different value in the SEED= option. In general, inferences should be similar between the two methods.

Another advantage of the FCS method is that any variable can be used in the imputation model for any other variable. We are not limited to the ordered list in the VAR statement as we are with the MONOTONE statement. Intuitively, it may seem strange to predict week 24 data based on week 48 data. However, we are actually trying to impute missing values at week 24 using observed week 48 data. In the end, the main goal is to impute missing values as accurately as possible, and being able to include as many variables in the model can only help to achieve this goal.

NUMBER OF IMPUTATIONS

Statistically speaking, the more imputations we perform, the better our estimates will be (Berglund and Heeringa, 2014, p. 40). Practically speaking, we need to determine a reasonable cutoff so computing time will not be excessive, but still produces reliable results. In theory, we want a relative efficiency (RE) close to 1.0, which is calculated as

$$RE = \left(1 + \frac{\lambda}{m}\right)^{-1}$$

where m is the number of imputations and λ is the fraction of missing information (Berglund and Heeringa, 2014, p. 39). PROC MI reports both pieces of information, which we will illustrate in our working example below.

Programmers will have to work in conjunction with statisticians to determine the appropriate number of imputations. Using m = 100 would probably be excessive in most situations, but as long as the imputation and analysis steps do not require significant computing time, this could be a conservative number to use as a starting point. Later, we will do a comparison of run times and number of imputations.

EXAMPLE WITH ADAM CONSIDERATIONS

When implementing the MI process, there are a couple different paths one could journey down. One option involves handling the 3-step MI process all within a single output program. As you can imagine, it would be quite a large, complex program and would not allow for reusability of the imputed results for various analyses. If a mismatch exists between production and validation results, it may be difficult to locate the source of the discrepancy. The discrepancy could be within the imputation, analysis, or pooling step.

We recommend splitting the process into two programs (one for the imputation step, and another for the analysis and pooling steps); this will allow for ease of debugging and documentation. A data set specification can handle the description of the imputation step, while an output shell with programming notes can explain the analysis and pooling step. Next, we will go through an example of creating an ADaM Basic Data Structure (BDS) data set, incorporating MI results, and performing the subsequent analysis.

IMPUTATION STEP

PROC MI requires a horizontal, one record per subject data set. More often than not, the data we impute will come from a vertical ADaM BDS data set. Therefore, we will need to first transpose to a horizontal structure in order to run PROC MI. It is possible, for some types of analyses, to leave it in this horizontal structure. However, for other types of analyses, a vertical data structure is required. For example, one record per subject per visit data to implement a repeated measures analysis using PROC MIXED or PROC GENMOD. Thus, we will transform back to a vertical BDS structure and utilize the DTYPE ADaM variable to indicate which values were imputed.

In our hypothetical study, we collected midichlorian count at several visits for each subject. See the Appendix for the full set of code and simulated data. We will use treatment group, gender, age and midichlorian count at each visit to impute the missing values of midichlorians at a given visit. The data structure is shown in Output 4.

SUBJID	TRTP	PARAMCD	PARAM	AVISITN	AVISIT	AVAL	BASE
101-011	Placebo	MIDI	Midichlorians (n)	1	BL	10108	10108
101-011	Placebo	MIDI	Midichlorians (n)	7	DAY 7	10067	10108
101-011	Placebo	MIDI	Midichlorians (n)	14	EOT	9949	10108
101-011	Placebo	MIDI	Midichlorians (n)	28	FU D28	9991	10108
101-011	Placebo	MIDI	Midichlorians (n)	42	FU D42	9850	10108
101-011	Placebo	MIDI	Midichlorians (n)	98	FU D98	9863	10108

Output 4. PROC PRINT Data Sample from ADaM BDS (AGE and SEX Not Shown).

We will need to transpose this information into a horizontal structure, where each visit record becomes its own variable. Note that in doing so, we will lose any visit information such as dates, times, and visit names that may be present in a typical ADaM BDS data set. The following code obtains a one record per subject data set:

```
PROC TRANSPOSE DATA=adeff OUT=onepersub PREFIX=MIDI;
    BY subjid age sex trtp base paramcd param;
    ID avisitn;
    VAR aval;
RUN;
```

- PREFIX= value is used in transposed variable names.
- ID variable values are used as the suffix of the transposed variable names (e.g. MIDI98).
- VAR statement lists the variable to be transposed.

Output 5 shows a sample subject with missing data. Note that typically missing data are not represented in ADaM BDS. To clarify, there would simply be no record for a particular visit with missing data. However, now that we have a horizontal structure, we see missing data represented in MIDI7 and MIDI98 for subject 101-002.

SUBJID	TRTP	BASE	MIDI1	MIDI7	MIDI14	MIDI28	MIDI42	MIDI98
101-002	Active	10075	10075	.	10188	10036	10044	.

Output 5. PROC PRINT Data Sample from Subject with Missing Data (AGE and SEX Not Shown).

Next, we will explore the missing data pattern by using PROC MI with 0 imputations:

```
PROC MI DATA=onepersub NIMPUTE=0;
   CLASS sex trtp;
   FCS;
   VAR sex trtp age base midi7 midi14 midi28 midi42 midi98;
RUN;
```

Note that we had to specify FCS (or MONOTONE) because the default MCMC method does not allow for classification variables. See the results from the PROC MI step above in Output 6 below.

Group	SEX	TRTP	AGE	BASE	MIDI7	MIDI14	MIDI28	MIDI42	MIDI98	Freq	Percent
1	X	X	X	X	X	X	X	X	X	131	65.50
2	X	X	X	X	X	X	X	X	.	9	4.50
3	X	X	X	X	X	X	X	.	X	12	6.00
…											
12	X	X	X	X	.	X	X	.	X	1	0.50

Output 6. Missing Data Pattern from PROC MI with NIMPUTE=0.

Note that 66% of the records are complete cases. Groups 2 through 12 have some missing data, and we can see from Group 3 that the pattern is arbitrary. Since we are imputing continuous variables with mixed covariates and an arbitrary missing data pattern, we will choose the FCS REG method from Table 3:

```
PROC MI DATA=onepersub OUT=midi_mi SEED=1999 NIMPUTE=5
   ROUND = . . . . 1 1 1 1 1;
   CLASS sex trtp;
   FCS REG (midi7 midi14 midi28 midi42 midi98);
   VAR sex trtp age base midi7 midi14 midi28 midi42 midi98;
RUN;
```

- SEED= option is used to reproduce results.
- CLASS statement specifies variables that are classification variables.
- ROUND= option tells PROC MI to round imputed results for MIDI7, MIDI14, …, MIDI98. Note that the order of values in the ROUND= option is based on the order of variables listed in the VAR statement.

- FCS REG (<vars>) lists the variables to be imputed by the FCS regression method.
- VAR statement lists the covariates used as a basis for imputing missing values.

See selected results from the PROC MI step with NIMPUTE=5 in Output 7 below.

	-----Variance---------				Relative Increase	Fraction Missing	Relative
Variable	Between	Within	Total	DF	in Variance	Information	Efficiency
MIDI7	1.236	38.935	40.42	178.39	0.038107	0.037357	0.992584
MIDI14	6.225	61.775	69.25	116.3	0.120929	0.113029	0.977894
MIDI28	8.711	49.013	59.47	72.03	0.213278	0.188233	0.963719
MIDI42	3.999	45.886	50.68	127.44	0.104574	0.098704	0.980641
MIDI98	0.910	72.602	73.69	192.06	0.015041	0.014926	0.997024

Output 7. Variance Information from PROC MI with NIMPUTE=5.

With only five imputations per missing value, we obtain an RE of 0.96 or higher for each variable. In the past, we have seen analysis plans requiring an RE of 0.98 or higher. It will be important to consult with the study statistician on the RE requirement, if it is not specified in the analysis plan. For now, we will move forward with our example and proceed to the analysis step. Unfortunately, we first need to transpose our imputed data back into a vertical structure. The _IMPUTATION_ variable from PROC MI will be included in the BY statement of our transpose:

```
PROC TRANSPOSE DATA=midi_mi OUT=backtobds;
   BY _imputation_ subjid age sex trtp base paramcd param;
   VAR midi1 midi7 midi14 midi28 midi42 midi98;
RUN;
```

While we technically can proceed with the analysis step, we choose to do some additional clean up and derive DTYPE in order to have a valid ADaM data set. Recall DTYPE is a BDS variable used to indicate the derivation method of the analysis value (i.e. AVAL or AVALC). DTYPE must be populated when AVAL or AVALC has been derived differently than other records for the same parameter (CDISC, 2016, p. 46).

First, we will need to re-derive AVISIT and AVISITN based on the former variable names (i.e. MIDI1, MIDI7, … MIDI98). Then, we need to merge back to the original (non-imputed) data set to determine which visit values were imputed. We will flag these as DTYPE=MI for values imputed by PROC MI. Lastly, we will rename _IMPUTATION_ to have a valid name not exceeding eight characters for purposes of ADaM:

```
DATA _adeffmi(DROP=_name_);
   SET backtobds;
   avisitn = INPUT(COMPRESS(_name_, 'MID'), BEST.);
   avisit = PUT(avisitn, VIS.);
RUN;

PROC SORT DATA=_adeffmi; BY subjid avisitn aval; RUN;

DATA adeffmi(RENAME=(_imputation_=imputeno));
   MERGE _adeffmi(IN=mi)
         adeff(IN=observed KEEP=subjid avisitn aval /*<ady adtm ...>*/);
   BY subjid avisitn aval;
   IF mi AND NOT observed THEN dtype='MI'; *Flag imputed records;
RUN;
```

At this stage, ADEFFMI could be output as a permanent data set, and the data set program would be complete. The output production program could then be created using ADEFFMI.

ANALYSIS STEP

Our analysis is a mixed effect model with repeated measures to assess the differences in midichlorian count between treatments at each visit. It will be performed on each imputation independently. Therefore, we must first sort by the imputation number:

```
PROC SORT DATA=adeffmi; BY imputeno subjid avisitn; RUN;

ODS OUTPUT DIFFS=diffbyvis(WHERE=(avisitn=_avisitn));
PROC MIXED DATA=adeffmi(WHERE=(avisitn > 1));
   BY imputeno;
   CLASS subjid trtp avisitn;
   MODEL aval = base trtp avisitn trtp*avisitn;
   REPEATED avisitn / SUBJECT=subjid TYPE=UN;
   LSMEANS trtp*avisitn / DIFF CL;
RUN;
```

- ODS OUTPUT DIFFS= retrieves the estimate differences from the LSMEANS statement. This will contain a comparison for each treatment combination within the same analysis visit.
- Note that since baseline midichlorian count is in the model, we only select post baseline records to enter the model (i.e. AVISITN > 1).
- BY IMPUTENO statement provides independent analyses for each imputation. This is required for the MI process. These results will later be pooled to obtain overall estimates for each visit.
- Note that the MODEL statement has fewer variables in the analysis model than there were in the imputation model (i.e. sex and age are not present). This was intentional since the MI process encourages additional information in the imputation step.

POOLING STEP

Lastly, we will need to sort (once again) in order to produce pooled results by visit using PROC MIANALYZE:

```
PROC SORT DATA=formianalyze; BY avisitn imputeno; RUN;

ODS OUTPUT PARAMETERESTIMATES=diffestvisit;
PROC MIANALYZE DATA=formianalyze;
    BY avisitn;
    MODELEFFECTS estimate;
    STDERR stderr;
RUN;
```

In Output 8 below, we end up with pooled results (one per visit) taking into account the variability between imputations.

AVISITN	Estimate	StdErr	LCLMean	UCLMean	DF	tValue	Probt
7	4.071768	13.555938	-22.6597	30.8032	199.31	0.30	0.7642
14	31.181263	16.098630	-0.3949	62.7575	1629	1.94	0.0529
28	6.595001	14.223452	-21.2844	34.4744	17375	0.46	0.6429
42	30.899041	13.727096	3.9851	57.8130	3497.9	2.25	0.0245
98	101.630556	15.869421	70.5131	132.7480	2693.9	6.40	<.0001

Output 8. PROC PRINT Output Data Set from PROC MIANALYZE.

The estimate column represents pooled results for the differences in midichlorian count between treatments at each visit. The overall standard error, 95% confidence interval, and p-values are also available. Note that for days 7, 14, and 28, the confidence intervals contain 0, and the treatment effect is not significant at the 0.05 level. However, at day 42 and 98 there is a significant difference between active and placebo groups.

Let us look closer at what is happening behind the scenes in PROC MIANALYZE. In Output 9, there are the five observations being fed into PROC MIANALYZE for AVISITN = 98.

IMPUTENO	Estimate	StdErr	WITHINVAR
1	103.53	15.8707	251.880
2	105.59	15.2563	232.755
3	98.8878	15.4334	238.190
4	99.4763	15.4012	237.198
5	100.67	15.8319	250.649

Output 9. PROC PRINT of FORMIANALYZE Where AVISITN=98, with Derived WITHINVAR.

The estimated variance of our estimate within each imputation is WITHINVAR = $STDERR^2$. Below, Output 10 shows the mean and variance descriptive statistics for the ESTIMATE and WITHINVAR variables across the five imputations.

Variable	Mean	Variance
Estimate	101.631	8.087
WITHINVAR	242.134	73.845

Output 10. PROC MEANS of ESTIMATE and WITHINVAR Where AVISITN=98

There are similarities in the information in Output 10 when compared against the PROC MIANALYZE data set from Output 8 and the variance information in Output 11 from the same PROC MIANALYZE.

	--------Variance-------				Relative Increase	Fraction Missing	
Parameter	Between	Within	Total	DF	in Variance	Information	Relative Efficiency
estimate	8.087	242.134	251.839	2693.9	0.040078	0.039247	0.992212

Output 11. Variance Information from PROC MIANALYZE Where AVISITN=98

The overall parameter estimate for AVISITN = 98, 101.631, is simply the mean of the ESTIMATE variable. The between-imputation variance, 8.087, is the variance of the ESTIMATE variable across the five imputations. The within-imputation variance, 242.134, is the mean of the variance within each of the five imputations. The total variance (TV) is a given by

$$TV = W + \left(\frac{m+1}{m}\right) \cdot B = 242.134 + \left(\frac{6}{5}\right) \cdot 8.087 = 251.839.$$

where B = between-imputation variance, W = within-imputation variance, and m = number of imputations performed. The standard error used for computing confidence intervals and t-statistics is the square root of the total variance, 15.869. One other insight is the Relative Increase in Variance (RIV) is given by

$$RIV = \frac{TV - W}{W} = \frac{251.839 - 242.134}{242.134} = 0.04008.$$

Suppose we had no missing values in our data, but performed an MI analysis anyway. Then, we would not have any between-imputation variation because all imputations would result in the same estimate. So, the total variance in this case would be equal to the within-imputation variance. One way to think of this is like a percent change from baseline calculation, where W represents our baseline (i.e., the smallest possible variance when no data are missing) and $TV - W$ represents the amount of new variation introduced by the imputation step.

NUMBER OF IMPUTATIONS EPISODE II: RUN-TIME COMPARISON

We would like to return to the number of imputations discussion. We want to do a comparative exercise looking at various numbers of MI repetitions and run-times of each step in the process. Below, Table 4 shows the run-times of 5, 25, 50, and 100 repetitions for the imputation, analysis, and pooling steps on the 200 simulated subjects.

Step\Imputations	5	25	50	100
Step 1: MI	0.14	0.55	1.07	2.06
Step 2: MIXED	0.27	0.93	2.25	4.05
Step 3: MIANALYZE	0.02	0.03	0.02	0.02

Table 4. Approximate Run-times (seconds) by Number of Imputations and Step of the MI Process

We see the run-time increases for the MI and MIXED steps as the number of imputations increase from 5 to 100, but it is only a few seconds. The pooling step done by PROC MIANALYZE is not affected by the number of MI repetitions. By increasing the number of subjects in the simulation to 2000 (a 10-fold increase), the run-times for 100 repetitions become 20.21, 23.08, and 0.03 seconds during the MI, MIXED, and MIANALYZE procedures, respectively. We see this has a significant impact on run-times, but still not too extreme. During program development, we could use fewer imputations for PROC MI, then later increase NIMPUTE=100 for the final run.

CONCLUSION

The MI process can be an intimidating and challenging task to implement. However, we have now covered the basics for performing a MI analysis. It is a 3-step process requiring MI and MIANALYZE procedures, in addition to an analysis procedure (e.g. FREQ, GENMOD or MIXED procedures). The method used in PROC MI depends on the types of variables to be imputed along with the missing data pattern. As for the number of imputations for each missing data value, we recommend 100 as a starting point and suspect it will provide adequate RE. We also revealed how the FCS method can be used to impute values for data with arbitrary missing data patterns. The FCS method is more flexible in that it allows for classification variables as covariates in the imputed model and can be used to impute various types of data (i.e. continuous, binary/ordinal, nominal). It also allows for all variables to be used in the imputation model, not just preceding variables. Therefore, we imagine most MI analyses, in order to incorporate demographic data such as race, sex, or ethnicity in the imputation model, will resort to using FCS.

REFERENCES

Berglund, Patricia and Heeringa, Steven, 2014. *Multiple Imputation of Missing Data Using SAS®*. Cary, NC: SAS Institute Inc.

CDISC. “Analysis Data Model Implementation Guide v1.1.” CDISC, 12 Feb. 2016. Web. 12 Mar. 2017.

SAS Institute Inc. 2014. *SAS/STAT® 13.2 User’s Guide*. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Name: Chris Smith
Enterprise: Cytel Inc.
E-mail: chris.smith@cytel.com
Web: www.linkedin.com/in/smithchr

Name: Scott Kosten, Ph.D.
Enterprise: DataCeutics Inc.
E-mail: kostens@dataceutics.com
Web: www.linkedin.com/in/kostens

APPENDIX

```
/*************************************************************************/
/* Appendix - Code for Simulated Data and Multiple Impuation (MI) Steps */
/*************************************************************************/

*Format for visit;
PROC FORMAT;
    VALUE vis 1 = 'BL'
              7 = 'DAY 7'
              14= 'EOT'
              28= 'FU D28'
              42= 'FU D42'
              98= 'FU D98';
RUN;

/* Core data simulation */
DATA dd1;
    LENGTH subjid trtp $7. paramcd $4. param $17.;

    *Visit structure;
    ARRAY visits (6) _TEMPORARY_ (1 7 14 28 42 98);

     *Sets seed for all rand functions ;
    CALL STREAMINIT(1977);

    *Data simulation loop;
    DO id = 1 TO 200;
        *Assign subject id;
        subjid='101-'||PUT(id, Z3.);

        *Assign gender;
        IF RAND('Bernoulli', 0.5) = 1 THEN sex='M';
        ELSE sex='F';

        *Assign treatment group;
        IF RAND('Bernoulli', 0.5) = 1 THEN trtp='Active';
        ELSE trtp='Placebo';

        *Assign age;
        age=ROUND(RAND('Normal', 50, 15));

        *Assign result name and short code;
        paramcd='MIDI';
        param='Midichlorians (n)';

        *Simulate results;
        DO i=1 TO DIM(visits);
            avisitn=visits(i);
            avisit=PUT(avisitn, VIS.);

            IF trtp='Active' THEN DO;
                *Produces advancing treatment effect over time;
                IF visits(i)=1 then aval=ROUND(RAND('Normal', 10000, 100));
                ELSE aval=ROUND(RAND('Normal', 10000, 100)+visits(i));
            END;
```

```
            ELSE DO;
                *Produces no treatment effect for placebo patients;
                aval=ROUND(RAND('Normal', 10000, 100));
            END;

            OUTPUT;
        END;
    END;

    DROP i id;
RUN;

*Derive baseline and simulate missing data;
DATA adeff;
    SET dd1;
    BY subjid;

    CALL STREAMINIT(1980);

    *Derive baseline value;
    RETAIN base .;
    IF first.subjid THEN base=.;
    IF avisit='BL' THEN base= aval;

    *Simulate arbitrary missing data pattern;
    IF avisit ne 'BL' THEN DO;
        IF RAND('Bernoulli', 0.08) THEN DELETE;
    END;
RUN;

*Transpose to horizontal structure;
PROC TRANSPOSE DATA=adeff OUT=onepersub PREFIX=MIDI;
    BY subjid age sex trtp base paramcd param;
    ID avisitn;
    VAR aval;
RUN;

*Complete case - for paper output;
PROC PRINT data=adeff (DROP=age sex) NOOBS;
    WHERE subjid='101-011';
RUN;

*Missing data case - for paper output;
PROC PRINT DATA=onepersub NOOBS;
    WHERE MISSING(midi7) and MISSING(midi98);
    VAR subjid trtp base midi1 midi7 midi14 midi28 midi42 midi98;
RUN;

/* Multiple Imputation - 3-step process */

*Step 1a - check missing data pattern (arbitrary vs monotone);
PROC MI DATA=onepersub NIMPUTE=0;
    CLASS sex trtp;
    FCS;
    VAR sex trtp age base midi7 midi14 midi28 midi42 midi98;
RUN;
```

```
*Step 1b - select appropriate method given missing
           data pattern and variables to be imputed;
PROC MI DATA=onepersub OUT=midi_mi SEED=1999 NIMPUTE=5
    ROUND = . . . . 1 1 1 1 1;
    CLASS sex trtp;
    FCS REG (midi7 midi14 midi28 midi42 midi98);
    VAR sex trtp age base midi7 midi14 midi28 midi42 midi98;
RUN;

*Step 2a - transpose back to vertical structure;
PROC TRANSPOSE DATA=midi_mi OUT=backtobds;
    BY _imputation_ subjid age sex trtp base paramcd param;
    VAR midi1 midi7 midi14 midi28 midi42 midi98;
RUN;

*Clean up;
DATA _adeffmi;
    SET backtobds;

    *Rederive avist/avisitn;
    avisitn=INPUT(COMPRESS(_name_, 'MID'), BEST.);
    avisit=PUT(avisitn, VIS.);

    DROP _name_;
RUN;

*Sort for merge with original data;
PROC SORT DATA=_adeffmi;
    BY subjid avisitn aval;
RUN;

*Flag imputed records;
DATA adeffmi (RENAME=(_imputation_=imputeno));
    MERGE _adeffmi(IN=mi)
          adeff  (IN=observed KEEP=subjid avisitn aval);
    BY subjid avisitn aval;

    IF mi AND NOT observed THEN dtype='MI';
RUN;

*Sort for analysis step;
PROC SORT DATA=adeffmi;
    BY imputeno subjid avisitn;
RUN;

*Step 2b - Analysis step;
ODS OUTPUT diffs=diffbyvis(WHERE=(avisitn=_avisitn));
PROC MIXED DATA=adeffmi (WHERE=(avisitn > 1));
    BY imputeno;
    CLASS subjid trtp avisitn ;
    MODEL aval = base trtp avisitn trtp*avisitn ;
    REPEATED avisitn / SUBJECT=subjid TYPE=un;
    LSMEANS trtp*avisitn /DIFF CL;
RUN;
ODS OUTPUT CLOSE;
```

```
*Derive variable for later exploration;
DATA formianalyze;
    SET diffbyvis;
    withinvar=stderr**2;
RUN;

*Step 3 - Pooling step;
PROC SORT DATA=formianalyze;
    BY avisitn imputeno;
RUN;

ODS OUTPUT PARAMETERESTIMATES=diffestvisit;
PROC MIANALYZE DATA=formianalyze;
    BY avisitn;
    MODELEFFECTS estimate;
    STDERR stderr;
RUN;
ODS OUTPUT CLOSE;

PROC PRINT DATA=diffestvisit NOOBS;
RUN;

*Exploratory - Within Variance;
PROC PRINT DATA=formianalyze NOOBS;
      WHERE avisitn=98;
      VAR imputeno estimate stderr withinvar;
RUN;

PROC MEANS DATA=formianalyze MEAN VAR MAXDEC=3;
      WHERE avisitn=98;
      VAR estimate withinvar;
RUN;
```