EVALUATION OF PROC IMPUTE AND SCHAFER'S IMPUTATION SOFTWARE

Mingxiu Hu, Sameena Salvucci, Stanley Weng, Synectics, and Michael P. Cohen, NCES
Mingxiu Hu, Synectics for Mgmt Decision Inc., 3030 Clarendon Blvd, Arlington, VA 22201

KEY WORDS: Imputation methods, Missing data, Software evaluation

I. INTRODUCTION. The problem of missing data commonly exists in surveys conducted by the National Center for Education Statistics (NCES). These missing values not only mean less efficient estimates because of the reduced size of the data base but also that standard complete-data methods can not be immediately used to analyze the data. Moreover, possible biases exist because the respondents are often systematically different from the nonrespondents; of particular concern, these biases are difficult to eliminate since the precise reasons for non-response are usually not known.

Imputation is one of the most popular ways to reduce the impact of missing values. A number of imputation software packages have been developed recently. We implemented and evaluated two advanced imputation software products, Proc Impute (PC version 2.0, 1991) and Schafer's Multiple Imputation Software (version 2.0, 1995), under a PC DOS or Windows environment. The original version of Proc Impute, created by the American Institute for Research (AIR), is a SAS procedure run on a JCL mainframe, but the recent version is a stand-alone Fortran program run under a DOS environment. It is a regression-based algorithm and is believed to be more general and to produce more accurate results than a standard "hot deck" procedure. This software also allows a user to generate multiple imputations, but the results may not be "proper" in the sense of Rubin's definition (Rubin, 1987). On the other hand, Schafer's Multiple Imputation Software, created by Dr. Schafer of Pennsylvania State University, is designed for a multiple imputation purpose. This software consists of three independent parts. The first part uses a multivariate normal model to perform imputations for continuous variables; the second part uses a saturated multinomial model and a constrained loglinear model to perform imputations for categorical variables; and the third part uses restricted and unrestricted general location models to perform imputations for mixed variables. This paper presents the evaluation of the first part, and the evaluations of the other two parts are under way.

Sections II and III of this paper present the detailed evaluations of Proc Impute and Schafer's Multiple Imputation Software, respectively. Section IV describes a simulation study which compares the two imputation algorithms in terms of average imputation error and mean bias. Some discussions are given in section V.

II. EVALUATION OF PROC IMPUTE. In this section, we first briefly describes the statistical algorithms used in this software, and then present our evaluations of the software in terms of its usability/performance under a PC 586 environment, its suitability for generating multiple imputations, and its adaptability to different NCES surveys.

ALGORITHM. Proc Impute is a regression-based distributional estimation procedure that is believed to be more general and to produce more accurate results than a standard "hot deck" procedure. Basically, this procedure assumes that relations among variables are constant for observed cases and missing cases, and considers each variable on the file in turn as a "target" variable whose missing values are to be filled in and uses information on other variables to minimize the error in imputing each "target" variable. For each "target" variable, regression analysis is used to find the best combination of predictors. Once the regression models are constructed and regression values are computed for all cases (both missing cases and observed cases), Proc Impute partitions the range of regression values into subsets. All cases in a given subset that are missing the target variable then are imputed with weighted averages of two values drawn from that regression function value subset and an adjacent subset with probability proportional to the distribution of reported values for that variable within these two subsets. The weighted average value is rounded to an integer if the integer flag is set for the target variable. The basic assumption of this algorithm is that within these homogeneous subsets, the missing value cases will have the same target value distribution as the cases with reported values on the target variable.

After all missing values have been imputed for a case, the case is written to the output file with all of the missing values filled in. Missing data flags are also created and set for each variable with a value of "I" corresponding to imputed values, "R" for real values, and "A" for skip missing values.

EVALUATION. Our evaluation is based on runs conducted on the NCES data set "1993 National Survey of Postsecondary Faculty" (NSOPF), which contains about 12,000 cases and 400 variables. The runs were performed in a Pentium (586) environment--90 MHZ clock speed, 16 megabytes of memory, and 600 megabytes of hard disk space. We focus on the following six questions (1)~ (6) in our evaluation.

(1) How many runs would it take to impute all variables in a survey?

Proc Impute is a regression-based algorithm. In our experience, it can effectively incorporate no more than 30 variables into any one regression model. Therefore, any "large" data set must be partitioned into subsets containing no more than 30 variables each before being processed by Proc Impute.

Given those considerations, how many runs does it take to impute all variables in a survey? In short, **one** run is all that is required to impute all missing values (for all variables) in any survey. However, in the case of "large" data sets, one run would consist of the following steps:

(i) First, the analyst must partition the variables into subsets. Let's use the NSOPF data set as an example. If the strategy is to merely use "adjacent" variables in the regression models, then the analyst would partition the NSOPF data set into about 14 (i.e., $400/30 \approx 13.3$) subsets. If the strategy is to use key predictor variables (e.g., sex, race, and region) in every regression model, then the analyst would divide the NSOPF data set into 15 (i.e., $397/27 \approx 14.7$) subsets. Obviously, how much the regression models are customized determines not only the number of subsets to be processed by Proc Impute, but also the amount of time devoted to the overall imputation process (see question (5) below).

(ii) Second, an ASCII data file must be created for each subset of variables (i.e., for each regression model).

(iii) The third step involves running Proc Impute on each data subset. To perform an imputation, a control file must be constructed. Fortunately, one can specify all of the regression models in the same control file, and run Proc Impute on the entire data set as a single batch job.

(iv) The final step in the process is combining the output (imputed) files into a single file that contains both the original and the imputed values for all variables, with flags indicating imputed values.

In one sentence, *Proc Impute* will impute **all** missing values in any data (sub)set that is specified in the control file in **one** run (as a batch job), while the amount of pre- and post-processing of a given data file is dependent upon the size of the file, the number of variables in the file, the relationships among the variables, etc.

(2) Can Proc Impute handle all types of variables (i.e., continuous, ordinal, and categorical) correctly?

Proc Impute will only process "numerical" data. Any "character" variables in the data set must be either recoded to "numeric" or removed from the data set. Once the data set holds the proper coding, each variable will have a continuous, an ordinal, or a categorical (dichotomous or polytomous) distribution. Then Proc Impute can be used to process all the variables in the data set. It should be noted that Proc Impute uses a regression-based algorithm, and assumes that each variable is continuous with normal residuals and homogeneous variance. It is very rare for all the variables to satisfy these conditions. A linear regression that is run on the variables which violate the distribution or variance assumptions often yields high probabilities of generating "out-of-range" predictions. However, in attempts to avoid imputing "out-of-range" values, Proc Impute uses knowledge about conditional frequency distributions along with its regression algorithm when imputing missing values.

Therefore, for continuous, ordinal, or dichotomous categorical variables, Proc Impute requires no special pre- or post-processing, and the imputed values for all such variables are reasonable--reasonable in the sense that the imputed values both fall within the range of the observed values and mimic the distributional properties of the observed values. However, polytomous categorical variables are potentially troublesome to Proc Impute. For each polytomous variable, the analyst needs to create an appropriate number of dummy (0/1) variables, and then run Proc Impute on the dummy variables. Proc Impute handles the dummy variables in the same fashion that it handles dichotomous variables. Since Proc Impute does not understand that the dummy variables are grouped as <u>sets of variables</u>, the imputed values may be meaningless[1]; however, since the dummy variables in any set representing a given polytomous variable are highly correlated, that should rarely happen. Fewer than four percent of the imputed values of our "reconstructed" polytomous variables were bad. In such cases "hot-deck" method may be appropriate to impute for those bad cases.

(3) How much special processing is required to handle skip patterns?

It is very convenient to handle skip patterns with the most recent version of Proc Impute. An analyst only needs to set the skip missing values to ".A" for a SAS data file or "A" for an ASCII data file; then Proc Impute will know not to impute any value for these missing cases and output a skip flag "A" for each of them.

(4) How much memory and disk space are needed?

The amount of required disk space is predominantly a function of the size of your data file and the number of sets of multiple imputations. It requires about 480 Kb conventional memory to run Proc Impute. With a 586 Pentium PC (16 Mb of total memory and 636 Kb conventional memory), we did not experience memory problems after we removed some programs to generate 488 Kb free conventional memory for Proc Impute. Considering the rate of technological developments, we do not foresee future difficulties.

[1]It may be the case that more than one of the (n-1) dummy variables, which represent an n-category polytomous variable, will be imputed with the value "1"--this would indicate that the original polytomous variable case assumes multiple categories simultaneously!

(5) How fast is it?

For the actual imputation processing, speed is not a serious issue. For the NSOPF survey, a run with 12,000 cases and 30 variables (one subset) took less than 20 minutes--this translates into a processing time of less than 280 minutes if the entire data set was run as a batch job (a total of 14 subsets). However, an analyst will devote his/ her major processing time to pre- and post-imputation file management, such as changing "character" variables to "numeric" variables and/or removing them, creating ASCII data files, constructing the control file, partitioning a large data set into subsets, creating dummy variables for polytomous variables, recoding skip patterns to ".A", combining the output (imputed) subsets into one overall completed file, etc.

(6) Does Proc Impute perform "proper" imputations in the sense of Rubin? If not, can Proc Impute be adapted to perform multiple imputations?

Multiple imputation involves imputing each missing value multiple times. Hence, in performing a multiple imputation, one creates multiple files of complete data, wherein each of the multiple data files has a different set of imputed values. Once the multiple files have been constructed, the analyst should replicate all subsequent analyses by using the information from all of the multiple files to assess the impact of random variation (of missing values) on statistical inferences.

Rubin's Proper Multiple Imputation (PMI) criteria are based on the asymptotic properties of the multiple imputation statistics; hence, the concept of "proper" imputation is exclusively suitable to multiple imputation approaches. Since Proc Impute uses a single imputation procedure based upon a (non-Bayesian) distributional estimation, Proc Impute cannot meet Rubin's criteria for "proper" imputation. However, it is the case that Proc Impute is designed to assess the impact of random variation (of missing values) on statistical inferences.

Even though the two methods cannot be compared in the "proper" sense, we can still examine the criteria for the optimalities of these two methods--the randomization-valid inferences for PMI are based on the concept of the central limit theorem whereas the distributional estimation method employed in Proc Impute is based on the Pitman Closeness criterion[2].

One can use the most recent version of Proc Impute to generate multiple imputations via the option "multiple= n" in the control file, where n is the number of imputations for each missing value. Then the question arises: "as the number of imputations increases, do these sets of imputed values adhere to Rubin's PMI criteria?" The answer depends on the data. Proc Impute uses regression to find the optimal combination of predictors. If the involved errors agree with the Gauss-Markov assumption then the least-squares estimator gives an optimal fit of the observations to theoretical models. It would not be difficult to verify that multiple imputations generated by Proc Impute are "proper", since both the observed "combination of predictors" and the observed "distribution of the cases in the range" would converge to the true "combination of predictors" and the true "distribution in the range", respectively. It should also be noted that the average of n estimators based on the n sets of imputed data is asymptotically unbiased (conditionally on the observed data) if the multiple imputation procedure is randomization-valid (Rubin, 1987, p. 116).

Since the design and structure of Proc Impute are fixed, it would not be easy to incorporate Rubin's strategies into the program.

III. EVALUATION OF SCHAFER'S MULTIPLE IMPUTATION SOFTWARE.

Schafer's multiple imputation software consists of three independent parts which perform imputations for continuous variables, categorical variables, and mixed variables, respectively. This section only presents the evaluation of the first part for continuous variables, and the evaluations of the other two parts are under way. From now on, when we use the term "Schafer's software", we always refer to the first part, not the other two. Next, we will first briefly describe the algorithm used in Schafer's software, and then present our evaluation of the software.

ALGORITHM. Suppose that the random vector $\mathbf{X} = (X_1, X_2, \ldots X_p)$ has a multivariate normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and the prior distributions for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are multivariate normal and normal-inverted Wishart, respectively. Then the posterior distributions for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are also multivariate normal and normal-inverted Wishart (Schafer, 1995). It is also assumed that the missing values occur at random (MAR).

First, the software uses the EM algorithm (Dempster, Laird, and Rubin, 1977; Little and Rubin, 1987) to find the Maximum Likelihood Estimates (MLE) of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, which are usually used as the starting values in the iterative simulation step. Then, the software applies the iterative simulation method to simulate one or more iterations of a single Markov chain (Schafer, 1995). Each iteration consists of a random imputation of the missing data drawn from multivariate normal distribution with current parameter values (I-step), followed by a random

[2] For an estimation problem with parameter space Θ, an estimator δ_1 is said to be *Pitman closer* (to θ) than δ_2, if, for every $\theta \in \Theta$, $P_\theta(|\delta_1(X)-\theta| < |\delta_2(X)-\theta|) > 0.5$. This criterion is called *Pitman closeness* or *Pitman nearness* or *Pitman domination.*

draw from the posterior distributions of the parameters, multivariate normal distribution for μ and normal-inverted Wishart distribution for Σ, given the observed data and the imputed data (P-step).

EVALUATION. All evaluations are based on runs conducted on the NCES data set "Administrator Component of 1990-91 School and Staffing Survey" (SASS.AC). These runs were performed in a Pentera (486) environment--90 MHZ clock speed, 16 MB of memory, and 520 MB of hard disk space. Our evaluation focuses on the following six questions (1)~(6).

(1) How many runs would it take to impute all continuous variables in a survey?

Schafer's software is run under an S-PLUS environment. Due to the limit of the dynamic memory in S-PLUS for Windows, a large data set must be partitioned into subsets. The partition strategy is to put variables with high correlations and close scales into the same subset. This strategy makes the convergence criteria in the iterative methods easier to set up and very likely produces more accurate results. The number of variables in each subset depends on the number of cases, while we do not recommend more than 30 variables in any subset. For example, in the SASS.AC data set which has 56 continuous variables and 9907 observations, it can be divided into two subsets with 28 variables apiece.

After we have partitioned the data set into subsets and read each subset into a data matrix, the following runs are required to impute the variables for each subset:

(i) Call function **prelim.norm** to perform preliminary data manipulations;

(ii) Call function **em.norm** to find the MLE for the incomplete data set using the EM algorithm. It returns a vector of parameter estimators which can be used as starting values of parameters for the simulation function "da.norm".

(iii) Call function **da.norm** to simulate one or more iterations of a single Markov chain. It draws parameter estimates from their posterior distributions.

(iv) Call function **imp.norm** to impute the missing values of the data matrix under a user-supplied value of the parameter (e.g. the result of "da.norm") and return a matrix of complete data.

Functions **da.norm** and **imp.norm** can be called multiple times to generate multiple imputations.

(2) How much special processing is required to handle skip patterns?

It is very easy to handle skips with this software. Suppose that "NA" represents the real missing values, "999" stands for the valid skips, and X is the data matrix; then the following four statements can be used to handle the skip patterns:

(i) Record positions of valid skips: **pos_(X==999)**

(ii) Set the valid skips as missing: **X[pos]_NA**

(iii) Impute all the holes (real missing and skips)

(iv) Remove imputed values for skips: **X[pos] _999**

Step (iii) is the imputation step. The other three steps that are used to handle valid skips will take less than one minute for each subset.

(3) How much memory and disk space would be required?

The amount of required memory and disk space depends on the size of the data matrix. We can run the software for a data set with 12,000 cases and 30 variables in a 486 PC environment. When either the number of cases increases to 15,000 (with 30 variables) or the number of variables increases to 50 (with 12,000 cases), the program runs out of dynamic memory. Therefore, in the case of a "large" data set, we must divide the data set into several subsets and run the software on one subset at a time. We experienced that the second run was hung up when we made two runs of Schafer's software in the same S session. It is advisable to quit an S session in which a user has run Schafer's software on a subset with over 10,000 cases and 25 variables, and then enter another S session to run the software on another subset.

(4) How fast is it?

The imputation processing time depends on the size of the data matrix and the steps of iterations specified in functions "em.norm" and "da.norm". Usually, 25 iterative steps will generate quite stable results. For the SASS.AC data, a run with 9,907 cases, 30 variables (a subset) and 25 iterative steps took less than 10 minutes. However, an analyst will devote more time to pre- and post-imputation file-management, such as recoding variables, partitioning a large data set into subsets before imputation, merging imputed subsets into one overall completed file, etc.

(5) How well documented is the software? Is it difficult to use?

Installation instructions provided by Schafer's software are for UNIX workstations and some are not applicable to a PC system. For a PC environment, we first need a WATCOM FORTRAN 77 compiler (WATCOM International Corporation, 1993) to compile the Fortran source files of the software, and then follow the installation instructions (2)-(4) provided by the software. Some errors about storage mode have to be corrected before the Fortran source files can be successfully compiled. Some minor modifications on the random generators of the software are also required to make the simulation functions (e.g., da.norm, imp.norm) work. After making all necessary modifications and correctly installing the software, the software is very easy to use if the user is familiar with S language.

How well documented is the software? By and large, the software is well documented, and the algorithms for the software are especially well developed. However, as Dr. Schafer said, this software is at its early stage and improvement will be made to it. And, as stated above, some modifications to the software are required to make the software work on a PC Windows environment. Moreover, this software does not supply many error messages, so if something does not seem to work, it's largely up to the user to figure out why.

(6) Can the software be adapted to interface easily with SAS?

The immediate answer to this question is "no". Schafer's software is written in S language and run under an S-PLUS environment, and S-PLUS and SAS can not interface with each other. However, with the help of the software DBMS/COPY (Conceptual Software, Inc. 1994), it is very easy to make transformations between SAS data files and S-PLUS data matrices so that one can use Schafer's software to impute the data under an S-PLUS environment and analyze the imputed data under a SAS environment. We may use the S-PLUS "File" pull-down menu to import and/or export SAS data files from within S-PLUS for Windows 3.3. In order to use this tool, we need to add a statement "DBMSCOPY=C:\DBMSCOPY" to the S-PLUS initial file "SPLUS.INI" if DBMS/COPY is installed on the C directory.

When we use DBMS/COPY to transfer a SAS data file to an S-PLUS data frame, say X, X has "list mode", but Schafer's software requires "single mode" of X. Two S-PLUS statements 'X_as.matrix(X)' and 'storage.mode(X) _"single"' can transfer X from "list mode" to "single mode" so that we can apply Schafer's software to it.

IV. A SIMULATION STUDY. We simulate three types of data sets: independent normal data, correlated normal data and independent contaminated data. Each data set has 8 variables (X_1-X_8) and 2000 cases. The first 7 variables have about 10% missing values apiece and the 8th variable has no missing value. Three types of missing mechanisms are considered: (M1) X is randomly missing; (M2) X is missing when Z< c and corr(X, Z)=0.6; and (M3) X is missing when Z< c and corr(X, Z)=0.9. We compare Proc Impute and Schafer's software in terms of average imputing error and mean bias. The average imputing error is defined as $\sqrt{[\Sigma(I_i - R_i)^2/m]}$, where I_i and R_i are imputed values and real values, respectively, and m is the number of missing values. Mean bias is obtained by subtracting the true mean from the imputed sample mean.

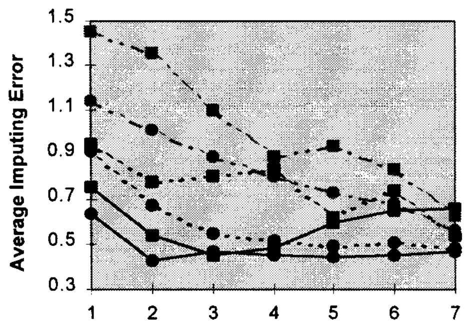
For the independent normal data, neither Proc Impute nor Schafer's software can correct the mean bias caused by the missing values, and the two methods have similar performance in terms of imputing error and mean bias. This is what we expect since the variables in the model provide little information for each other to predict the missing values in this case. So we omit the detailed simulation results for this type of data set and present those for the correlated normal data and independent contaminated data in Figure 1 and 2.

For the correlated normal data, Figure 1(a) shows that Schafer's software has smaller average imputing errors for all 7 variables with missing values for all three types of missing mechanisms. Figure 2(a) shows that, for M1, the two imputed mean biases and the unimputed mean bias are all close to zero, while, for M2, both imputation methods have tremendously improved the mean bias, and Schafer's software is slightly better than Proc Impute. We have similar results for M3 which do not show in figure 2(a).

For the independent contaminated data, Figure 1(b) shows that Proc Impute is better than Schafer's software for some variables, but it is the other way around for the others in terms of average imputing error (in the scale of σ_c, the complete sample standard error). Proc Impute has worse stability of performance. Figure 2(b) shows that, in case of M1, both imputation methods worsen the mean bias since the model assumptions are not satisfied and the data are independent. For M2, it seems that none of the three mean estimators has any advantage over the others.

Figure 1. Comparison of the software in terms of imputing error via simulation with 2000 cases and 8 variables

(a) correlated normal data ($cor(X_i, X_j)=0.1*|i-j|$)



(b) contaminated data (90% normal & 10% Cauchy)

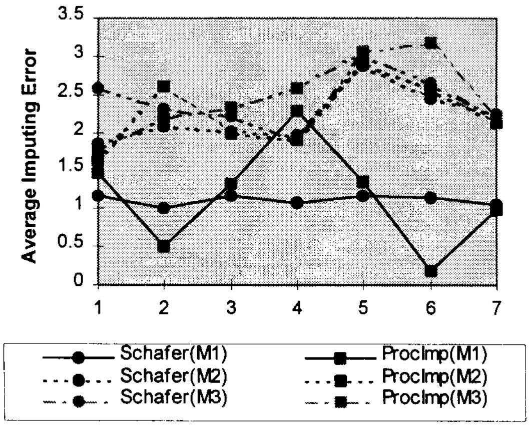
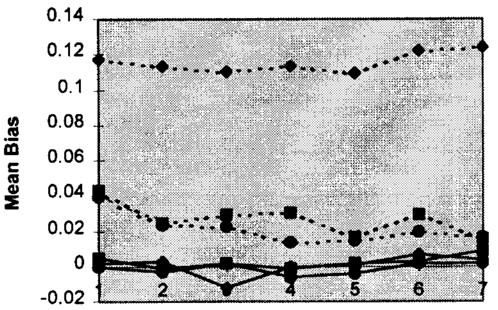
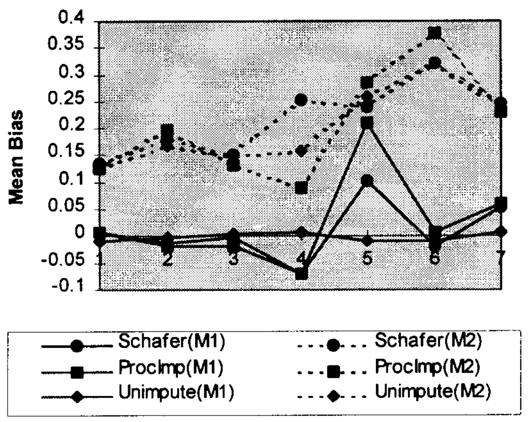


Figure 2. Comparison of the software in terms of mean bias via simulation with 2000 cases and 8 variables

(a) correlated normal data (cor(X_i, X_j)=0.1*|i-j|)



(b) contaminated data (90% normal &10% Cauchy)



V. DISCUSSION. Proc Impute is based on two assumptions: (1) the relations among variables stay the same for the observed cases and missing cases; (2) within each homogeneous subset defined by regression function values, the missing values have the same distribution as the observed values (see section II). These assumptions are almost equivalent to the assumption of missing at random, which is required by Schafer's software. MAR requires that the missing values behave like a random sample of all values within subclasses defined by the observed data. The difference of the assumptions between the two algorithms is that the homogeneous subsets, within which the missing values and observed values have the same distribution, are defined by different measures. The subsets in Proc Impute are constructed by the regression values, but those in Schafer's software are implicitly defined by the observed values. Obviously, the assumptions in both imputation algorithms are less restrictive than missing completely at random (MCAR), which requires that the missing values are a simple random sample of all data values. As seen in the simulation, no matter what the missing mechanism is, both imputation algorithms can significantly improve the mean bias caused by missing values if the variables are highly correlated. Moreover, as long as the normal distribution assumption holds, Schafer's software is always better than Proc Impute in terms of imputing error and mean bias regardless of the missing mechanism.

However, the normal distribution assumption is very important to Schafer's software, which generates imputations directly from the estimated normal distribution. In practice, we may use common transformations (e.g., logarithm, square root, etc.) to make the variables as close to normal as possible, and then apply Schafer's software to it. If no common transformation is available to make the variables close to normal, we may try Proc Impute, whose theoretical assumption is that the residuals in the regression models are normal. Proc Impute should be better than Schafer's software when the variables themselves are not normal but have normal residuals. It should be noted that, in our simulation, even for the contaminated data set, we still have 90% of the data coming from normal distributions. It is possible that Proc Impute will show better performance than Schafer's software if the data further depart from normality.

Although the newest version of Proc Impute allows an analyst to generate multiple imputations, it may not be "proper" in the sense of Rubin's definition. On the other hand, Schafer's software is created for the multiple-imputation purpose; it adheres to the "proper" criterion if the sample is a simple random sample.

The biggest advantage of Proc Impute over Schafer's software is its convenience to run on large data sets. Due to the limit of dynamic memory of S-PLUS for Windows, the working environment for Schafer's software, it usually requires more than one S session and a number of runs to carry out all the imputations for a large data set. But Proc Impute only needs one run (one single batch job) no matter how big the data base is.

REFERENCE

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977), "Maximum Likelihood Estimation from Incomplete Data via the EM Algorithm (with discussion)." *J. of the Royal Statistical Society Series B*, **39**, pp 1-38.

Little, R. J. A. and Rubin, D. B. (1987), *Statistical Analysis with Missing Data.* New York: J.Wiley & Sons.

Rubin, D. B. (1987), *Multiple Imputation for Nonresponse in Surveys.* New York: J. Wiley & Sons.

SAGE (1980), *Guidebook for Imputation of Missing Data.* Prepared for NCES (contract #300-78-150). American Institutes for Research: Palo Alto, CA.

Schafer, J. L (1995), *Analysis of Incomplete Multivariate Data.* To be published.