



www.quant.ku.edu

KUANT Guides

Guide No.
KUANT 011.3

Missing Data in Large Data Projects:

**Two methods of missing data imputation
when working with large data projects.**

Little, T.D., McConnell, E.K., Howard, W.J., & Stump, K.N. (2008)

Preliminaries

This handout outlines *two* methods of missing data imputation when working with *large data* projects. The first one involves imputation from *aggregated scores* and the second begins with a *principal components analysis*. This guide is recommended *only* for situations in which imputation of the full item set does not work.

In the following two examples, there are 10 scales where each scale consists of 8 - 10 items for a total of 97 items in the data set.

Method One: Imputing at the level of aggregated scales

1) Create aggregate scales of the items on the data set (use the average).

```
ScaleA = mean(of a1 a2 a3 a4 a5 a6 a7 a8 a9 a10);  
ScaleB = mean(of b1 b2 b3 b4 b5 b6 b7 b8 b9 b10);  
ScaleC = mean(of c1 c2 c3 c4 c5 c6 c7 c8 c9 c10);  
ScaleD = mean(of d1 d2 d3 d4 d5 d6 d7 d8 d9 d10);  
ScaleE = mean(of e1 e2 e3 e4 e5 e6 e7 e8 e9 e10);  
ScaleF = mean(of f1 f2 f3 f4 f5 f6 f7 f8 f9 f10);  
ScaleG = mean(of g1 g2 g3 g4 g5 g6 g7 g8 g9 g10);  
ScaleH = mean(of h1 h2 h3 h4 h5 h6 h7 h8 h9 h10);  
ScaleI = mean(of i1 i2 i3 i4 i5 i6 i7 i8);  
ScaleJ = mean(of j1 j2 j3 j4 j5 j6 j7 j8 j9);
```

Notes:

In this example, notice that the items are *averaged*. It is *not* recommended to *sum* the items even if they are *standardized* before imputing. If some of the items are missing for some individuals, the sum method creates means and standard deviations that may be on very different metrics within a scale. Furthermore, the metric of the scales may be different. Consider that a person with 2 items missing would have a lower *sum* score than a person with 0 items missing regardless of the person's true score. Utilize the *average* of the items to obtain estimates closer to the true score.

2) Impute at the level of the aggregate scales.

The SAS Proc MI syntax will look something like this:

```
Proc mi data=sample out=outmi nimpute=1 seed=761253;  
  em maxiter=1000;  
  mcmc initial=em (maxiter=1000);  
  var ScaleA ScaleB ScaleC ScaleD ScaleE ScaleF ScaleG ScaleH ScaleI ScaleJ;  
Run;
```

*At this step the **scales** now have NO missing data but the **items** still contain missing data*

3) Use the imputed scales as anchors to impute missing data in the items in a sequential process

Now the key is to put the right variables into the variable list for estimating the missing data:

```
Proc mi data=outmi out=outmi.....;
  var ScaleA ScaleB ScaleC Scaled ScaleE ScaleF ScaleG ScaleH ScaleI
      j1 j2 j3 j4 j5 j6 j7 j8 j9;
* At this step ScaleJ is not in the list but the items for ScaleJ are now imputed
  ScaleJ needs to be removed because it is a linear combination of its items;

Proc mi data=outmi out=outmi.....;
  var ScaleA ScaleB ScaleC Scaled ScaleE ScaleF ScaleG ScaleH          ScaleJ
      i1 i2 i3 i4 i5 i6 i7 i8;
* At this step ScaleI is not in the list but the items for ScaleI are now imputed;

Proc mi data=outmi out=outmi.....;
  var ScaleA ScaleB ScaleC Scaled ScaleE ScaleF ScaleG          ScaleI ScaleJ
      h1 h2 h3 h4 h5 h6 h7 h8 h9 h10;
* At this step ScaleH is not in the list but the items for ScaleH are now imputed;

Proc mi data=outmi out=outmi.....;
  var ScaleA ScaleB ScaleC Scaled ScaleE ScaleF          ScaleH ScaleI ScaleJ
      g1 g2 g3 g4 g5 g6 g7 g8 g9 g10;
* At this step ScaleG is not in the list but the items for ScaleG are now imputed;

Proc mi data=outmi out=outmi.....;
  var ScaleA ScaleB ScaleC Scaled ScaleE          ScaleG ScaleH ScaleI ScaleJ
      f1 f2 f3 f4 f5 f6 f7 f8 f9 f10;
* At this step ScaleF is not in the list but the items for ScaleF are now imputed;

Proc mi data=outmi out=outmi.....;
  var ScaleA ScaleB ScaleC          ScaleF ScaleG ScaleH ScaleI ScaleJ
      e1 e2 e3 e4 e5 e6 e7 e8 e9 e10;
* At this step ScaleE is not in the list but the items for ScaleE are now imputed;

Proc mi data=outmi out=outmi.....;
  var ScaleA ScaleB ScaleC          ScaleE ScaleF ScaleG ScaleH ScaleI ScaleJ
      d1 d2 d3 d4 d5 d6 d7 d8 d9 d10;
* At this step Scaled is not in the list but the items for Scaled are now imputed;

Proc mi data=outmi out=outmi.....;
  var ScaleA ScaleB          Scaled ScaleE ScaleF ScaleG ScaleH ScaleI ScaleJ
      c1 c2 c3 c4 c5 c6 c7 c8 c9 c10;
* At this step ScaleC is not in the list but the items for ScaleC are now imputed;

Proc mi data=outmi out=outmi.....;
  var ScaleA          ScaleC Scaled ScaleE ScaleF ScaleG ScaleH ScaleI ScaleJ
      b1 b2 b3 b4 b5 b6 b7 b8 b9 b10;
* At this step ScaleB is not in the list but the items for ScaleB are now imputed;

Proc mi data=outmi out=outmi.....;
  var          ScaleB ScaleC Scaled ScaleE ScaleF ScaleG ScaleH ScaleI ScaleJ
      a1 a2 a3 a4 a5 a6 a7 a8 a9 a10;
* At this step ScaleA is not in the list but the items for ScaleA are now imputed;
```

At this point ALL of the items are imputed.

Method Two:

1) Do a *principal components* analysis of the items on the dataset and output the meaningful components as component scores.

The SAS Proc Princomp syntax (Proc Factor may also be used) will look something like this:

```
Proc princomp data=sample out=Prin;  
  var a1-a10 b1-b10 c1-c10 d1-d10 e1-e10 f1-f10 g1-g10 h1-h10 i1-i8 j1-j9;  
Run;
```

**For Proc princomp options: <http://support.sas.com/onlinedoc/913/docMainpage.jsp>;*

In this example, let's assume that the first 12 components accounted for a meaningful amount of variance and any more components would not yield much more reliable information.

2) Estimate any missing data on the component scores

The SAS Proc MI syntax will look something like this:

```
Proc mi data=Prin out=outmi nimpute=1 seed=761253;  
  em maxiter=1000;  
  mcmc initial=em (maxiter=1000);  
  var Component1-Component12;  
Run;
```

**For Proc mi options see: <http://support.sas.com/onlinedoc/913/docMainpage.jsp>;*

*At this step the **component scores** now have NO missing data
but the **items** still contain missing data*

3) Use the estimated component scores as anchors to estimate the missing data in the items on the dataset.

```
Proc mi data=outmi out=outmi.....;  
  var Component1-Component12 a1-a10 b1-b10;
```

** At this step we plug approximately 20 items into the variable list to estimate their missing values;*

```
Proc mi data=outmi out=outmi.....;  
  var Component1-Component12 c1-c10 d1-d10;
```

** At this step we plug in approximately 20 more items into the list to estimate their missing values;*

```
Proc mi data=outmi out=outmi.....;  
  var Component1-Component12 e1-e10 f1-f10;
```

** At this step we plug in approximately 20 more items into the list to estimate their missing values;*

```
Proc mi data=outmi out=outmi.....;  
  var Component1-Component12 g1-g10 h1-h10;
```

** At this step we plug in approximately 20 more items into the list to estimate their missing values;*

```
Proc mi data=outmi out=outmi.....;  
  var Component1-Component12 i1-i8 j1-j9;
```

** At this step we plug the remaining items to estimate their missing values;*

*At this point, ALL of the **items** are imputed.*

What about Multiple Imputations:

- 1) Change the number of imputations option (*nimpute=*) and change the MCMC chain option from single to multiple (MCMC chain=multiple) in Step 2 (aggregate scale level) of the Proc MI syntax

The SAS Proc MI syntax will look something like this:

```
Proc mi data=sample out=outmi nimpute=20 seed=398127;
  em maxiter=1000;
  mcmc initial=em (maxiter = 1000);
  var ScaleA ScaleB ScaleC ScaleD ScaleE ScaleF ScaleG ScaleH ScaleI ScaleJ;
Run;
```

* At this step all scales are now imputed 20 times;

Notes:

Multiple imputations should be carried out during *Step 2* of the previous two examples. Notice that the input and output datasets in the SAS syntax are different for this imputation step; the output dataset (outmi) includes a stack of imputed datasets on top of each other 1 – 20. If the original dataset (*data=sample*) contained 200 observations then the new data set (*out=outmi*) with 20 imputations will contain 4000 observations with the 201st observation being the first observation in the 2nd imputation. SAS will create a new variable called “_Imputation_” with a value of 1 for the first imputation, 2 for the second imputation, and so on to a value of 20 for the last imputation.

- 2) Use the output dataset from the multiple imputation of scales (outmi) as the input dataset for the imputation of items (*data=outmi*). Return the number of imputations option to 1 (*nimpute=1*) and return the MCMC chain option to single (MCMC chain=single). Additionally, replace the aggregate scale values with the item values (as in Step 3 above); however, include a *BY _IMPUTATION_* statement (as noted below).

```
Proc mi data=outmi out=outmi nimpute=1 seed=398127;
  em maxiter=1000;
  mcmc initial=em (maxiter=1000);
  var ScaleA ScaleB ScaleC ScaleD ScaleE ScaleF ScaleG ScaleH ScaleI
      j1 j2 j3 j4 j5 j6 j7 j8 j9;
  BY _IMPUTATION_;
Run;
```

* At this step ScaleJ is *not* in the list but the *items* for ScaleJ are now imputed. Items are imputed one time (nimpute=1) per scale imputation (by *_imputation_*), creating 20 imputations for items in ScaleJ;

```
Proc mi data=outmi out=outmi nimpute=1.....;
  var ScaleA ScaleB ScaleC ScaleD ScaleE ScaleF ScaleG ScaleH          ScaleJ
      i1 i2 i3 i4 i5 i6 i7 i8;
  BY _IMPUTATION_;
Run;
```

* At this step ScaleI is *not* in the list but the *items* for ScaleI are now imputed. Items are imputed one time (nimpute=1) per scale imputation (by *_imputation_*), creating 20 imputations for items in ScaleI;

.
.
.

Continue this process for all scales (remember there are A - J scales).

.
. .
.

```
Proc mi data=outmi out=outmi nimpute=1.....;
  var      ScaleB ScaleC ScaleD ScaleE ScaleF ScaleG ScaleH ScaleI ScaleJ
          a1 a2 a3 a4 a5 a6 a7 a8 a9 a10;
  BY _IMPUTATION_;
```

* At this step ScaleA is *not* in the list but the *items* for ScaleA are now imputed. Items are imputed one time (nimpute=1) per scale imputation (by `_imputation_`), creating 20 imputations for items in ScaleA;

Notes:

Note that the input and output datasets in this section of the SAS syntax do *not* change with each item imputation step. The final result will be a dataset (outmi) with all items imputed 20 times.

For additional information regarding further analysis of a dataset with multiple imputations in LISREL see the Rubin's Rules Pooler for LISREL Estimates:

<http://www.quant.ku.edu/resources/tools.html>

Helpful SAS Proc MI options:

Data = SAS_data_set this option names the input SAS data set to read-in for analysis

Out = SAS_data_set this option names the output SAS data set to export after analysis

nimpute = value (default is 1) this option specifies the number of imputations

seed = value (default is based on the computer clock) this option specifies the seed to begin the random number generator and is needed to replicate results (it is best to choose a 6 – 7 digit odd number)

EM maxiter = value (default is 200) this option specifies the maximum number of iterations for the expectation-maximization (EM) algorithm

initial = EM (default is the EM estimates) this option specifies the initial parameter estimates for the MCMC method

maxiter = value (default is 200) this option specifies the maximum number of iterations for the EM algorithm

For more option details see:

<http://support.sas.com/rnd/app/papers/miv802.pdf>