NORM Users Guide

Multiple imputation of incomplete multivariate data under a normal model

Joseph L. Schafer

Version 2

Development of this software has been supported by grant 2R44CA65147-02 from National Institutes of Health, and by grant 1-P50-DA10075 from the National Institute on Drug Abuse (NIDA). This ongoing work is carried out at The Pennsylvania State University, in the Department of Statistics and at The Methodology Center.

Table of Contents

About NORM

What is NORM?

NORM is a Windows 95/98/NT program for multiple imputation (MI) of incomplete multivariate data. Its name refers to the multivariate normal distribution, the model used to generate the imputations. The main procedures in NORM are:

- an EM algorithm for efficient estimation of mean, variances, and covariances (or correlations); and
- a data augmentation procedure for generating multiple imputations of missing values.

Additional features include:

- pre- and post-imputation processing of data (transformations, rounding, etc.), which can be helpful for imputing certain kinds of non-normal variables;
- plots for monitoring the convergence of data augmentation; and
- a utility for combining the results of a multiply-imputed data analysis, using Rubin's (1987) rules for scalar estimands, to produce overall estimates and standard errors that incorporate missing-data uncertainty. A utility for multiparameter inference is also provided.

Computational routines used in NORM are described by Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data* (London: Chapman & Hall). For more references, see Additional reading.

What does NORM do for me?

Most statistical packages do not handle missing data well. Cases with missing values are typically discarded, resulting in substantial loss of information and perhaps biasing the results in unpredictable ways. NORM helps to solve this problem by performing multiple imputation (MI).

Using information obtained from the observed part of the data set, NORM simulates the missing part $m > 1$ times, creating m equally plausible versions of the complete data. Each of the m data sets is analyzed by standard complete-data techniques. The m sets of results are then combined, using Rubin's (1987) rules for scalar estimands or the extended rules for multiparameter inference to produce one set of estimates and standard errors that incorporate missing-data uncertainty. In most cases, good results can be obtained with even a small number (typically 5-10) of imputations. NORM creates multiple imputations by an algorithm called data augmentation (DA), a special kind of Markov chain Monte Carlo (MCMC) technique.

NORM is not designed to replace well-established statistical packages like SAS or SPSS. It does not perform statistical analyses (e.g. linear or logistic regression) for you. Rather, NORM acts as a pre-processor, filling in the missing values so that other statistical packages can make full use of your data. NORM also acts as a post-processor, combining the output from the m analyses to produce a single set of results.

Authors

NORM was created by Joseph L. Schafer of the Department of Statistics, The Pennsylvania State University, with assistance from Maren K. Olsen.

Acknowledgements

Financial support for the development of NORM was provided by the National Institute on Drug Abuse (NIDA) through award 1-P50-DA10075, Center for the Study of Prevention Through Innovative Methodology.

NORM is free!

NORM was created as a service to the scientific researchers, to encourage the responsible analysis of data with missing values. NORM is distributed free of charge and may be used by anyone if credit is given. The authors assume no liability for its use or misuse.

How to cite NORM

The suggested citation for NORM and this user guide is:

NORM: Multiple imputation of incomplete multivariate data under a normal model (Version 2) [Software] (1999). University Park: The Methodology Center, Penn State. Retrieved from http://methodology.psu.edu

Schafer, J.L. (1999) *NORM users' guide (Version 2)*. University Park: The Methodology Center, Penn State. Retrieved from http://methodology.psu.edu

Additional reading

The statistical methods used in NORM are described in detail in the book:

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall.

A comprehensive reference on multiple imputation is:

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

Less technical introductions to these methods appears in the following articles:

Schafer, J.L. and Olsen, M.K. (1998) Multiple imputation for multivariate missing-data problems: a data analyst's perspective. *Multivariate Behavioral Research*, 33, 545-571.

Schafer, J.L. (1999) Multiple imputation: a primer. *Statistical Methods in Medical Research*, in press.

For more references, see individual Help pages on specific topics, such as EM algorithm and Data augmentation.

Obtaining a copy of NORM

NORM is available from the website:

http://methodology.psu.edu/missing

Go there to download the latest version of NORM for Windows. The file you download will be a self-extracting ZIP archive. Double-click on the file icon within Windows to unpack the files.

Other imputation programs

NORM is the first program in a suite of imputation software for Windows 95/98/NT. The other programs are:

- CAT: imputation of incomplete categorical data using loglinear models;
- MIX: imputation of mixed continuous and categorical data; and
- PAN: imputation of panel data under a multivariate linear mixed model.

These other programs are still under development and will be released as soon as they are ready. They will be distributed with NORM as part of the winimp.exe archive.

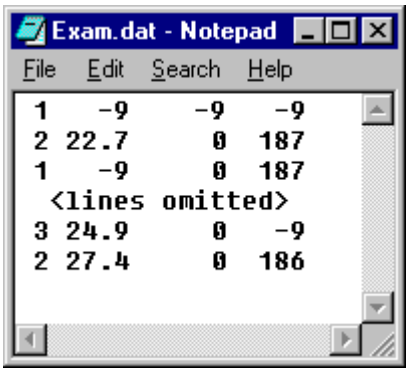
Problems? Questions?

Because NORM is distributed free of charge, our ability to handle users' questions is limited. We are unable to provide detailed advice regarding the use of NORM in specific data sets. In our experience, many questions arise because users are unfamiliar with the new statistical techniques implemented here. Potential users of NORM should first become familiar with these techniques (see Additional reading). You are also encouraged to browse through these Help pages, especially the Frequently asked questions.

Preparing your data

Before using NORM, you must prepare a data file. NORM accepts data only in ASCII (text) form. Data for each individual or sample unit should be placed on a single line. The variables on each line should be separated by one or more blank spaces or tab characters, and each line should end with a carriage return. Missing values must be denoted by a single *numeric* code, such as -9 , -99, or 10000, not by blank spaces, periods, or other non-numeric characters. NORM expects the file to be named *.dat.

Example. The data file EXAM.DAT distributed with this package contains four variables (AGE, BMI, HYP, and CHOL) for 25 individuals. The file looks like this:



Notice that this file is formatted so that each line has the same length and the variables line up nicely in columns. That is not necessary. Any number of blank spaces between variables on a line is allowed.

Limitations. There is essentially no limit to the number of variables or the number of cases in NORM. Very large data sets are not a problem, provided that your computer has enough memory (RAM) to process the data. The only firm limit is that each line of the data file must be less than 2000 characters long, including spaces.

Non-numeric data. Non-numeric or character data are not allowed in the data file. If your data set contains non-numeric variables, you should either (a) convert them to numeric codes or (b) remove them from the data file before using NORM.

Number format. The numbers in the *.dat file may be integers, decimals, or in exponential format (such as 1.56E-02). Embedded commas (as in 10,042) are not allowed.

Variable names. Variable names are not part of the data file. You may provide names for your variables by typing them into NORM after your data file has been read. Alternatively, you may provide them through a variable names file, allowing NORM to read them automatically when your data file is read. For more information. see variable names.

Beginning a session

After starting the NORM program, you may either begin a new session or open an existing session.

New session. Select “New” from the File menu. You will be prompted for the name of the file that contains your data (presumably named *.dat).

Open an existing session. Select “Open” from the File menu. You will be prompted for the name of an existing session (previously saved as a *.nrm file).

Saving your session

You may save your NORM session at any time by selecting “Save” or “Save As” from the File menu. The session is saved to a file called *.nrm.

The data file

When you begin a new NORM session or open an existing one, NORM reads the data from the specified data file. The data file is displayed in a small window on the Data file sheet . As soon as this data file sheet is displayed, you should make sure that the numeric missing value code is correct.

Hint. If you would like to see a larger, more readable version of the data file, use the Display menu to select and display it.

Variables in NORM

The variables in your data set are managed by the variables grid. On this grid you may do the following.

Edit variable names. See variable names.

Select variables for the model. NORM uses a multivariate normal model for estimating parameters (means, variances, and covariances) and for imputing missing values. You may add a variable to the model or remove it by clicking on its box in the “In model” column.

Select variables to be written to imputed data sets. When generating imputations, you may want to include variables in the imputed data sets that were not present in the multivariate normal model (a case ID number, for example). Or you may want to omit certain variables from the imputed data sets that were present in the model. The “In *.imp” column determines which variables are to be written to the imputed data sets.

Apply transformations to variables. See Transforming variables.

Round off variables to a specified precision. See Rounding.

Examine a variable’s distribution. From the variables grid, typing T will tabulate a variable. Typing P will plot the variable. Typing C will display a graphical comparison of the observed and the most recently imputed values.

Hint. The tabulate (T), plot (P) and compare (C) options can also be accessed by right-clicking the mouse.

Transforming variables

When variables are not normally distributed, it often helps to apply transformations before imputing. If a variable is right-skewed, for example, it may be sensible to impute the variable on a square-root or log scale, and then transform it back to the original scale. NORM will perform these functions automatically.

The variables grid has a column called “Transformations”. Clicking on a variable in this column will bring up a window from which you can specify a transformation. The following choices are available:

Power transformations. The Box-Cox family of power transformations which can be helpful for correcting skewness. See Power transformations.

Logit transformations. The logit or logistic transformation applied to an interval from *a* to *b*. Helpful for a variable that takes values only within a limited range (e.g. a proportion or percentage). See logit transformations.

Dummy coding. A categorical variable with *k* levels may be included as a set of (*k*-1) dummy codes, so that associations between this categorical variable and other variables may be preserved. This option may be applied only to categorical variables that are completely observed. See Dummy coding.

Hint. To examine the distribution of a variable before and after transformation, plot the variable. See plotting variables.

Rounding

When NORM creates an imputed version of your dataset (*.imp file), it automatically rounds each variable to a precision which you specify. Rounding, perhaps in conjunction with transformations, helps you to impute values that resemble the observed data.

The variables grid has a column called “Rounding”. Clicking on a variable in this column will bring up a window from which you can specify a rounding method. The following choices are available:

No rounding. The variable will be written to imputed data (*.imp) files as unrounded double precision floating point numbers. This option is not recommended.

Round to nearest x. The variable will be rounded to the nearest multiple of x, where x is specified by the user.

Round to nearest observed value. Each imputed value will be rounded to the nearest observed value for that variable. This option is very useful for imputing binary and ordinal variables. For example, suppose a variable takes values 1, 2, 3, 4, and 5. Rounding to the nearest integer could occasionally produce imputed values of 0 or 6. Rounding to the nearest observed value, however, will ensure that imputed values are 1, 2, 3, 4, or 5.

Hint. Choosing a rounding precision can help you to decide on appropriate endpoints for a logit transformation.

Summarizing data

Important features of any one variable can be seen by tabulating and plotting the variable from the variables grid. But NORM can also report important features of all variables at once, including means, standard deviations, rates and patterns of missingness.

To produce a summary, go to the summarize sheet by clicking on the “Summarize” tab. Select an appropriate name for the file where the output is to be stored and press the “Run” button. The file will be created and displayed in a small window. This summary contains information on all the variables currently in the model.

Hint. To see a larger, more readable version of the summary output file, use the Display menu to select and display it. You can also open it in a text editor (e.g. Notepad) to edit or print it.

Running EM

The EM algorithm in NORM estimates means, variances and covariances using all of the cases in your dataset, including those that are partially missing. Before using NORM to impute missing data, it's almost always a good idea to run EM first.

To run EM, go to the EM sheet by clicking on the "EM algorithm" tab in your NORM session. Then press the "Run" button.

Any run of the EM algorithm will create two files: an output (*.out) file reporting the results of EM, and a parameter (*.prm) file where the resulting parameter estimates are stored. When EM is finished running, the output file is automatically displayed but the parameter file is not. Either of these files may be displayed at any time by using the Display menu.

Experienced users may wish to access the EM computing options via the "Computing…" button.

Running data augmentation

The data augmentation (DA) algorithm in NORM simulates random values of parameters and missing data from their posterior distribution. It is the method by which NORM creates proper multiple imputations for the missing data.

Before running DA , it's a good idea to run EM first. Running EM first will provide a nice set of starting values for the parameters.

To run DA, go to the DA sheet by clicking on the "Data augmentation" tab in your NORM session. Then press the "Run" button.

Any run of DA will create two files: an output (*.out) file reporting the results of DA, and a parameter (*.prm) file where the final simulated values of the parameters are stored. When DA is finished running, the output file is automatically displayed but the parameter file is not. Either of these files may be displayed at any time by using the Display menu.

The number of DA cycles and various other computing options may be set via the "Computing…" button.

Imputation options may be set via the "Imputation…" button.

To diagnose the convergence behavior of DA, NORM allows you to save a stream of simulated parameter values in a parameter series (*.prs) file for later examination and plotting. The series options, which are accessed through the "Series…" button, control how the *.prs file is created.

Creating multiple imputations

In NORM, proper multiple imputations are created through data augmentation. Running data augmentation for *k* iterations, where *k* is large enough to guarantee convergence, produces a random draw of parameters from their posterior distribution. Imputing the missing data under these random parameter values results in one imputation. Repeating the whole process *m* times produces *m* proper multiple imputations.

In practice, one should save the parameters from a data augmentation as a parameter series (*.prs) file and examine series plots to determine a number of iterations *k* by which the autocorrelations have essentially died down.

There are several different ways to use NORM to create a set of *m* proper multiple imputations.

Method 1. First guess a value for *k*. A reasonable guess can be obtained by running EM first and setting *k* equal to or greater than the number of iterations needed for EM to converge. Run data augmentation for a total of $N = mk$ iterations or cycles, producing an imputation at every *k*th cycle. While running data augmentation, save the parameters in a parameter series (*.prs) file. When the run is finished, examine series plots to verify that your presumed value for *k* was large enough.

Method 2. Perform a long run (several thousand cycles or more) of data augmentation, saving the parameters in a parameter series (*.prs) file. Examine series plots to determine a value of *k* large enough for the autocorrelations in all parameters to have died down. Then go to the Impute from parameters sheet in the NORM session and create one imputation from the parameters stored in the *.prs file at iteration *k*. Create another imputation from the parameters stored at iteration 2*k*, and so on.

Impute from parameters

The Impute from parameters sheet allows you to create a random imputation for the missing data under parameter values obtained from a NORM parameter (*.prm) file or a NORM parameter series (*.prs) file.

Running this procedure will create two files: an output (*.out) file reporting the results of the procedure, and an imputation (*.imp) file containing the imputed dataset.

Many users of NORM will never need to use this procedure, because proper multiple imputations are typically created during a data augmentation run. However, there are at least two situations where the Impute from parameters procedure can be quite useful:

- **Generating a single imputation under ML estimates for diagnostic purposes.** See How can I create one imputation for exploratory purposes?.
- **Imputing from parameters stored in a parameter series (*.prs) file.** With NORM, it is possible to run data augmentation and store the parameters in a series (*.prs) file, and later create imputations from any set of parameters in that file. This allows you to do an exploratory run of data augmentation to diagnose its convergence behavior, and then generate imputations without re-running data augmentation.

Series plots

NORM's series plotting procedure allows you to examine the behavior of parameters over the course of a data augmentation run to diagnose the convergence behavior of the algorithm.

To examine series plots, you must first run data augmentation, setting the Series options to store the parameters in a parameter series (*.prs) file.

After the parameter series file has been created, you should open it by selecting "Open" from the "Series" menu on the NORM main window.

Once the series file has been opened, you can create create series plots for single parameters (means, variances and covariances) or for the worst linear function of the parameters by choosing "Plot" from the "Series" menu. It is always a good idea to plot the worst linear function, because if the autocorrelations in this function have died down by *k* cycles, they will almost certainly have died down in the other parameters by *k* cycles as well.

See also: How do I interpret time series and autocorrelation plots?.

MI inference for scalar estimands

Once you have created *m* imputed versions of your data set, you may analyze them in nearly any manner that would be appropriate without missing data. NORM does not perform these analyses for you. However, NORM does offer a helpful facility for combining the results (estimates and standard errors) from your *m* analyses into a single set of results, according to Rubin's (1987) rules for scalar estimands. For joint inferences concerning a group of parameters, use the MI multiparameter inference option.

Running MI infererence for scalar estimands. To invoke this facility, choose "MI inference: scalar" from the "Analyze" menu on the NORM main window. NORM will first prompt you for the name of a single data (*.dat) file containing the estimates and standard errors saved from your *m* analyses, after which an MI scalar inference session window will appear. Select your options in this window and press the"Run" button.

File format. The file containing estimates and standard errors from the *m* analyses is assumed to be an ordinary text (ASCII) file. The estimates and standard errors may be arranged in one of two formats: stacked rows or stacked columns.

Number of estimands and number of imputations. Be sure to indicate the correct number of estimands and number of imputations represented in your file so that NORM can properly read the estimates and standard errors.

Names for estimands. To make the printed results from this procedure more readable, you may wish to provide a names (*.nam) file containing labels or names for the quantities being estimated. See names files.

Results. The results from the MI scalar inference procedure are written to an output (*.out) file and displayed. The results printed for each estimand are:

- overall estimate and standard error;
- t-ratio (the esimate divided by its standard error), appropriate for testing the null hypothesis that the quantity is equal to zero;
- degrees of freedom for the Student's t approximation;
- p-value for testing the null hypothesis that the quantity is equal to zero, against the two-sided alternative hypothesis that it is not zero;
- lower and upper endpoints of a confidence interval;
- and the estimated percent rate of missing information.

MI multiparameter inference

Some statistical questions cannot be addressed by estimates and standard errors for single quantities. For example, suppose that you are performing a regression analysis and want to assess the joint significance of a group of coefficients, testing the null hypothesis that all of them are simultaneously zero,

$$H_0: \beta_1 = \beta_2 = \cdots = \beta_k = 0,$$

versus the alternative that at least one is not zero. An efficient test of this hypothesis should take into account the possible correlations among the estimates for individual parameters.

NORM can perform multiparameter inference using the extension of Rubin's (1987) rules for multidimensional estimands. Once you have imputed and analyzed your data *m* times, NORM will combine *m* vector estimates and *m* covariance matrices to produce a single vector esimate and covariance matrix. NORM will also perform an F-test of the hypothesis that all parameters are simultaneously zero.

Running MI infererence for multiparameter estimands. To invoke this facility, choose "MI inference: multiparameter" from the "Analyze" menu on the NORM main window. NORM will first prompt you for the name of a single data (*.dat) file containing the estimates and covariance matrices saved from your *m* analyses, after which an MI multiparameter inference session window will appear. Select your options in this window and press the"Run" button.

File format. The file containing estimates and covariance matrices from the *m* analyses is assumed to be an ordinary text (ASCII) file. The estimates and covariance matrices are assumed to be arranged in a stacked matrix format.

Number of estimands and number of imputations. Be sure to indicate the correct number of estimands and number of imputations represented in your file so that NORM can properly read the estimates and covariance matrices.

Names for estimands. To make the printed results from this procedure more readable, you may wish to provide a names (*.nam) file containing labels or names for the quantities being estimated. See names files.

Results. The results from the MI multiparameter inference procedure are written to an output (*.out) file and displayed. The printed results include:

- the overall estimate and standard error for each quantity;
- overall covariance matrix;
- average relative increase in variance due to nonresponse;
- average percent rate of missing information;
- results from the F-test of the null hypothesis that all parameters are simultaneously zero, versus the alternative that at least one is not zero.

Statistical assumptions

NORM applies three types of statistical assumptions to your data.

- Missing values are assumed to be missing at random (MAR). This is also called the ignorability assumption.

- The variables in your dataset are jointly normally distributed. For more comments on this, see Limitations of the normal model and Imputer's versus analyst's model.

- The data augmentation algorithm is a Bayesian procedure and thus relies on a prior distribution for the unknown parameters (means, variances, and covariances). NORM allows you to choose either a standard noninformative prior or a ridge prior.

Imputer's versus analyst's model

An imputation model should be chosen to be (at least approximately) compatible with the analyses to be performed on the imputed datasets. In particular, the model should be rich enough to preserve the associations or relationships among variables that will be the focus of later investigation.

For example, suppose that you use NORM to impute a variable *Y* under a model that includes the variable *X*. After imputation, an analyst uses linear regression to predict *Y* from *X* and another variable *Z* which was not in the imputation model. The estimated coefficient for *Z* from this regression would tend to be biased toward zero, because *Y* has been imputed without regard for its possible relationship with *X*.

In general, any variable(s) that may prove important in subsequent analyses should be present in the imputation model.

The converse of this rule, however, is not at all necessary. If Y has been imputed under a model that includes *Z*, there is no need to include *Z* in future analyses involving *Y* unless the *YZ* relationship is of substantive interest. Results pertaining to *Y* cannot be biased by the inclusion of extra variables in the imputation phase. Therefore, a rich imputation model that preserves a large number of associations is desirable because it may be used for a variety of post-imputation analyses.

Limitations of the normal model

The normal model used by NORM assumes that each variable in your dataset is normally distributed. In practice, transformations can often help you to produce reasonable imputations for variables that are continuous and non-normal. Binary or ordinal variables can often be imputed as normal and then rounded off to the nearest observed value. Fully observed categorical variables can be replaced by a set of dummy codes to preserve their associations with other variables.

Interactions. One important feature of the normal model is that it does not preserve interactions among variables. An interaction means that the relationship (e.g. correlation) between two variables varies by levels of a third variable. If your post-imputation analyses will involve interactions, you may need to take special measures to ensure that the interactions of interest are preserved in the imputations. For example, suppose that interactions involving a binary variable GENDER are of interest and GENDER is fully observed. You could split the dataset by GENDER, imputing males and females separately; this will automatically preserve all two-way interactions involving GENDER.

Frequently asked questions

Why should I run EM before imputing?

Before using NORM to impute missing data, it's almost always a good idea to run EM first. Running EM first will:

Provide good starting values for DA. The data augmentation (DA) algorithm, which is used to create imputations, requires starting values for the parameters. The parameter estimates produced by EM provide excellent starting values.

Help you to predict the likely convergence behavior of DA. The convergence behavior of both EM and DA are related to the rates of missing information in your data. If EM converges quickly then DA is likely to converge quickly. See note on convergence of data augmentation.

How many imputations do I need?

In most cases, good results can be obtained with only a small number of imputations. Rubin (1987) shows that the efficiency of an estimate based on *m* imputations is approximately

$$\left(1+\frac{\gamma}{m}\right)^{-1}$$

where γ is the rate of missing information. The table below shows the percent efficiency achieved for various rates of missing information and values of *m*. Unless the fraction of missing information is unusually high, the efficiency gained by taking more than 5-10 imputations is minimal.

	γ				
m	0.1	0.3	0.5	0.7	0.9
3	97	91	86	81	77
5	98	94	91	88	85
10	99	97	95	93	92
20	100	99	98	97	96

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

How can I create one imputation for exploratory purposes?

Suppose that you generate *m* imputations and then analyze the data by regression or some other modeling procedure. How should you decide on an appropriate model? Or how should you explore the data or check the modeling assumptions, e.g. by diagnostic residual plots? Applying formal or informal model-selection procedures to each of the *m* imputed data sets could potentially result in *m* different models. And repeating an exploratory analysis on *m* datasets can be very tedious.

In these situations, it can be quite useful to create one special imputation for exploratory purposes. First, run the EM algorithm to obtain maximum likelihood (ML) or posterior mode estimates of the model parameters. Then use the Impute from parameters procedure to generate one imputation under these estimates.

When a single imputation is generated in this manner, any quantities estimated from this dataset will tend to be close to estimates obtained from a set of *m* proper imputations. Exploratory or diagnostic plots produced from this dataset will be typical of plots created from any one of the imputed datasets. Statistical significance of effects will be somewhat overstated, however, because this single imputation will not properly reflect parameter uncertainty. Model selection procedures applied to this data set will tend to detect all significant or important effects, and perhaps also some unimportant ones.

If you use this one imputed dataset to select a model, you should then discard the results, refit the model to a set of *m* proper multiple imputations, and obtain proper estimates and standard errors using Rubin's rules for scalar estimands as implemented in NORM's MI inference for scalar estimands procedure.

How do I interpret time series and autocorrelation plots?

Invoking the series plot procedure will display a window containing two plots: A time series plot for the parameter in question, and a plot of the autocorrelation function for that parameter. The latter estimates the correlation between that parameter's value at any iteration or cycle and its value *k* cycles later for *k* = 1, 2, …

These plots help you to identify how many cycles are needed for the parameter to become approximately independent of its initial value.

- Each autocorrelation plot displays a sequence of critical values (red band) beyond which the autocorrelations are significant at the .05 level.
- For slowly converging series, a large number of cycles may be needed to obtain stable estimates of the autocorrelation.
- Under ordinary circumstances, one would expect the autocorrelation to start at some positive value, quickly or gradually drop to zero as the lag increases, and remain at zero for higher lags.
- A plot showing oscillating or negative autocorrelations suggests that the estimated autocorrelations are too noisy, and a longer series is required to estimate them well.

Example 1. A rapidly converging series.

Example 2. A slowly converging series.

Example 3. A pathological situation where data augmentation fails to converge.

Example 4. A situation where the series is not long enough to estimate the autocorrelations well.

Hint. Use the "Properties" menu item on the plot window to change basic properties of the plot, such as the maximum lag for which autocorrelations are displayed.

About data augmentation

Data augmentation (DA) is an iterative simulation technique, a special kind of Markov chain Monte Carlo (MCMC). In DA there are three types of quantities: observed data, missing data, and parameters. The missing data and parameters are unknown. DA alternately performs the following steps:

- I-step: Impute the missing data by drawing them from their conditional distribution given the observed data and assumed values for the parameters; and
- P-step: Simulate new values for the parameters by drawing them from a Bayesian posterior distribution given the observed data and the most recently imputed values for the missing data.

Alternating between these two steps sets up a Markov chain that converges to a stationary distribution, the joint distribution of the missing data and parameters given the observed data. DA bears a strong resemblance to the EM algorithm, and may be regarded as a stochastic version of EM. It is useful for multiple imputation of missing data. By running DA for a large number of cycles, and storing the results of a few I-steps along the way (with enough cycles in between to ensure independence), one obtains proper multiple imputations of the missing data.

See also convergence of data augmentation.

Tanner, M.A. and Wong, W.H. (1987) The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, 82, 528-550.

Convergence of data augmentation

Data augmentation (DA) converges to a *distribution* of values rather than a single set of values. To say that “DA has converged by *k* cycles” means that the simulated quantities (missing data and parameters) at cycle *t* and the simulated quantities at cycle *t+k* are statistically independent of each other.

The convergence behavior of DA is governed by the rates of missing information (how much information about the parameters is contained in the missing part of the data relative to the observed part). High rates of missing information cause successive iterates to be highly correlated, so that a large number of cycles will be needed for the dependence to die down. Low rates of missing information produce low correlation and rapid convergence.

When using DA to create multiple imputations, one needs to allow enough cycles between imputations to ensure that they are statistically independent. Series plots are very useful for judging how many cycles may be needed. A preliminary run of the EM algorithm is also useful, because the convergence behavior of EM is closely related to that of DA. A useful rule-of-thumb: If the EM algorithm converges by *k* cycles, then it is very likely that DA will achieve independence by *k* cycles.

About the EM algorithm

The EM algorithm is a general method for obtaining maximum-likelihood estimates of parameters from incomplete data. EM iterates between the following two steps:

- E-step: Replace missing sufficient statistics by their expected values given the observed data, using estimated values for the parameters; and
- M-step: Update the parameters by their maximum-likelihood estimates, given the sufficient statistics obtained from the E-step.

The convergence behavior of EM is related to the rates of missing information (how much information about the parameters is contained in the missing part of the data relative to the observed part). High rates of missing information can lead to slow convergence; low rates lead to rapid convergence.

Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977) Maximum-likelihood estimation from incomplete data via the EM algorithm (with discussion*). Journal of the Royal Statistical Society Series B*, 39, 1-38.

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall, Chapter 3.

About Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a collection of techniques for simulating random draws from difficult probability distributions via Markov chains. In MCMC, one sets up a Markov chain (a sequence of random draws, where the distribution of each draw depends on the previous one) that converges to the desired target distribution. MCMC is especially useful in Bayesian statistical analyses and for solving difficult missing-data problems. Examples of MCMC include the Metropolis-Hastings algorithm, the Gibbs sampler, and data augmentation.

Gilks, W.R., Richardson, S., and Spiegelhalter, D.J., eds. (1996*) Markov chain Monte Carlo in Practice*. London: Chapman & Hall.

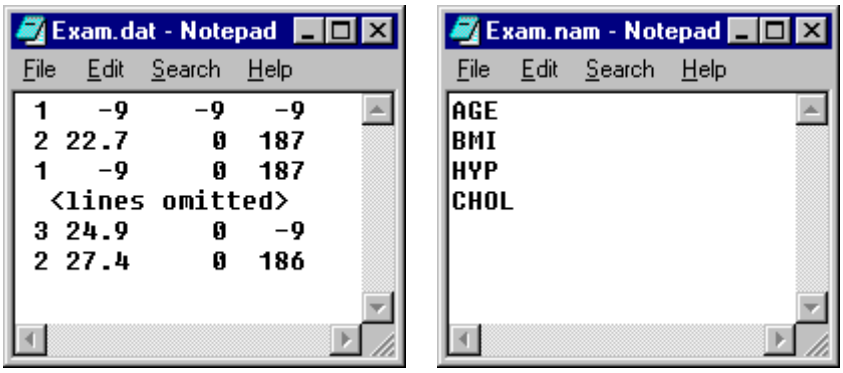
Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data* (London: Chapman & Hall), Chapter 3.

Variable names

Variable names in NORM may be up to eight characters long. There are two different ways to enter variable names in NORM.

Type them in. After a data file has been read by NORM, go to the variables grid and edit the variable names manually.

Provide a names file. Alternatively, you may provide variable names as a file, allowing NORM to read them automatically as the data are being read. The names file should have the same name as the data file, except that it should end with *.nam rather than *.dat. The names file should be an ordinary text (ASCII) file with each variable name appearing on a separate line. For example, the sample data file EXAM.DAT distributed with this package and the corresponding names file EXAM.NAM look like this:

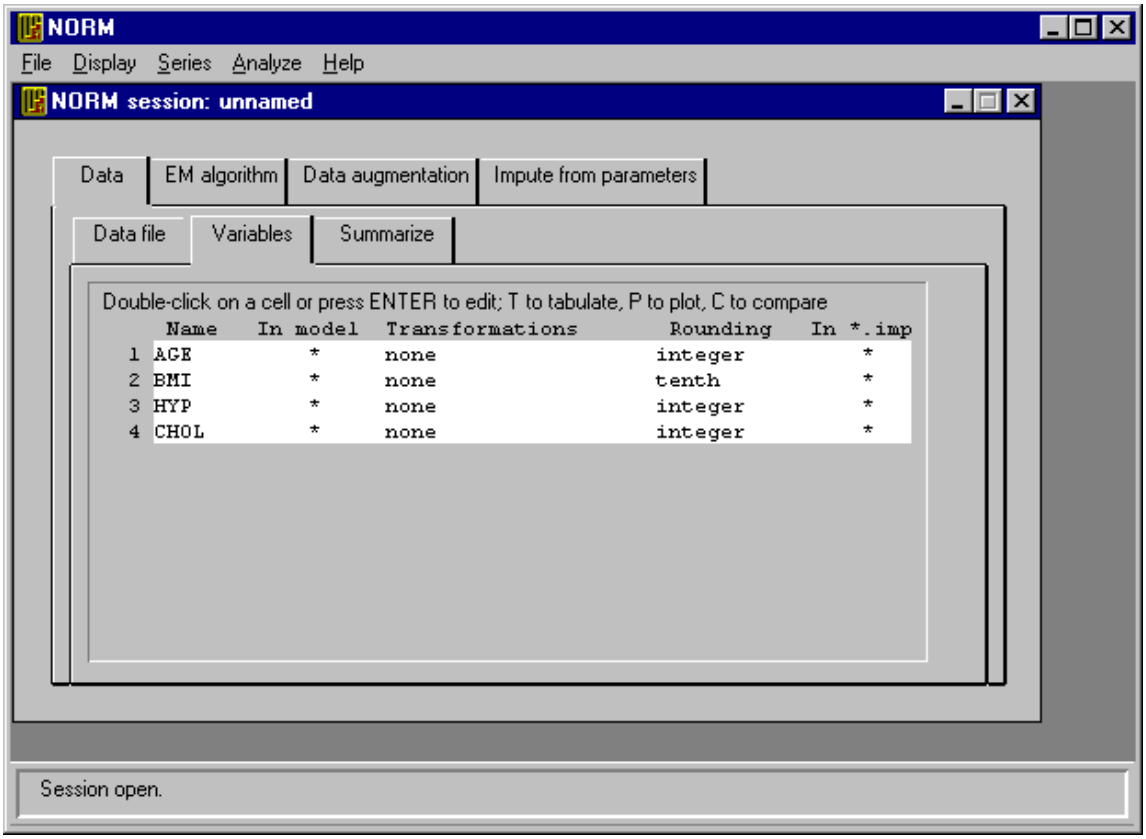


Whenever a *.dat file is opened, NORM looks for a corresponding *.nam file in the same directory or folder. If such a file exists, the variable names will be read automatically.

Note: If you provide a *.nam file, you may still edit the variable names afterward by going to the variables grid.

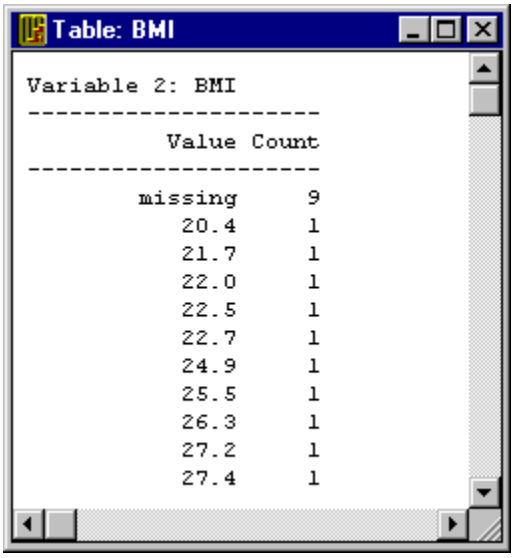
Variables grid

The variables grid, which can be reached by clicking on the “Variables” tab in the NORM session, allows you to interactively change important properties of the variables in your data set.



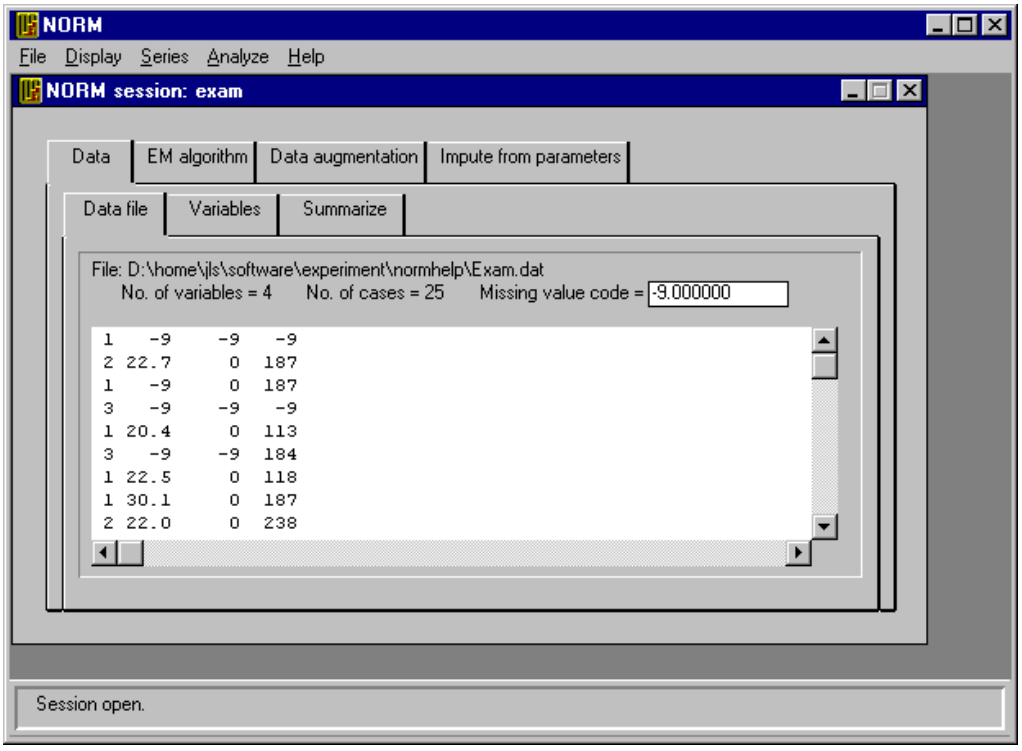
Tabulating variables

Typing T from the variables grid will display a tabular summary of the distinct values for a variable. You can also tabulate a variable by right-clicking the mouse.



Data file sheet

The data file sheet can be reached by clicking on the “Data file” tab in the NORM session. It displays the data file, number of cases and number of variables. It also allows you to specify the numeric missing value code.



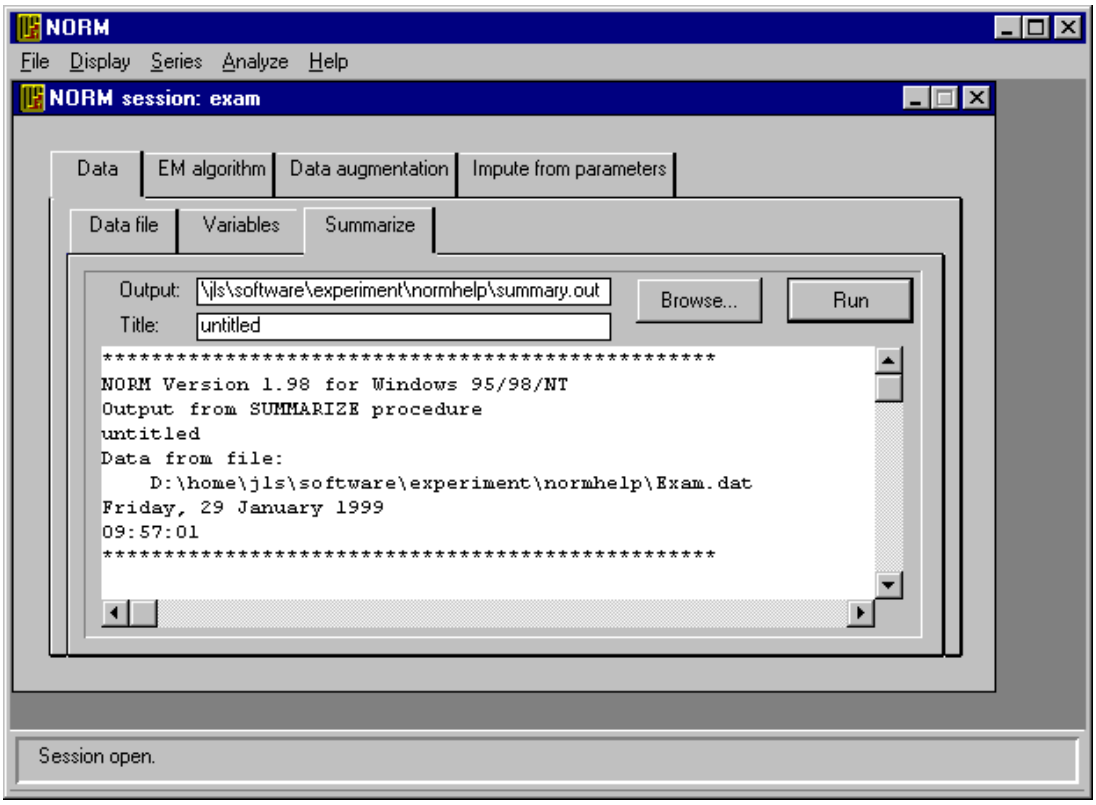
Missing value code

NORM assumes that missing values are represented by a single numeric code throughout the dataset. For example, if the code is –99, then any observation of –99 (or –99.0, –99.000, etc.) for any variable is interpreted as a missing value.

Hint. When preparing a data file for NORM, it makes sense to choose a missing value code that is well outside the natural range of possible values for all the variables. Otherwise some non-missing values might accidentally be interpreted as missing.

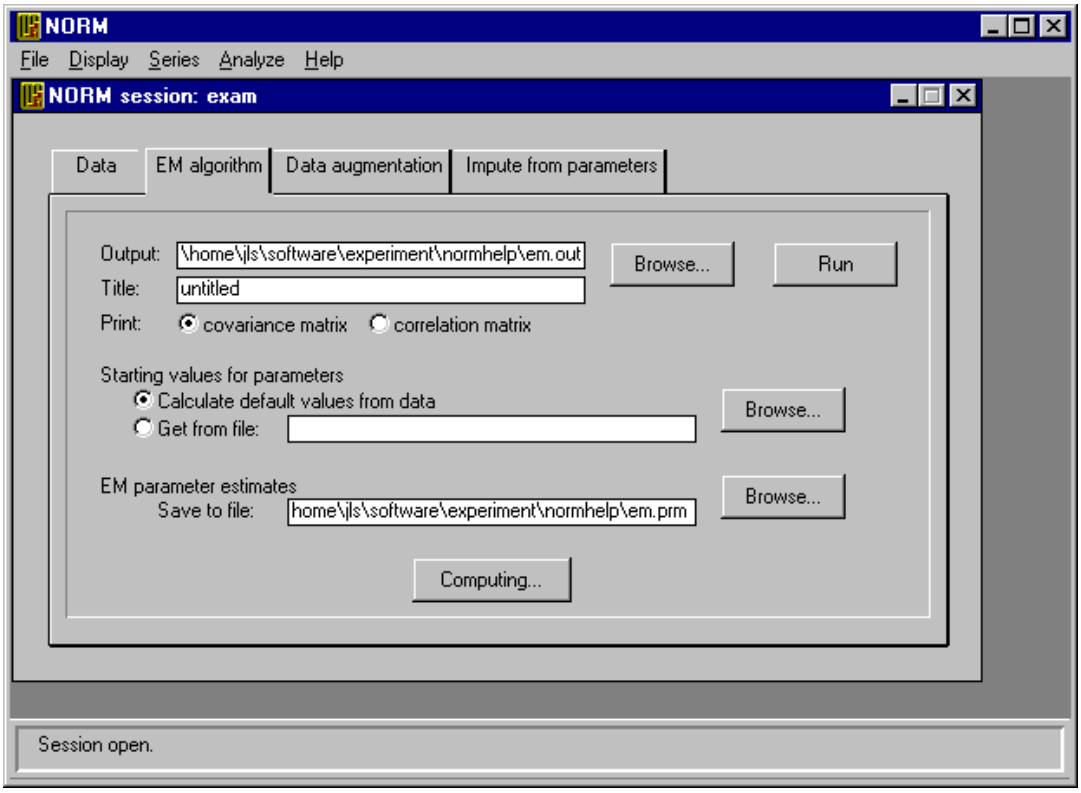
Summarize sheet

The summarize sheet can be reached by clicking on the “Summarize” tab in the NORM session. Pressing the “Run” button will create a file summarizing all the variables currently in the model.

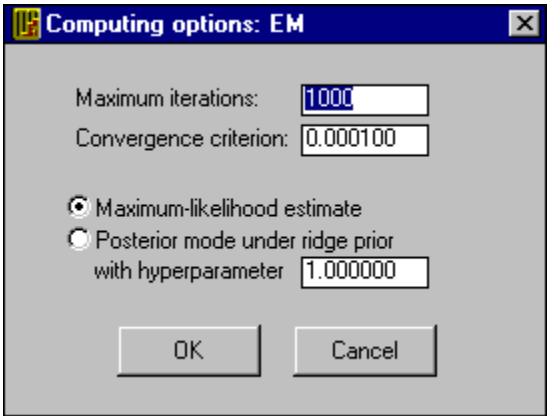


EM sheet

The EM sheet can be reached by clicking on the “EM algorithm” tab in the NORM session. Pressing the “Run” button will initiate a run of the EM algorithm to estimate means, variances, and covariances for the variables currently in your model. The “Computing…” button provides access to EM computing options.



EM computing options



Maximum iterations. If EM has not converged by this many iterations, the procedure will halt .

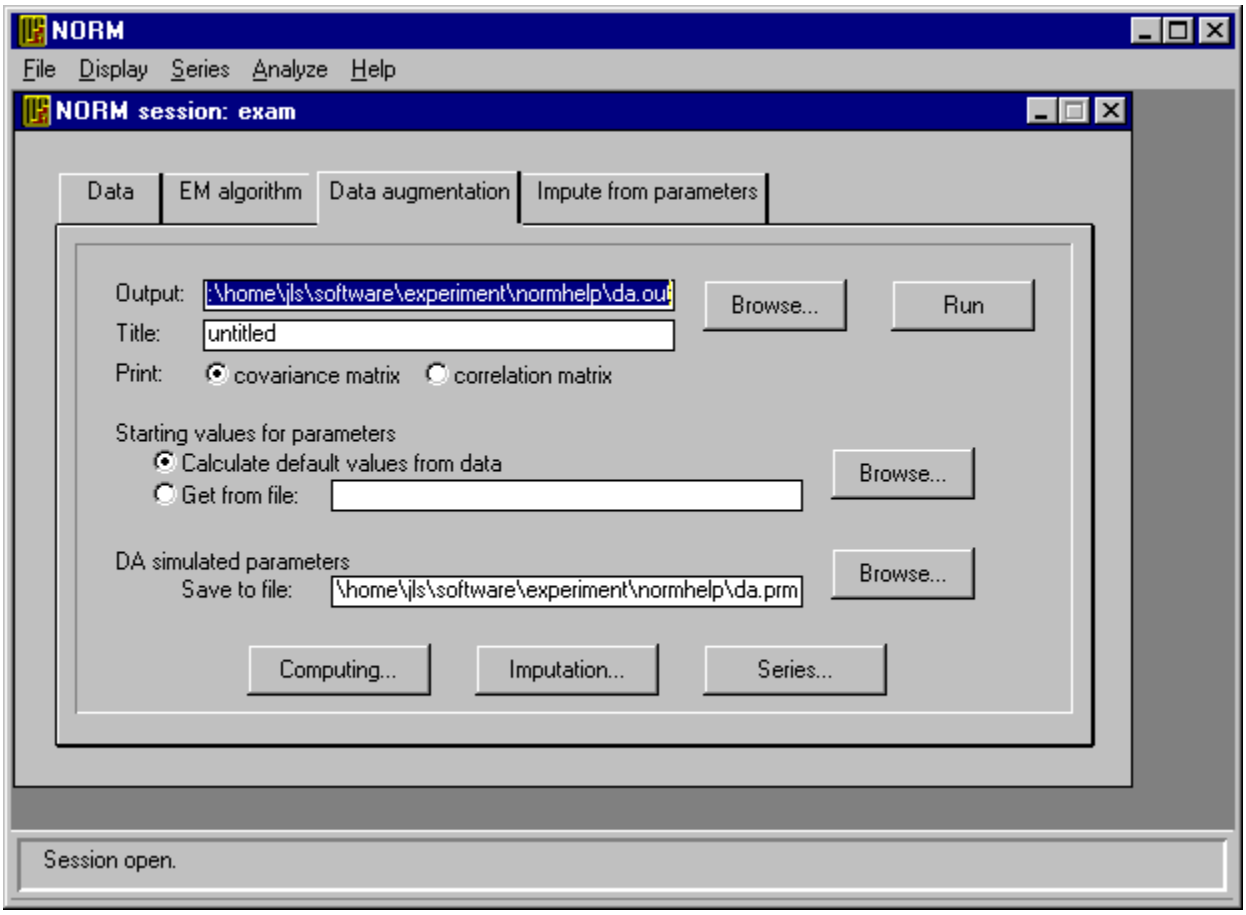
Convergence criterion. The maximum relative change in the value of any parameter from one cycle to the next.

Estimation method. By default, EM will calculate maximum likelihood (ML) estimates. But it can also be used to find a posterior mode under a ridge prior, which can be helpful for dealing with poorly estimated or unidentified parameters.

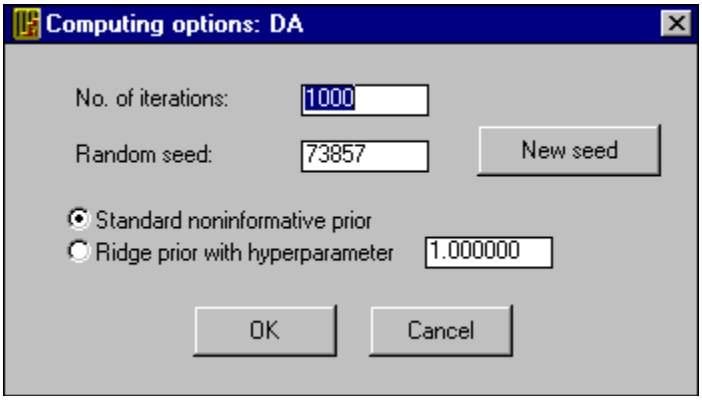
Data augmentation sheet

The data augmentation (DA) sheet can be reached by clicking on the “Data augmentation” tab in the NORM session. Pressing the “Run” button will initiate a run of data augmentation to simulate parameters and, if you desire, create multiple imputations.

The “Computing…”, “Imputation…” and “Series…” buttons provide access to computing, imputation, and series options, respectively.



DA computing options

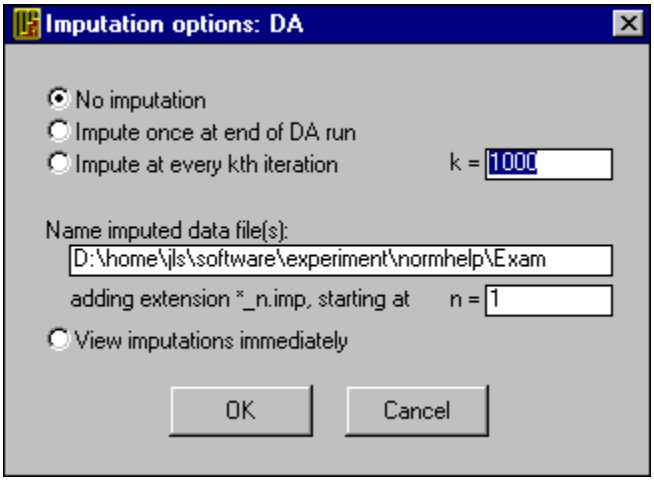


Number of iterations. Number of cycles of data augmentation to be performed. Each cycle consists of an imputation or I-step followed by a posterior or P-step.

Random seed. A positive integer value that initializes the pseudorandom number generator. A random value for this seed is chosen automatically when the session is created. A new random seed can be obtained by pressing the “New seed” button.

Prior distribution. By default, a standard noninformative prior is used. You also have the option of applying a ridge prior, which can be helpful for dealing with poorly estimated or unidentified parameters.

DA imputation options



These options determine how imputed datasets (*.imp files) are generated and stored.

No imputation: No imputations are saved. This option is useful for an exploratory run of data augmentation,

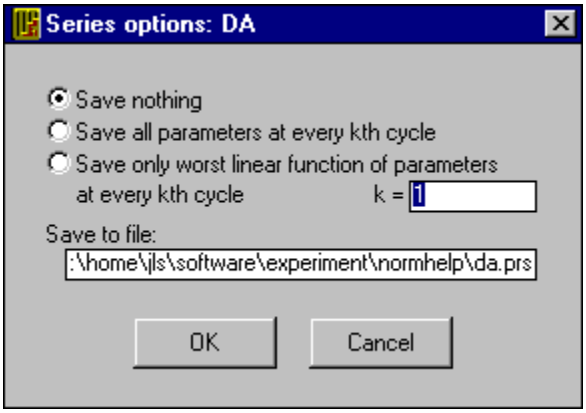
Impute once at end of DA run: Only the final imputations from the last cycle of data augmentation are saved.

Impute at every kth iteration: Save the imputations from every kth cycle of data augmentation. By setting k large enough to ensure convergence, you can produce any number of proper multiple imputations.

Name imputed data file(s): Set the pattern for imputed data set (*.imp) file names.

View imputations immediately: Display each *.imp file in a separate window after it is generated.

DA series options



These options determine whether and how the simulated parameters from data augmentation are saved to a parameter series (*.prs) file for later examination and plotting.

Save all parameters at every kth cycle: All parameters (means, variances, and covariances) are saved to the *.prs file. This will allow you to display time series and autocorrelation plots for individual parameters. The advantage of this option is that if data augmentation exhibits slow convergence or erratic behavior, individual plots may help you to diagnose which parameters may be causing difficulty (e.g. because they may be unidentified). The disadvantage of this option is that the *.prs file may become very large, especially if the increment k is set to a small value.

Save only worst linear function of parameters: Under this option, only the worst linear function of the parameters is saved to the *.prs file. This will allow you to examine the time series and autcorrelation plot for only this one function.

Worst linear function of parameters

The worst linear function is a weighted sum of the model parameters (means, variances and covariances), weighted by how much they change over a single step of the EM algorithm. When EM is near its stationary (convergent) solution, this sum approximates a function for which the rate of missing information is highest. Parameters that change a lot in one step have a high rate of missing information; those that change little have a low rate of missing information.

Series plots for the worst linear function may help to quickly assess the convergence behavior of the data augmentation. If the autocorrelation in this series appears to have died down by *k* iterations, then it is likely (but not certain) that the autocorrelations for all individual parameters have died down by k iterations as well. The exception to this rule occurs when one or more parameters are unidentifed. Under most circumstances, you will be able to diagnose the convergence behavior of data augmentation algorithm by looking at plots for just the worst linear function of the parameters, rather than all parameters.

How NORM calculates the worst linear function. Whenever you run the data augmentation algorithm from a given set of starting values, NORM will first take a single step of EM to see how much the initial parameter values change; these changes are then used as weights in calculating the worst linear function.

Hint. For best results, begin your run of data augmentation with starting values obtained from the EM algorithm This will ensure that the starting values are very close to the stationary point of EM, so that the worst linear function calculated by NORM will be most meaningful.

Noninformative prior distribution

By default, NORM uses a standard noninformative prior distribution in its data augmentation procedure. The "density" function, which is not really a proper density, is proportional to

$$|\Sigma|^{-(p+1)/2}$$

where Σ is the covariance matrix and *p* is the number of variables in the model. This function, which may be motivated by the Jeffreys invariance principle, is the standard noninformative prior for Bayesian analyses involving the multivariate normal distribution.

Using a noninformative prior in data augmentation. Press the "Computing…" button on the data augmentation sheet to bring up the DA computing options window.

Ridge prior

A ridge prior can be used to stabilize the estimation of parameters. An unfortunate pattern of missing and observed values in a dataset may cause certain parameters to have high rates of missing information or to be entirely unidentified. This can lead to the following problems:

- If one or more parameters have very high rates of missing information, then the EM and data augmentation algorithms will converge very slowly.
- If parameters are unidentified, then EM may converge quickly but the resulting estimates will not be unique; running EM from different starting values will produce different estimates. Data augmentation, on the other hand, may never converge; parameters may drift to implausible values as seen in series plots.

These problems can often be ameliorated by introducing a small amount of prior information.

NORM allows you to select a ridge prior, as described by Schafer (1997, Chapter 5), which tends to shrink estimated correlations toward zero. The degree of shrinkage is specified by the hyperparameter, a positive real number which corresponds roughly to the number of prior observations being introduced; the higher the value for the hyperparameter, the greater the shrinkage.

Using a ridge prior in EM. Press the “Computing…” button on the EM sheet to bring up the EM computing options window.

Using a ridge prior in data augmentation. Press the “Computing…” button on the data augmentation sheet to bring up the DA computing options window.

Selecting the hyperparameter. There are no strict rules for selecting the hyperparameter. A small value is desirable, but if it is too small then the problem might not be solved. In practice, you should probably use a value no greater than 5% or 10% of the actual sample size or number of cases in your dataset, to avoid introducing large biases in correlations that are well estimated.

Unidentified parameters

A parameter is said to be *unidentified* if the observed data contain no information about it.
For example, if a variable is entirely (100%) missing, then its mean, variance, and covariances with other variables are unidentified.

NORM does not allow you to model variables that are entirely missing. However, other parameters may be unidentified due to an unfortunate pattern of missing and observed values. For example, suppose that 50% of your cases have variables X and Y observed, and the other 50% have X and Z observed, but there are no cases for which Y and Z are observed jointly. In this example, the partial correlation between Y and Z given X is unidentified.

For another example, suppose that X is binary, taking the values 0 or 1, and Y is missing whenever X=0. The data contain no information on the relationship between X and Y.

When parameters are unidentified, the EM algorithm may converge quickly but the estimates are not unique. Different starting values may produce different answers.

Unidentified parameters are especially troublesome for data augmentation. Series plots for these (and closely related) parameters may drift aimlessly because the algorithm never converges.

How to detect the problem. Often the first sign of the problem comes when you view series plots from data augmentation. If you run the EM algorithm from different sets of starting values, and EM converges to different parameter estimates *with the same log likelihood value*, then you have unidentified parameters. **Hint.** An easy way to vary the starting values for EM is to run data augmentation. Data augmentation generates random values for the parameters which are stored in a parameter (*.prm) file; this file can then be specified as the source of starting values on the EM sheet.

How to handle the problem. One way to handle unidentified parameters is to simplify your model by removing unnecessary variables. Another way is to introduce a modest amount of prior information by selecting a ridge prior.

Power transformations

Power transformations are useful for correcting skewness. Box and Cox (1964) defined a family of power transformations which varies continuously as a function of the power (exponent) parameter and which includes the natural logarithm as a special case. The Box-Cox transformation of a variable *y* is

$$y' = \begin{cases} (y^{\lambda} - 1) / \lambda & \text{for } \lambda \neq 0, \\ \log y & \text{for } \lambda = 0. \end{cases}$$

Choosing $\lambda < 1$ helps to correct positive (right) skewness, whereas $\lambda > 1$ helps to correct negative (left) skewness.

Shifting. Box-Cox transformations should be applied only to positive data. If the variable *y* contains negative or zero values, it is customary to use

$$y' = \begin{cases} ((y + c)^{\lambda} - 1) / \lambda & \text{for } \lambda \neq 0, \\ \log(y + c) & \text{for } \lambda = 0, \end{cases}$$

where *c* is a shift parameter chosen large enough to ensure that (*y* + *c*) > 0 for all observed values of y.

Technical note. The reverse transformation is given by

$$y = \begin{cases} (\lambda y' + 1)^{1/\lambda} - c & \text{for } \lambda \neq 0, \\ (\exp y') - c & \text{for } \lambda = 0. \end{cases}$$

Except when $\{\lambda = 0$, this reverse transformation requires that

$$(\lambda y' + 1) > 0$$

which is not guaranteed to happen when values for the transformed variable are drawn from a normal distribution. When NORM is imputing a power-transformed variable and generates an imputed value that violates this condition, the anomalous value is discarded and re-drawn.

Box, G.E.P. and Cox, D.R. (1964) An analysis of transformations (with discussion). *Journal of the Royal Statistical Society Series B*, 26, 211-252.

Logit transformations

The logit or logistic transformation of a variable *y*, defined as

$$y' = \log\left(\frac{y}{1-y}\right),$$

converts values of *y* within the interval (0,1) to numbers that may fall anywhere on the real line. The scaled logit transformation, defined as

$$y' = \log\left(\frac{(y-a)/(b-a)}{1-(y-a)/(b-a)}\right),$$

does the same thing for values of *y* within the interval (*a*,*b*). This transformation is useful for a variable that takes values only within a limited range, such as a proportion or a percentage.

Hint. By choosing an appropriate rounding precision and endpoints *(a,b)* slightly beyond the range of your data, you can ensure that the imputed data will have precise minimum and maximum values. For example, suppose one of your variables represents a percentage, taking possible values 0, 1, 2, …, 100. Choosing a logit transformation with *a* = -0.5 and *b* = 100.5 and rounding to the nearest integer will ensure that all imputed values are integers between 0 and 100, inclusive.

Dummy coding

Consider a categorical variable *y* that takes values 1, 2, …, *k* (or any other set of *k* numbers). If *y* has no missing values, you may include it in the model as a set of *k*-1 dummy codes, so that associations between *y* and other variables will be preserved in the imputed data sets. If dummy coding is selected, then NORM replaces *y* with the codes

$$D_j = \begin{cases} 1 & \text{if } y = j, \\ 0 & \text{otherwise} \end{cases}$$

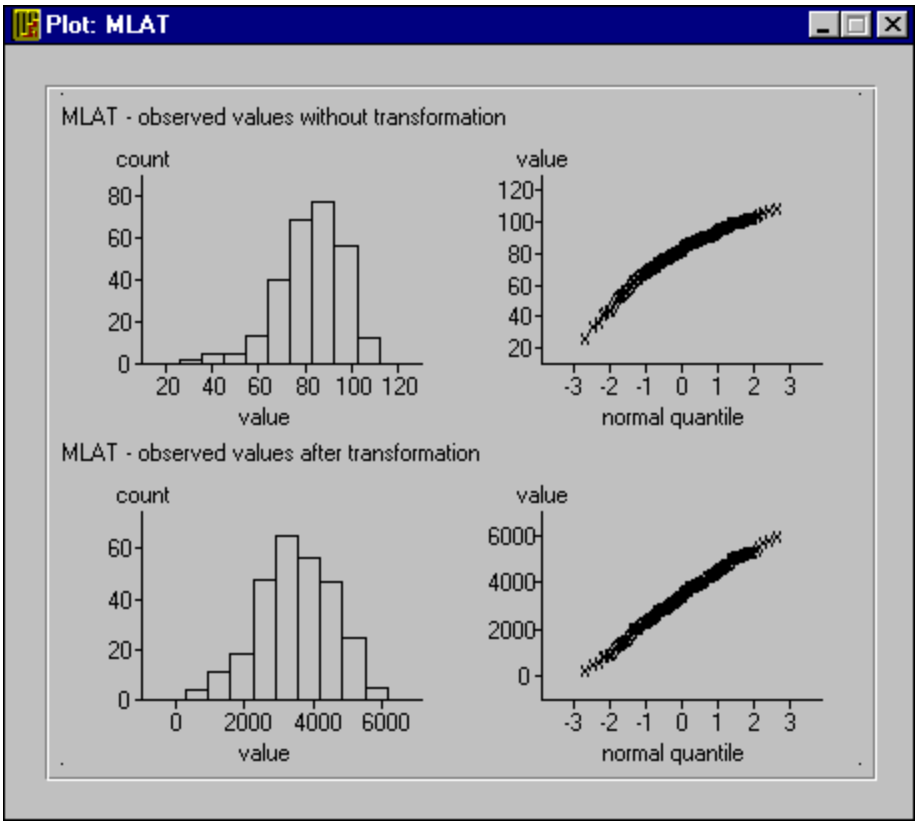
for $j = 1, \ldots, k - 1$ when running EM or data augmentation.

Imputed data sets. When NORM creates imputed data sets, it is the original variable y rather than the dummy codes that is written to the *.imp files.

Ordered categories. If the values of *y* represent ordered categories, you may wish to include *y* in the model without recoding, which will preserve linear associations between *y* and other variables. An advantage of this method is that it may be used when y has missing values. If you do this, you will probably want to round off the continuous imputed values to the nearest observed values of *y* (see Rounding).

Plotting variables

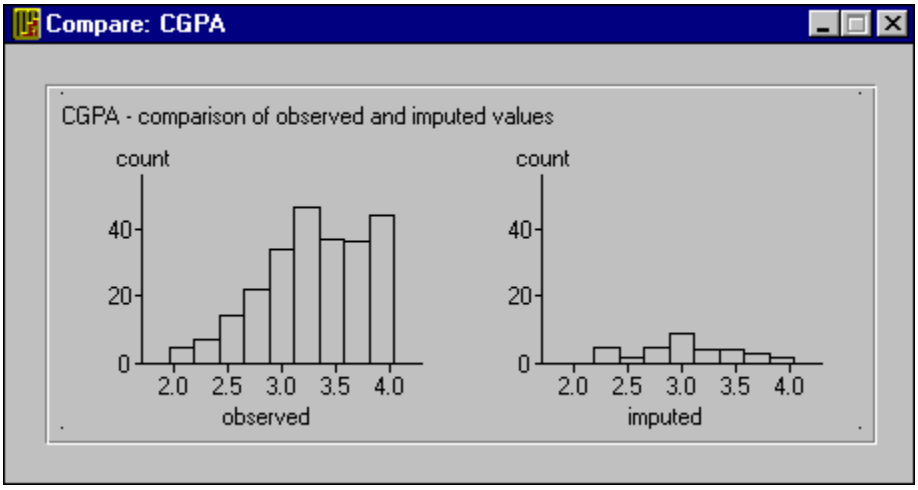
Typing P from the variables grid allows you to view the distribution of any variable, which can help you to decide whether a transformation is needed. This option brings up a graphics window containing a histogram and a normal probability plot. If a transformation has been selected, the same information is displayed both for the original and the transformed versions of the variable. The plot option can also be accessed by right-clicking the mouse.



Comparing observed and imputed values

Typing C from the variables grid allows you to graphically compare the observed and most recently imputed values for a variable. This option brings up a graphics window with side-by-side histograms. It helps you to identify situations where imputed values fall outside the range of plausibility. The compare option can also be accessed by right-clicking the mouse.

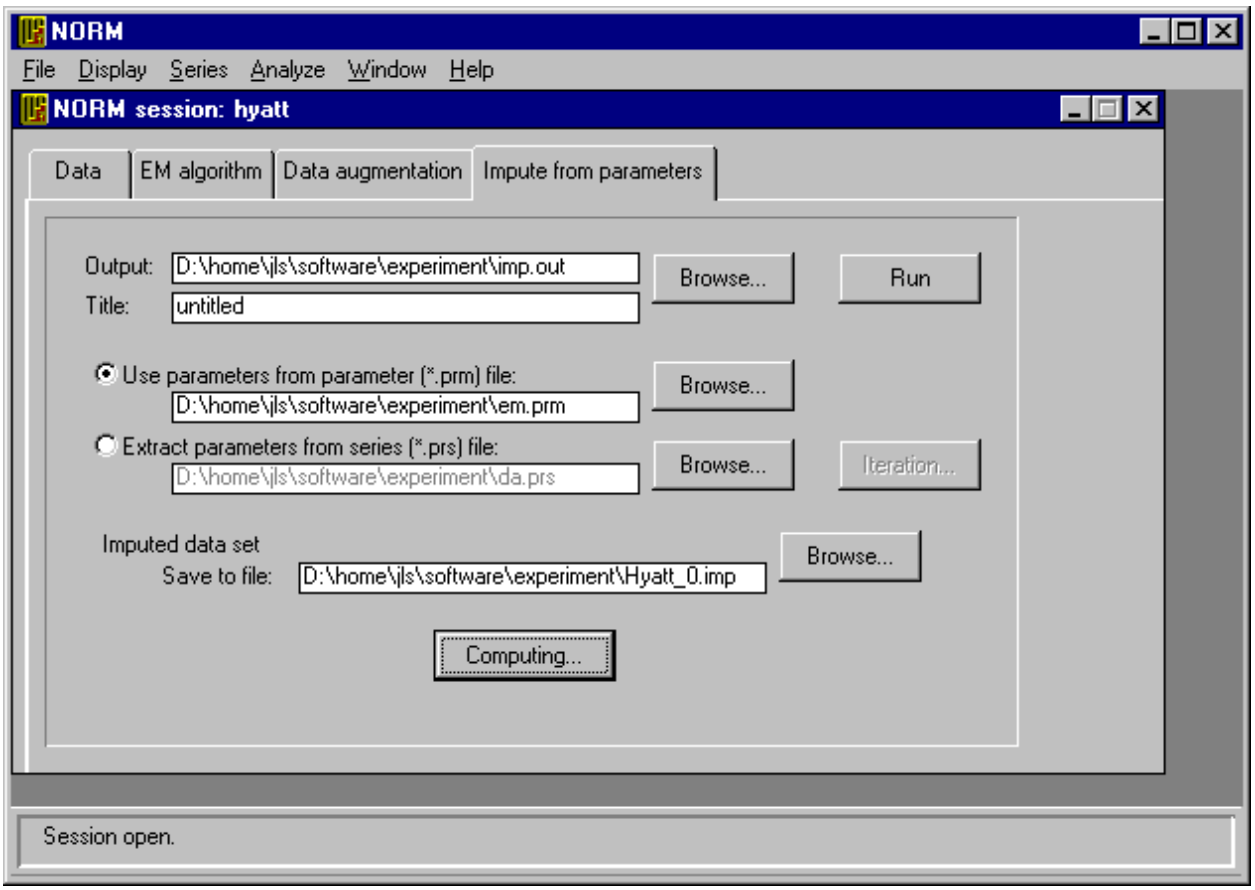
Example. The variable shown below represents grade-point averages for a sample of college students. Note that grade-point averages cannot exceed 4.0. If this variable had been imputed without transformation, some of the imputed values could have fallen above 4.0. To prevent this from happening, a logit transformation was applied with an upper endpoint of 4.005, and the data were rounded to the nearest hundredth of a point.



Impute from parameters sheet

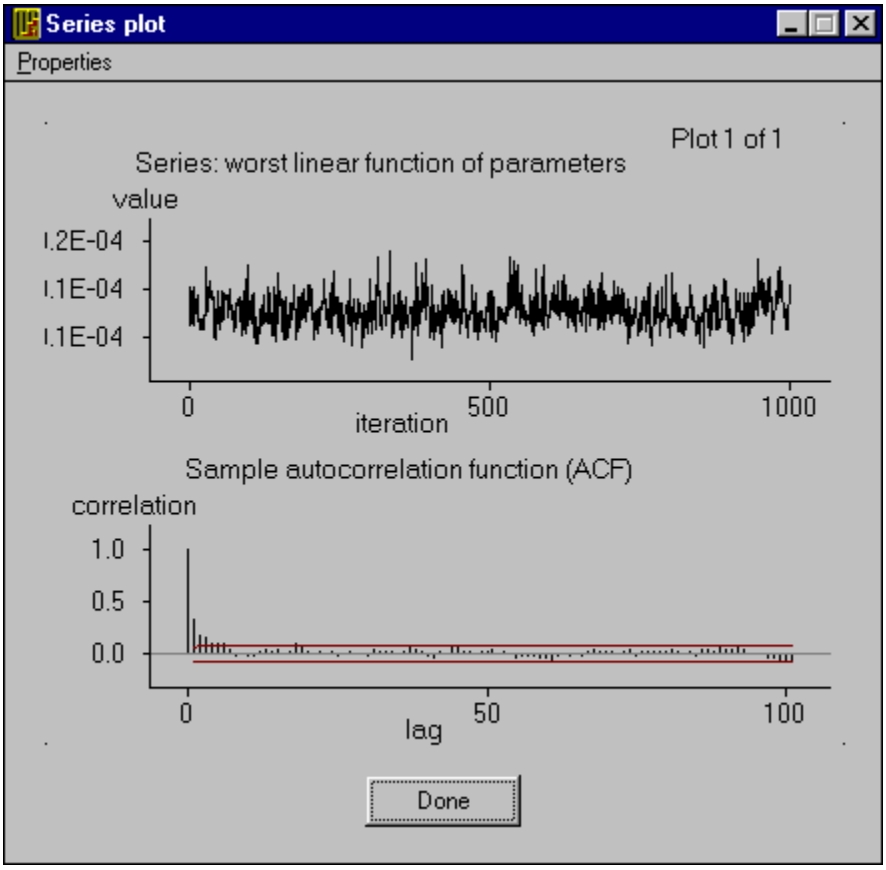
The Impute from parameters sheet can be reached by clicking on the “Impute from parameters” tab in the NORM session. Pressing the “Run” button will create one random imputation under a set of parameters obtained from a NORM parameter (*.prm) file or a NORM parameter series (*.prs) file.

The random number generator seed can be set by pressing the “Computing…” button.



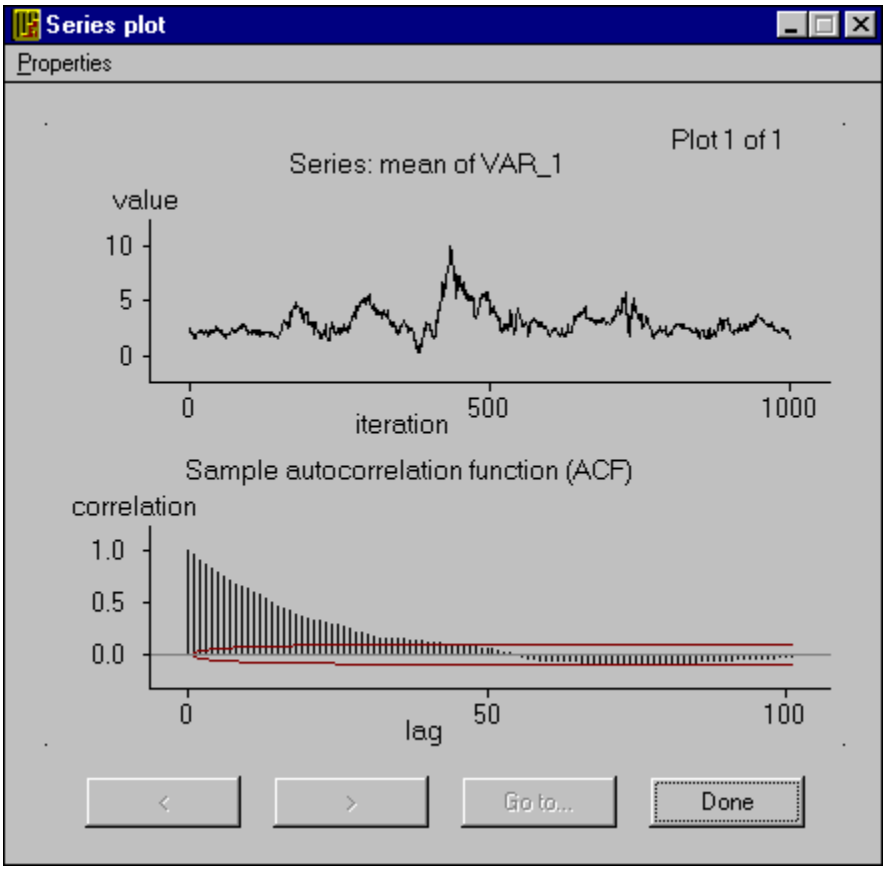
Series plot example: fast convergence

In this example, the autocorrelations essentially die down by lag 5, suggesting that data augmentation converges rather quickly.



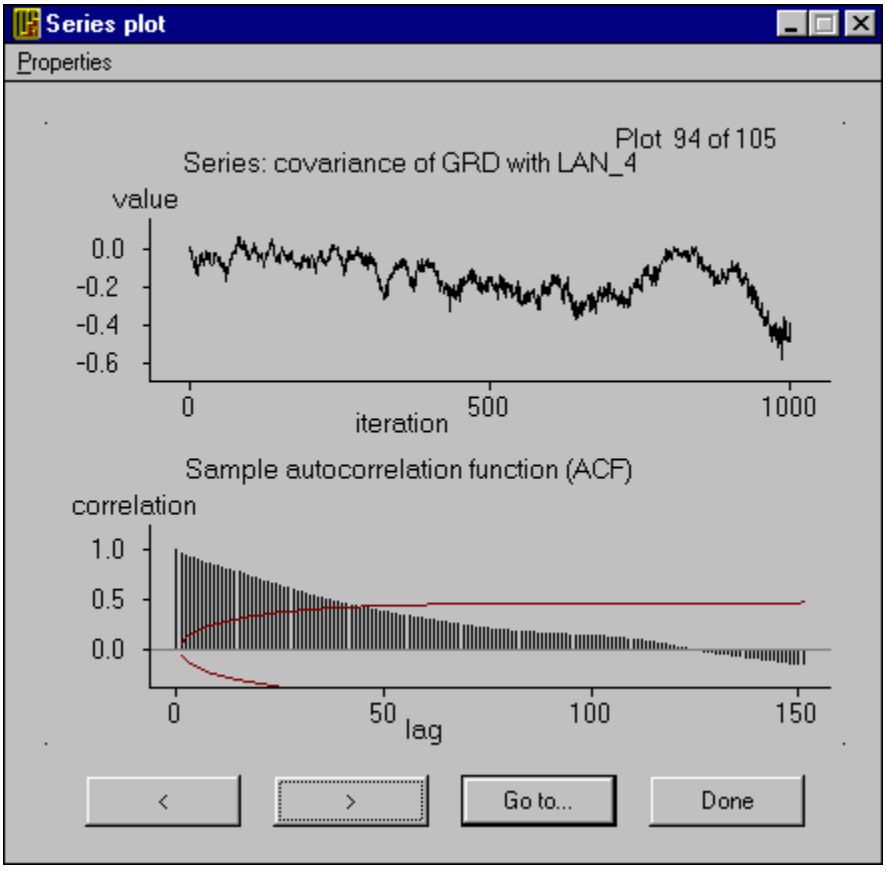
Series plot example: slow convergence

In this example, the autocorrelations do not die down until about lag 50.



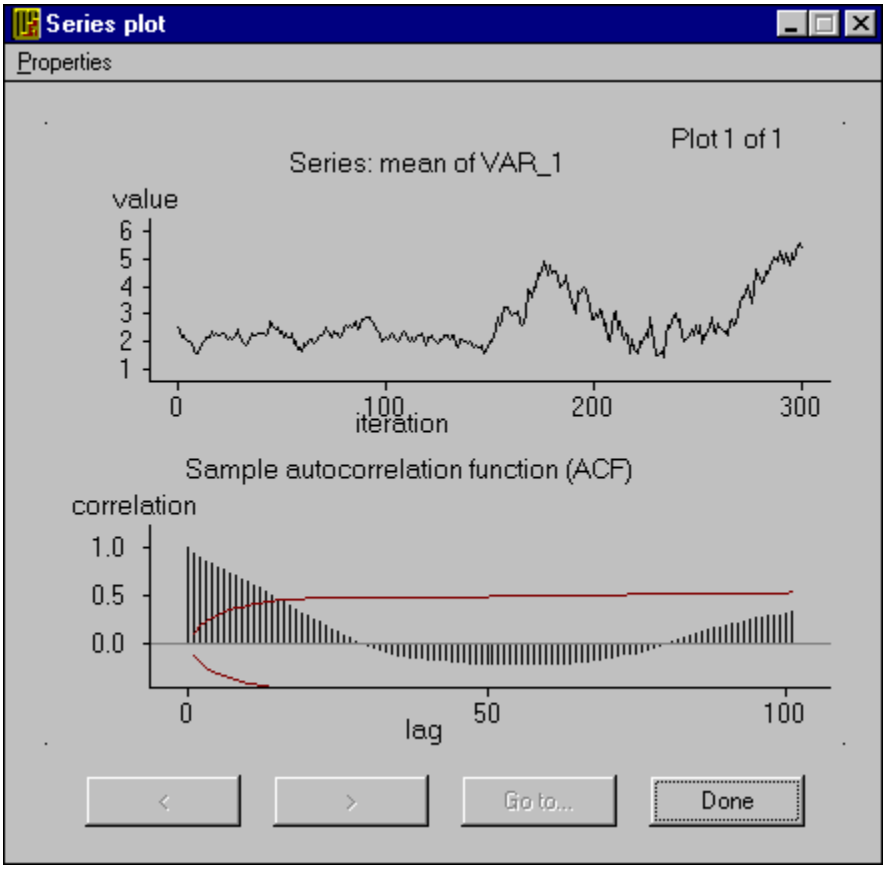
Series plot example: DA fails to converge

This example comes from a data set where some parameters are unidentified. At first glance, it may appear similar to slow convergence. But the series plot shows the parameter drifting aimlessly. Increasing the length of the series only causes it to drift more. No matter how long the series is run, the autocorrelations never seem to die down.



Series plot example: autocorrelations too noisy

In this plot, the autocorrelation function seems to oscillate, alternately drifting above and below zero. Under ordinary circumstances we would not expect autocorrelations to become negative, so this indicates that the autocorrelation function itself is poorly estimated. Increasing the length of the series stabilizes the estimate, producing the plot shown here.



Missing at random (MAR) assumption

Under missing at random (MAR), also called the ignorability assumption, the probability that any observation is missing may depend on data that are observed but not on data that are missing (Rubin, 1976; Little & Rubin, 1987).

For example, consider a simple bivariate dataset with one variable *X* that is always observed and a second variable *Y* that is sometimes missing. Under MAR, the probability that *Y* is missing for a sample subject may be related to the subject's value of *X* but not to his value of *Y*.

Some comments on this assumption:

- MAR is defined relative to the variables in a dataset. If a variable X is related both to the missingness of other variables and to the values of those variables, and X is removed from the dataset, then MAR will no longer be satisfied. For this reason, it is good practice to include in an imputation procedure variables that are predictive of missingness, because MAR then becomes more plausible.

- MAR hypothesis cannot be tested from data at hand; doing so would require knowledge of the missing values themselves.

- Data that are missing by design (e.g. in a study with planned missingness) are generally MAR.

Rubin, D.B. (1976) Inference and missing data. *Biometrika*, 63, 581-592.

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

About multiple imputation

Multiple imputation (MI) is a simulation-based approach to the analysis of incomplete data. The missing part of the data set is simulated $m > 1$ times, producing m equally plausible versions of the complete data. Each of these m versions is analyzed in an identical fashion, using standard complete-data methods. The variation among the m sets of results is a reflection of missing-data uncertainty. The results are then combined into a single set of estimates and standard errors, using Rubin's (1987) rules for scalar estimands. Joint inferences about groups of parameters are possible using the extension of Rubin's rules for multidimensional estimands.

To create multiple imputations, one must make certain assumptions about the data and the missing values. Moreover, the modeling assumptions used to create the missing values should bear some resemblance to those used in the subsequent analysis; see imputer's versus analyst's model.

Special computational methods are usually needed to create multiple imputations. One method is data augmentation, a special kind of Markov chain Monte Carlo (MCMC) technique.

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

Rubin, D.B. (1996) Multiple imputation after 18+ years*. Journal of the American Statistical Association*, 91, 473-489.

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall.

Rubin's (1987) rules for scalar estimands

Rubin (1987) presented this method for combining results from a data analysis performed *m* times, once for each of *m* imputed data sets, to obtain a single set of results. These rules are implemented in NORM's MI inference for scalar estimands procedure.

From each analysis, one must first calculate and save the estimates and standard errors. Suppose that

$\hat{Q}_j =$ estimate from data set j,

$U_j =$ squared standard error for $\hat{Q}_j$.

The overall estimate is the average of the individual estimates,

$$\overline{Q} = \frac{1}{m}\sum_{j=1}^{m}\hat{Q}_j.$$

For the overall standard error, obtain the within-imputation variance,

$$\overline{U} = \frac{1}{m}\sum_{j=1}^{m}U_j.$$

and the between-imputation variance,

$$B = \frac{1}{m-1}\sum_{j=1}^{m}\left(\hat{Q}_j - \overline{Q}\right)^2.$$

The total variance is

$$T = \overline{U} + \left(1 + \frac{1}{m}\right)B.$$

The overall standard error is the square root of *T*. Confidence intervals are obtained by taking the overall estimate plus or minus a number of standard errors, where that number is a quantile of Student's t-distribution with degrees of freedom

$$df = (m-1)\left(1 + \frac{m\overline{U}}{(m+1)B}\right)^2.$$

A significance test of the null hypothesis $Q = 0$ is performed by comparing the ratio

$$t = \overline{Q}/\sqrt{T}$$

to the same t-distribution.

See also rate of missing information.

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

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data* (London: Chapman & Hall), Chapter 4.

Rules for multiparameter inference

The method of Rubin (1987) for combining *m* sets of point estimates and standard errors has been extended to allow joint inferences about groups of parameters.

From each of the *m* imputed data sets, one must calculate and store estimates of the parameters in question along with an estimated covariance matrix. Suppose that

$\hat{Q}_j =$ vector of estimates (length = k) from data set j,

$U_j =$ covariance matrix ($k \times k$) for $\hat{Q}_j$.

The overall estimate is the average of the individual estimates,

$$\overline{Q} = \frac{1}{m}\sum_{j=1}^{m}\hat{Q}_j.$$

For the overall covariance matrix, obtain the average within-imputation covariance matrix,

$$\overline{U} = \frac{1}{m}\sum_{j=1}^{m}U_j.$$

and the between-imputation covariance matrix,

$$B = \frac{1}{m-1}\sum_{j=1}^{m}\left(\hat{Q}_j - \overline{Q}\right)\left(\hat{Q}_j - \overline{Q}\right)^{\mathsf{T}}.$$

The estimated total covariance matrix is

$$T = (1+r)\,\overline{U},$$

where

$$r = \left(1+m^{-1}\right)tr\left(B\overline{U}^{-1}\right)$$

is the average relative increase in variance due to nonresponse across the components of *Q*. The average rate of missing information is

$$\gamma = r / (r+1).$$

A test of the null hypothesis that all components of *Q* are simultaneously zero, versus the alternative that at least one component is not zero, is performed by comparing the Wald-type statistic

$$F = k^{-1}\left(\overline{Q}^{\mathsf{T}}T^{-1}\overline{Q}\right)$$

to an F-distribution with *k* numerator degrees of freedom. The denominator degrees of freedom, due to an approximation by Li, Raghunathan and Rubin (1991), are

$$4 + (t-4)\left[1+\left(1-2t^{-1}\right)r^{-1}\right]^2$$

if $t = k(m\text{-}1)$ is greater than 4, and

$$\frac{1}{2}t\left(1+k^{-1}\right)\left(1+r^{-1}\right)^2$$ otherwise.

Rate of missing information

When performing a multiply-imputed analysis, the variation in results across the imputed data sets reflects statistical uncertainty due to missing data. Rubin's (1987) rules for scalar estimands provide some diagnostic measures that indicate how strongly the quantity being estimated is influenced by missing data. The estimated rate of missing information is

$$\gamma = \frac{r + 2 / (df + 3)}{r + 1},$$

where

$$r = \frac{\left(1 + m^{-1}\right) B}{\overline{U}}$$

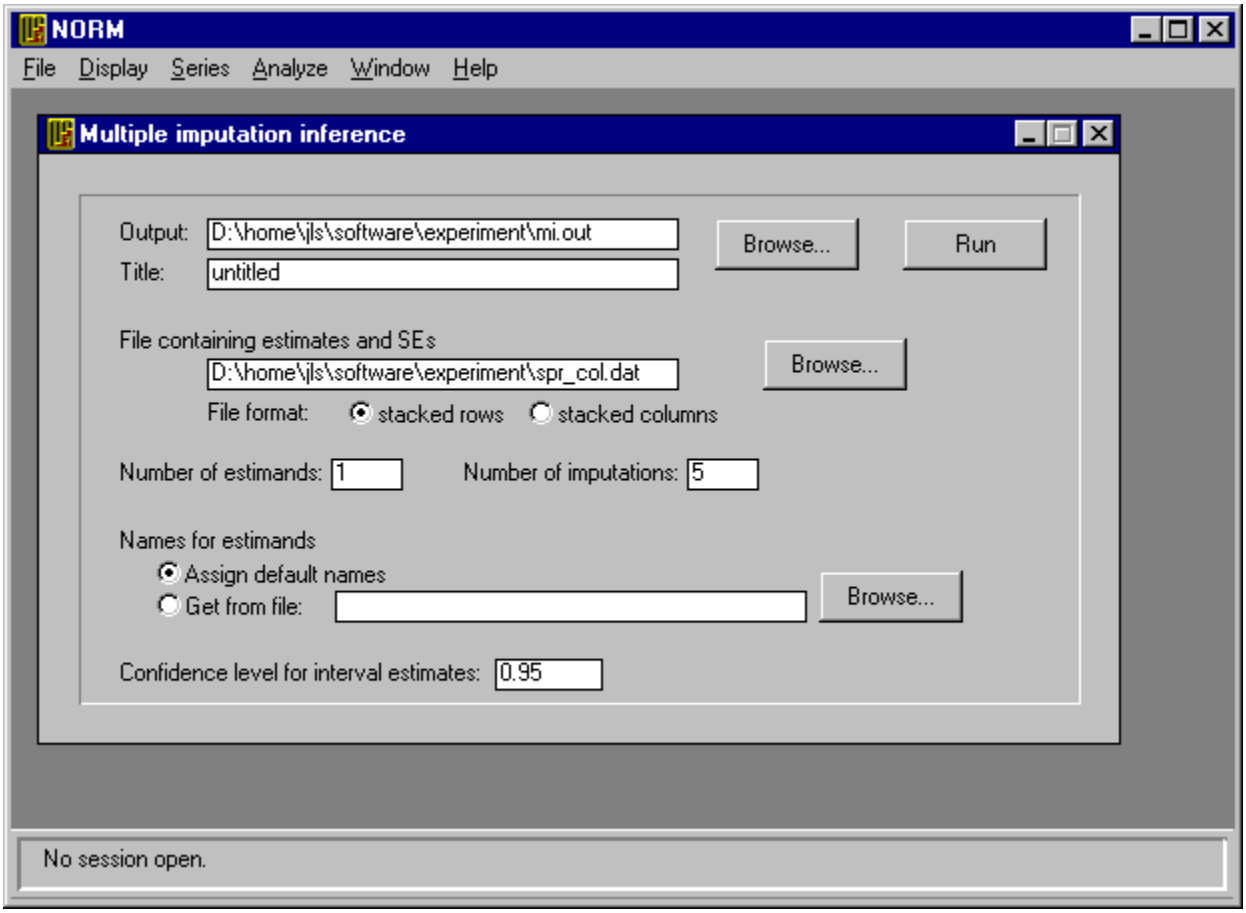
is the relative increase in variance due to nonresponse.

The rate of missing information, together with the number of imputations *m*, determines the relative efficiency of the MI inference; see How many imputations do I need?

Note. This estimate of the rate of missing information can be quite noisy, particularly when the number of imputations *m* is not large; it should be interpreted only as a rough guide. For example, if the estimated rate based on 5 imputations is 25%, it may actually be anywhere from about 20% to 30%.

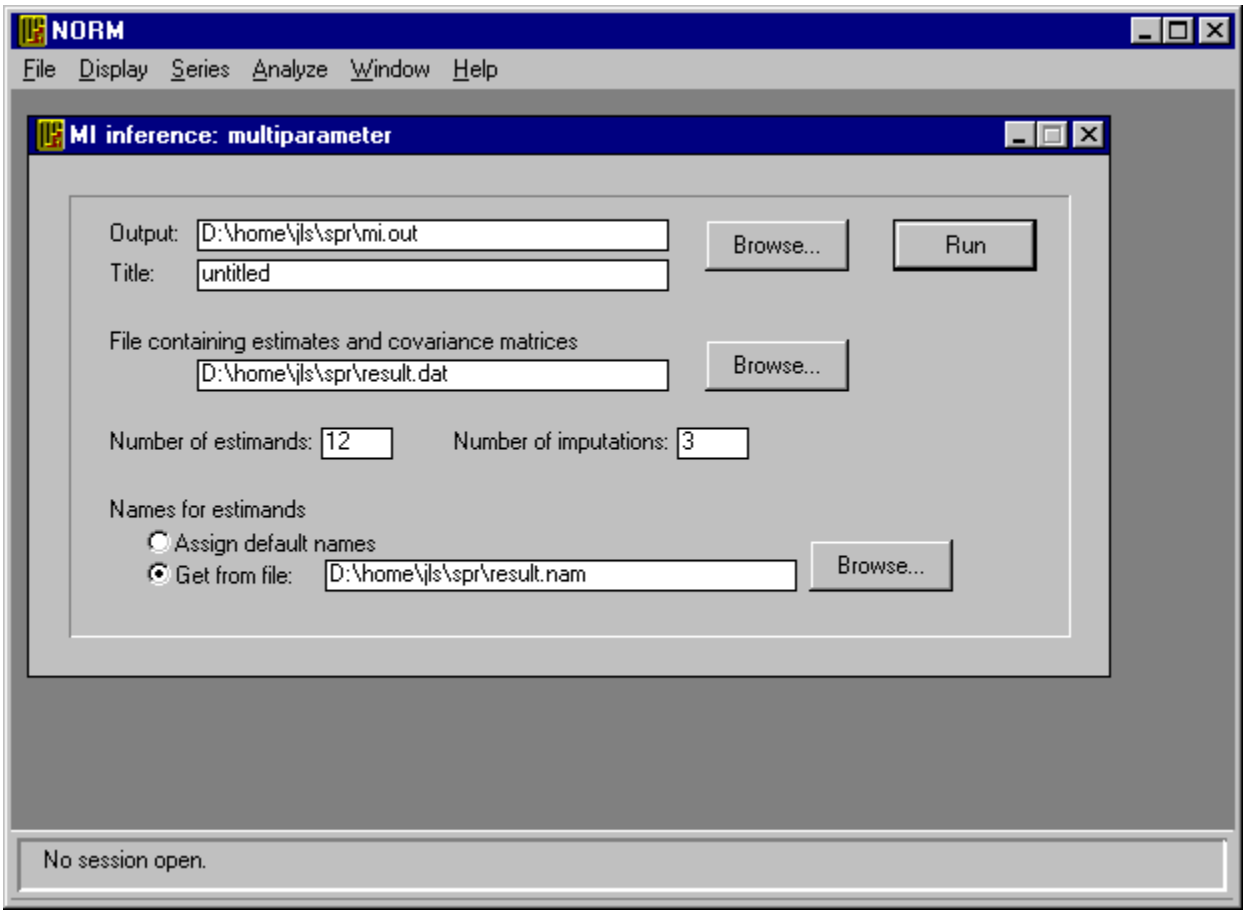
MI scalar inference session window

This window allows you to set options for the MI scalar inference utility.



MI multiparameter inference session window

This window allows you to set options for the MI multiparameter inference utility.



Stacked matrix format

In stacked matrix format, estimates and covariance matrices from the *m* analyses are arranged like this:

[Analysis 1:]

$$\begin{matrix} est_1 & est_2 & \cdots & est_k \\ var_1 & cov_{12} & \cdots & cov_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ cov_{k1} & cov_{k2} & \cdots & var_k \end{matrix}$$

[Analysis 2:]

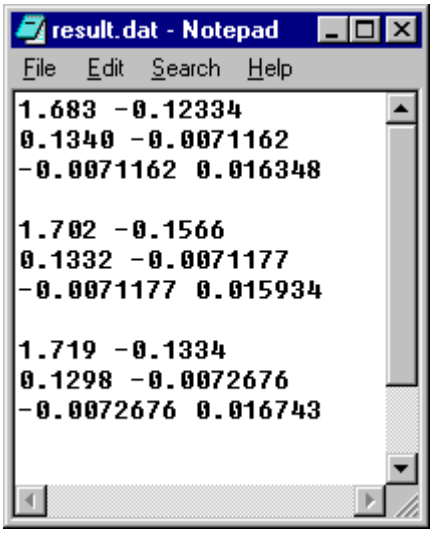
$$\begin{matrix} est_1 & est_2 & \cdots & est_k \\ var_1 & cov_{12} & \cdots & cov_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ cov_{k1} & cov_{k2} & \cdots & var_k \end{matrix}$$

$$\vdots$$

[Analysis *m*:]

$$\begin{matrix} est_1 & est_2 & \cdots & est_k \\ var_1 & cov_{12} & \cdots & cov_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ cov_{k1} & cov_{k2} & \cdots & var_k \end{matrix}$$

For example, here is a stacked matrix file for *k* = 2 estimands from *m* = 3 analyses:



Please note the following.

- The numbers of each line may be separated by any number of blank spaces or tab characters.
- The blank lines separating the results from the *m* analyses are merely a cosmetic device. Blank lines in the file are ignored.

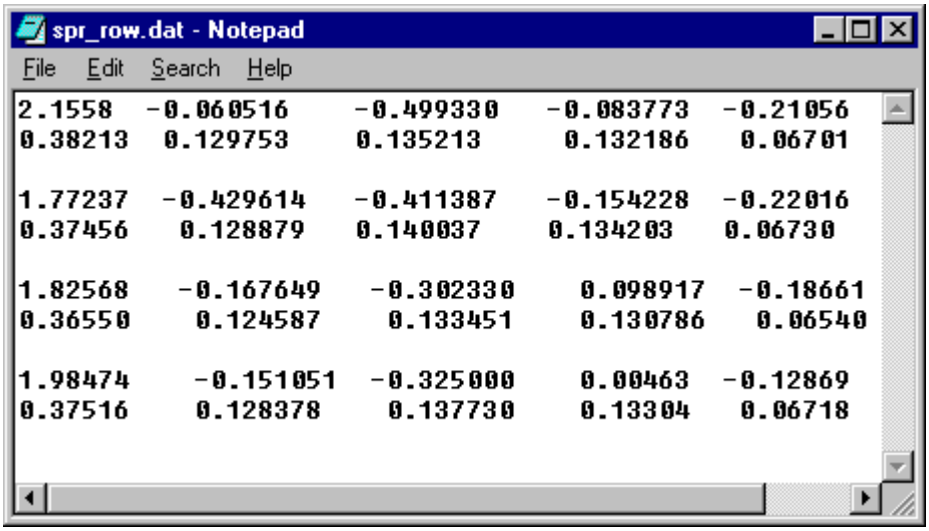
- Keep the *k* estimates on a single line, and each row of a covariance matrix on a single line. Do not wrap them around to the next line, even if *k* is large. Very long lines (up to 2 kilobytes each) are not a problem.

Stacked row file format

In stacked row format, estimates and standard errors from the *m* analyses are arranged like this:

[Analysis 1:]
$est_1 \quad est_2 \quad \cdots \quad est_k$
$SE_1 \quad SE_2 \quad \cdots \quad SE_k$
[Analysis 2:]
$est_1 \quad est_2 \quad \cdots \quad est_k$
$SE_1 \quad SE_2 \quad \cdots \quad SE_k$
$\vdots$
[Analysis *m*:]
$est_1 \quad est_2 \quad \cdots \quad est_k$
$SE_1 \quad SE_2 \quad \cdots \quad SE_k$

For example, here is a stacked row file for *k* = 5 estimands from *m* = 4 analyses:



Please note the following.

- The numbers of each line may be separated by any number of blank spaces or tab characters.
- The blank lines separating the results from the *m* analyses are merely a cosmetic device. Blank lines in the file are ignored.
- Keep the *k* estimates or standard errors from any one analysis on a single line. Do not wrap them around to the next line, even if *k* is large. Very long lines (up to 2 kilobytes each) are not a problem.

Stacked column file format

In stacked column format, estimates and standard errors from the *m* analyses are arranged like this:

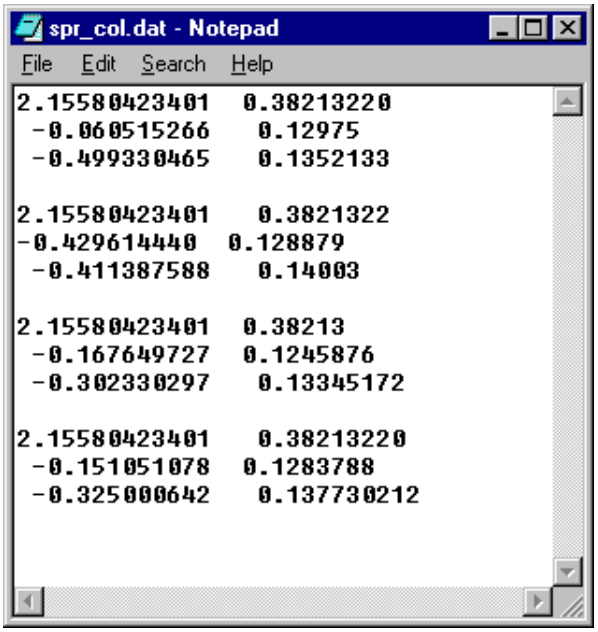
[Analysis 1:]
est_1 SE_1
est_2 SE_2
⋮ ⋮
est_k SE_k

[Analysis 2:]
est_1 SE_1
est_2 SE_2
⋮ ⋮
est_k SE_k

⋮

[Analysis *m*:]
est_1 SE_1
est_2 SE_2
⋮ ⋮
est_k SE_k

For example, here is a stacked column file for *k* = 3 estimands from *m* = 4 analyses:



Please note the following.

- The numbers of each line may be separated by any number of blank spaces or tab characters.
- The blank lines separating the results from the *m* analyses are merely a cosmetic device. Blank lines in the file are ignored.

Names file for MI inference

A names file for MI inference is an ordinary text (ASCII) file. It will typically have the same name as the file containing estimates and standard errors (or covariances), except that it will end with *.nam rather than *.dat. The name for each estimand should appear on a separate line. For example, suppose that you want to combine results for a set of $k = 4$ regression coefficients obtained from $m = 5$ imputed data sets. A names file for this situation might look like this:

Each name may be up to 10 characters long. Names longer than 10 characters will be truncated.