Multiple Imputation for Multilevel Models with Missing Data Using Stat-JR

Introduction

In this document we introduce a Stat-JR super-template for 2-level data that allows for missing values in explanatory or response variables, and can handle normal or categorical variables. This covers similar content to the DEEP eBook accompanying the template (the eBook further allows you to enter your inputs as you proceed through worked examples).

Stat-JR *super*-templates are so-called because they call other templates; the super-template documented here, **2LevelImpute**, allows the user separately to specify the imputation and model of interest (MOI). The multiply-imputed datasets are then produced, and the MOI fitted to each (via a variety of other Stat-JR templates 2LevelImpute calls); these are combined using Rubin's rules, and the results for the final MOI fit are then returned (together with results from a complete case analysis, for comparison). If your computer has multiple processors, these will be used in parallel for the imputation models.

It will be assumed that you have a knowledge of the basics of methods for missing data using a joint modelling approach. If not, you might like to look at the above reference and for a general background overview the missing data web site http://missingdata.lshtm.ac.uk/ that has references to papers and recent developments.

The template we will be using incorporates the existing REALCOM procedures, but provides a very much faster implementation.

Authors of templates:	Core team templates written by Chris Charlton and William Browne
Authors of documentation:	Harvey Goldstein and Richard Parker

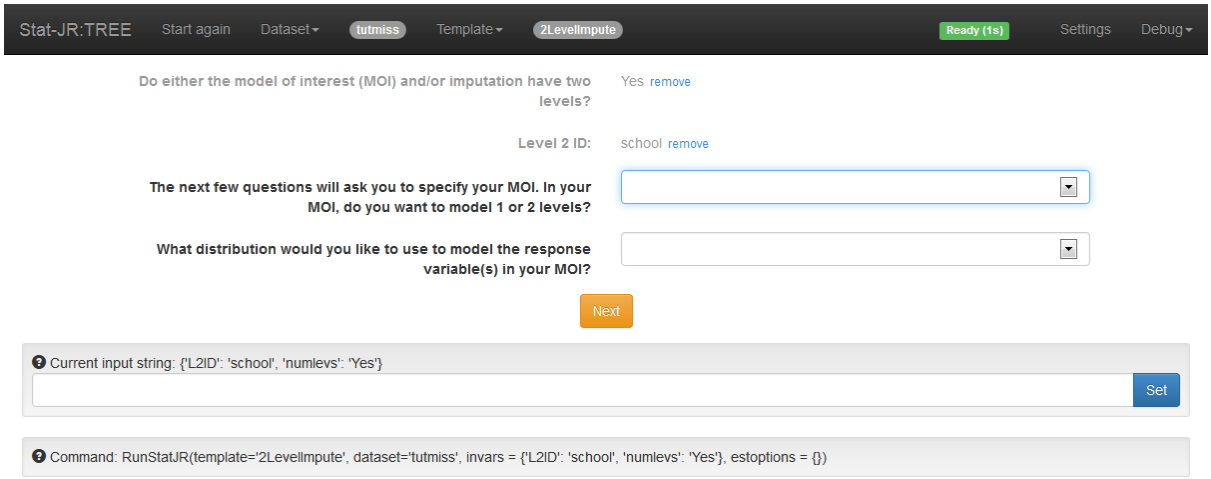
Below, we run through the following:

1. Using the TREE interface
2. Overview of inputs
3. Worked example

Using the TREE interface

For details of how to install Stat-JR see http://www.bristol.ac.uk/cmm/software/statjr/order-statjr/

Below we have a screenshot from TREE, in which we have selected the template **2LevelImpute** and the dataset **tutmiss** (as used in this example), and have started to specify our inputs. For details of how to select templates and datasets when using Stat-JR's TREE interface, see A Beginner's Guide to Stat-JR's TREE interface (http://www.bristol.ac.uk/cmm/media/migrated/1-0-1/manual-tree-beginners.pdf).



You can, of course, upload your own dataset in TREE as well. If it is already saved as a .dta file, then you can do so via (i) **Dataset** > **Upload**, which will upload it into the temporary memory cache, or by (ii) saving your dataset in the StatJR/datasets folder, and then selecting **Debug** > **Reload** datasets (see top-right of the screen, in the black bar). If, instead, you have it (iii) saved as a .txt file, you can use Stat-JR's **LoadTextFile** template to save it into the temporary memory cache. In the case of option (i) and (iii) the dataset will be available for use in the current session, but you then need to download it (as a .dta file) via **Dataset** > **Download** (e.g. saving it into the StatJR/datasets folder) for use in the future sessions too.

In the screenshot below, we have specified all the inputs, and after pressing the **Next** button a final time the **Run** button will appear, along with some other outputs in the pane at the bottom of the browser window (not shown here).

Stat-JR:TREE | Start again | Dataset | tutmissOnePass | Template | 2LevelImpute | Ready (1s) | Settings | Debug

Do either the model of interest (MOI) and/or imputation have two levels?	Yes remove
Level 2 ID:	school remove
The next few questions will ask you to specify your MOI. In your MOI, do you want to model 1 or 2 levels?	2 levels remove
What distribution would you like to use to model the response variable(s) in your MOI?	Normal remove
In your MOI, do you want to fit random slope/coefficient(s)?	No remove
In your MOI, what is the response variable?	normexam remove
Explanatory variables for MOI:	cons,standlrtmiss,girlmiss,schgendmiss:cat remove
The next few questions will now ask about your IMPUTATION MODEL. What responses are at level 1?	normexam,standlrtmiss,girlmiss remove
Response Type for response normexam:	Normal remove
Response Type for response standlrtmiss:	Normal remove
Response Type for response girlmiss:	Binary remove
Explanatory variables for normexam in imputation model:	cons remove
Explanatory variables for standlrtmiss in imputation model:	cons,vrband:cat remove
Explanatory variables for girlmiss in imputation model:	cons remove
Are there any responses at level 2 for imputation model?:	Yes remove
Responses at level 2 for imputation model:	schgendmiss remove
Response Type for response schgendmiss:	Unordered remove
Number of categories for schgendmiss:	3 remove
Explanatory variables for category 0 at level 2 for imputation model:	cons remove
Explanatory variables for category 1 at level 2 for imputation model:	cons remove
Use conditional marginal algorithm?	Yes remove
Number of imputed data sets:	4 remove
Number of iterations before first imputation:	1000 remove
Number of iterations between subsequent imputations:	50 remove
Length of burnin for MOI:	100 remove
Number of iterations for MOI:	250

Next

Current input string: {'want2': 'Yes', 'MOIlevel': '2 levels', 'bin1_3': 'Binary', 'bin1_2': 'Normal', 'bin1_1': 'Normal', 'imputefirst': '1000', 'yimp2': 'schgendmiss', 'numlevs': 'Yes', 'MOIslope': 'No', 'condmarg': 'Yes', 'L2ID': 'school', 'burnin': '100', 'yimp1': 'normexam,standlrtmiss,girlmiss', 'numimpute': '4', 'imputeevery': '50', 'bin2_1': 'Unordered', 'ximp1_1': 'cons,vrband:cat', 'ximp1_0': 'cons', 'ximp1_2': 'cons', 'MOIdist': 'Normal', 'y': 'normexam', 'x': 'cons,standlrtmiss,girlmiss,schgendmiss:cat', 'ncatsschgendmiss': '3', 'ximp2_0_1': 'cons', 'ximp2_0_0': 'cons'}

ıs', 'ximp2_0_0': 'cons'} Set

Command: RunStatJR(template='2LevelImpute', dataset='tutmissOnePass', invars = {'want2': 'Yes', 'bin1_3': 'Binary', 'bin1_2': 'Normal', 'bin1_1': 'Normal', 'numlevs': 'Yes', 'MOIslope': 'No', 'condmarg': 'Yes', 'x': 'cons,standlrtmiss,girlmiss,schgendmiss:cat', 'burnin': '100', 'ximp1_1': 'cons,vrband:cat', 'ximp1_0': 'cons', 'ximp1_2': 'cons', 'ximp2_0_1': 'cons', 'ximp2_0_0': 'cons', 'imputefirst': '1000', 'L2ID': 'school', 'yimp2': 'schgendmiss', 'yimp1': 'normexam,standlrtmiss,girlmiss', 'numimpute': '4', 'imputeevery': '50', 'bin2_1': 'Unordered', 'MOIdist': 'Normal', 'y': 'normexam', 'MOIlevel': '2 levels', 'ncatsschgendmiss': '3'}, estoptions = {})

Edit | Popout

Overview of inputs

Here we give a brief overview of the inputs Stat-JR requires in order to run the 2LevelImpute super-template:

Multilevel or not?

- You are first asked if either the model of interest (MOI) and/or imputation have two levels (or just one). In general, you would want to fit the same number of levels in your imputation model as in the MOI, but there may be some situations where for simplicity, where the level 2 random effects are small, the MOI might be a single level model, but you would still wish to fit a 2-level imputation model. In such a case you will still need to specify the level 2 ID.

About your MOI

- You are then asked a few questions about the structure of your MOI, including whether it is a 2-level model or not (see note, above), the distribution you would like to use to model the response variable, whether you would like to fit a random slope (or coefficient) model or not (if applicable), and your response and explanatory variables.

About your imputation model

- After that, you are prompted for the level 1 variables to be used as responses in the imputation model, and to specify their distribution and explanatory variables. Then, if applicable, you are asked about the variables at level 2 in the imputation model as well.
- Typically these response variables are any that have missing values. Those variables without missing values, if they are to be used in the MOI, can be declared as either response *or* explanatory variables in the imputation model. In addition you may wish to include 'auxiliary' variables (not in the MOI) in the imputation model if these are associated with the propensity to be missing.
- You may have a different set of explanatory variables for each response. This may be useful where you wish to have auxiliary variables relevant to certain responses only.

Estimation options

- Finally, you are asked various questions about the estimation procedures, including the number of imputed datasets to use, the interval between iterations (to ensure approximate independence) at which to impute a complete dataset and, for the MOI, the burn in and number of iterations.

Note that all variables are stored as vectors of the same length as those in the full data set. Where these are declared as level 2 variables the first one in each level 2 unit is chosen. In fact the template checks to determine whether such values are actually constant within each level 2 unit and the user will be notified if not.

Worked example

Dataset

In the following example we will consider the tutorial dataset that has been used many times as an example of a 2-level educational dataset. See the MLwiN manual, for example.[1]

The dataset consists of a sample of records of school achievement for 4059 pupils within 65 schools - some missing values have been randomly introduced.

The dataset (saved as **tutmiss**) is summarised in the table below.

Column name	N	Missing	Min	Max	Description
school	4059	0	1	65	Numeric school identifier
student	4059	0	1	198	Numeric student identifier
normexam	4059	0	-3.67	3.67	Students' exam score at age 16, normalised to have approximately a standard Normal distribution.
cons	4059	0	1	1	A column of ones. If included as an explanatory variable in a regression model, its coefficient is the intercept.
standlrt	4059	0	-2.93	3.02	Students' score at age 11 on the London Reading Test (LRT), standardised using Z-scores.
girl	4059	0	0	1	Students' gender: 0=boy; 1=girl
Schgend	4059	0	1	3	School gender: 1=mixed; 2=boys' school; 3=girls' school
Avslrt	4059	0	-0.76	0.64	Average LRT score in school
Schav	4059	0	1	3	Average LRT score in school, coded into 3 categories: 1=bottom 25%; 2=middle 50%; 3=top 25%
Vrband	4059	0	1	3	Students' score in test of verbal reasoning at age 11, coded into 3 categories: 1=top 25%; 2=middle 50%; 3=bottom 25%
schgendmiss	4059	439	0	2	As schgend, but with missing values randomly introduced.
Avslrtmiss	4059	431	-0.76	0.64	As avslrt, but with missing values randomly introduced.
standlrtmiss	4059	400	-2.93	3.02	As standlrt, but with missing values randomly introduced.
Girlmiss	4059	435	0	1	As girl, but with missing values randomly introduced.

Working through the inputs

In this guided example, using the dataset described above, we will be fitting a 2-level model.

1. Specifying the number of levels, and structure of MOI

- First you will be asked if *either* the MOI or imputation model has two levels; we're going to fit a 2-level MOI, so will answer **Yes**.
- Note that normally you will wish to have the same number of levels or classifications for the MOI and the imputation model. In some situations, however, for example if level 2 effects are very small, you may want to fit a 1 level MOI for simplicity, while still carrying out

[1] Rasbash, J., Steele, F., Browne, W.J. and Goldstein, H. (2012). A user's guide to MLwiN Version 2.27. Centre for Multilevel Modelling, University of Bristol.

imputation at 2 levels. If either the MOI or the imputation model is at 2 levels, you will be asked for the level 2 ID.

- You will then be asked to enter the level 2 ID, which in this example is **school**; this will be used in the MOI and/or imputation model, as appropriate.

- Next, you will be asked a series of questions about your MOI model: whether you want to model 1 or 2 levels (we're modelling **2 levels** in this example), the distribution you would like to use to model the response variable in your MOI (**Normal**, in our example), and finally whether you would like to fit random slopes / coefficients or not (we'll answer **No**; note, as indicated in the hover-over help available for this input, if you would like to fit a random slope/coefficicent(s) model, make sure that you include the response(s) in your MOI, AND the variables whose coefficients you wish to randomly-vary at level 2, as responses in the imputation model).

- The **2LevelImpute** template takes this input, and uses it to choose the best Stat-JR template to fit your MOI. If you make the choices suggested above, it will choose a template called **2LevelMod**, which fits random intercept models for Normal, binomial and Poisson responses.

2. Specifying the variables in the MOI

You will then be asked for the response and explanatory variables for the MOI. In this worked example we suggest a simple 2 level variance components model with:

- **normexam** as the response;

- **cons**, **schgendmiss**, **standlrtmiss** and **girlmiss** as explanatory variables (as you click on these you'll see that they appear in the box beneath; to deselect any chosen in error, simply click on the variable name in the lower box); remember to specify that **schgendmiss** is categorical (you can do so by using the checkboxes which appear below the input box when you select your explanatory variables).

3. Specifying the imputation model

Once you have specified the MOI, you will then, for each level, be asked to enter the responses for the imputation model; we'll work through each of the two levels in the sections below.

- Note that all explanatory variables *must* have no missing values - if any variable does, then include it as a response.

Level 1

- Given our MOI, we suggest the variables **normexam**, **standlrtmiss** and **girlmiss** for level 1; you'll then need to specify their distributions, here as **Normal**, **Normal**, and **Binary**, respectively.

- You will also be asked to enter the explanatory (predictor) variables for each of these as follows:
    - for **normexam** just use an intercept **cons**;
    - for **standlrtmiss** use **cons** and also **vrband** (making sure that you tick the box that asks whether **vrband** should be treated as categorical - this will then use appropriate dummy variables where the final category is taken as the reference) – see note below;
    - for **girlmiss** use **cons**.
- Since we have selected values to be missing purely at random the use of a second predictor for **standlrtmiss** is not necessary, but you can include it to demonstrate that, as an auxiliary variable (not in the MOI) it can be used to help ensure missingness at random (MAR).
- Finally, you'll be asked if there are any responses at level 2 for the imputation model; we'll answer **Yes**.

Level 2

- Next, you'll be asked to enter the responses for the imputation model at this level. We suggest using the variable **schgendmiss** which is coded 0 for mixed schools, 1 for boy's school and 2 for girl's school: i.e. when asked, indicate that it has an **Unordered** distribution, with **3** categories (note, as indicated in the hover-over help available for this input, your categories, as represented in your dataset, need to be numbered from zero, sequentially in steps of one (i.e. 0,1,2 if you had 3 categories); if they are not, an error message will be returned).
- You will then be asked for explanatory variables for each category dummy - we suggest you use **cons** for each.
- Note: If we have a response at level 2, it is not meaningful, in general, to have explanatory variables at level 1. So you should not specify these for any level 2 responses in the imputation model. You may, of course, specify level 2 explanatory variables for level 1 responses in the imputation model: this may be particularly useful for auxiliary variables.

A note about the latent normal model

Multiple imputation theory strictly applies only where the set of (response plus explanatory) variables for which we wish to impute missing values have a joint multivariate normal distribution for the variables all treated as responses. Where we have categorical data this is clearly not the case and we therefore use a latent normal formulation (See: Goldstein, H., Carpenter, J., Kenward, M. and Levin, K. (2009). Multilevel models with multivariate mixed response types. Statistical Modelling, 9(3), 173-197). This works as follows:

- For a binary response we utilise a probit model where we assume that the (0,1) response is derived from an underlying standard normal distribution with a mean value (determined by whatever explanatory variable predictors are in the imputation model for this response) that acts as a threshold - values above which are observed as a '1' and below which as a '0'. The MCMC algorithm incorporates a step that takes a random draw from the corresponding part of the standard normal distribution according to whether a '1' or '0' is observed, or imputes randomly if a value is missing.

- For an ordered categorical response a similar procedure is used with additionally a set of thresholds defined on the standard normal scale that delineate the ordered categories.

- For an unordered categorical variable with p categories a (p-1) dimensional multivariate normal distribution is generated.

- For each of these the underlying normal distributions condition on the other responses (and explanatory variables) so that a joint multivariate normal is finally generated.

4. Specifying the number of iterations, etc., and fitting the model

- Next, you will be asked if you want to use the conditional algorithm or not; we suggest answering **Yes**, since the conditional algorithm is faster.

- You will then be asked to specify the number of imputed datasets to use, the interval before the first imputation (this includes any burn-in period), the interval before any subsequent imputations (to ensure approximate independence) and, for the MOI, the burn in and number of iterations. Obviously the numbers you enter here will depend on the characteristics of your data, etc.

- After clicking on the last **Next** button, and then the **Run** button if using the template via the Stat-JR:TREE interface. The template will then run (the progress gauge in the black bar will change from *Ready* to *Working* to indicate it is busy).

If your computer has more than one core processor, the imputation models and associated MOIs will in fact be run with parallel chains.

Inspecting the results

Once it has finished, and all the results have been returned, you can view the outputted files in the pane towards the bottom of the browser window, or press 'Popout' to view the selected output in another browser tab (see below for details of the various outputs returned).

For example, to view the **imputed datasets**, choose *"Imputation_Model_impute_datafile_chain0_iter1000"*, *"Imputation_Model_impute_datafile_chain1_iter1000"*, etc. Note that the nomenclature here will

depend on both the inputs you have chosen, and the number of processors on your machine. For example, if I chose:

- **Number of imputed data sets: 5**
- **Number of iterations before first imputation: 1000**
- **Number of iterations between subsequent imputations: 500**

on my machine with four processors, then I would see:

- *"...chain0_iter1000"*
- *"...chain1_iter1000"*
- *"...chain2_iter1000"*
- *"...chain3_iter1000"*
- *"...chain0_iter1500"*
- *"...chain1_iter1500"*
- *"...chain2_iter1500"*
- *"...chain3_iter1500"*

Here it has ran a chain / thread on each of the four available processors, and derived a dataset from each after the 1000th iteration (as stated earlier, this interval includes the burn-in). However, since this number is less than the requested **5** imputed datasets I asked for, it has also constructed four more datasets following a further 500 iterations (since I specified **Number of iterations between subsequent imputations: 500**). However, it will only use the first of these (*"...chain0_iter1500"*) to make up the five datasets it needs.

The imputed datasets are available in the list of datasets accessible via **Dataset > Choose** in the black bar at the top. You can download these (as Stata format **.dta* files) by first selecting the relevant dataset from the list, pressing the neighbouring **Use** button, and then downloading via **Dataset > Download** (again via the black bar at the top; note that both *Imputation_Model_impute_datafile_chain0_iter1000*, etc. and *impute_datafile_chain0_iter1000*, etc. will appear in this list, but there's no need to download both (they're the same, but simply saved twice, with and without the Imputation_Model prefix; also, don't confuse these with the level 2 datasets (e.g. *impute__L2Data_chain0_iter1000* or *Imputation_Model_impute__L2Data_chain0_iter1500*).

Alternatively, one can press the green **Download** button to download all these outputted files (now supported for the **2LevelImpute** template from Stat-JR version 1.0.2). Note that this may take some time, as Stat-JR prepares all the outputs for downloading; you may see a flurry of activity in the corresponding command prompt window as it does so. Note also that the downloaded dataset files lack the .dta file extension, so this will need to be added manually.

What is returned in the results pane?

ResultsTable:Imputation

Estimates generated using Rubin's rules

ResultsTable:CompleteCases

Estimates from a complete cases model which is simply ran for comparison. Since, in the worked example above, the data were set missing at random we would not expect the estimates to differ much from the estimates returned in **ResultsTable:Imputation**, although the standard errors *tend* to be increased for the complete case analysis where 30% of the records have at least one variable in the MOI missing and have been deleted.

Imputation_Model_ModelParameters; Imputation_Model_ModelResults

For this template these outputs return the same information (because there's no DIC)

Objects with prefix CompleteCasesModel

Everything with prefix **CompleteCasesModel** refers to the complete cases model which is simply ran for comparison. Most of these files won't be of much interest, although estimates from the complete cases model can be found in **CompleteCasesModel_ModelParameters**, **CompleteCasesModel_ModelFit** and **CompleteCasesModel_ModelResults**.

Imputation_Model_impute_datafile_chainA_iterB

Imputed level 1 datasets. As discussed above, the nomenclature here (i.e. for 'A' and 'B') will depend on both the inputs you have chosen, and the number of processors on your machine.

Imputation_Model_impute__L2Data_chainA_iterB

As above, but at level 2.

Imputation_Model_out

Chain dataset for the imputation model, length of which will depend on the estimation options chosen (replicated as **out** in the list of datasets).

CombinedResults

Chains for each imputation model, we 'pretend' that each MOI from each imputed dataset is a chain from a multiple chain model, which allows us to combine them to derive diagnostic graphs.

Objects with the prefix Model1, Model2, etc.

Everything with the prefix **Model1**, **Model2**, etc. (up to the number of imputed datasets requested), relates to the fitted MOI models (they are the outputs from the Stat-JR template called by 2LevelImpute when it fits the MOI towards the end of the execution).

**.svg*

MCMC diagnostic plots. The top-left graph shows the values plotted against iteration number, and is useful to confirm that the chain is mixing well, meaning that it visits most of the posterior distribution in few iterations. The top-right graph contains a kernel density plot representing the posterior distribution for this parameter. The two graphs in the middle row are time series plots known as the autocorrelation (ACF) and partial autocorrelation (PACF) functions. The ACF indicates the level of correlation within the chain; this is calculated by moving the chain by a number of iterations (called the lag) and looking at the correlation between this shifted chain and the original. The PACF picks up the degree of auto-regression in the chain. By definition a Markov chain should act like an autoregressive process of order 1, as the Markov definition is that the future state of the chain is independent of all the past states of the chain given the current value. If, for example, in

reality the chain had additional dependence on the past 2 values, then we would see a significant PACF at lag 2. The bottom-left plot is the estimated Monte Carlo standard error (MCSE) plot for the posterior estimate of the mean. As MCMC is a simulation-based approach this induces (Monte Carlo) uncertainty due to the random numbers it uses. This uncertainty reduces with more iterations, and is measured by the MCSE, and so this graph details how long the chain needs to be run to achieve a specific MCSE. The sixth (bottom-right) plot is a multiple chains diagnostic: a Brooks-Gelman-Rubin diagnostic plot (BGRD; Brooks and Gelman, 1998). This plot looks at mixing across the chains: the green and blue lines measure variability between and within the chains, and the red is their ratio. For good convergence the red line should be close to 1.0.

Note that the diagnostics the 2LevelImpute template automatically returns are derived in different ways: it returns separate trace plots (on the same graph) for each chain, separate kernel density plots (on same graph) for each chain, it joins together the chains for the ACF, PACF and MCSE, but treats the chains separately for the BGRD (which is a multiple chains diagnostic); it also adds together the ESS value for each chain to derive the ESS value returned in the results.

script.py
This is the internal script, written in Python, which runs the execution you have requested.

**.cpp*
C++ code fragments used to fit the model.

Inputs
A list of inputs for internal purposes.

Imputation_Model_equation.tex
Currently not implemented for this template, but in some other templates this returns a LaTeX rendering of the model equation.

Imputation_Model_algorithm.tex
Since this template executes via custom C code, this isn't terribly informative here, but in some other templates it returns the algebra for the conditional posterior distributions.

Imputation_Model_Chains
An object used for internal purposes, doesn't actually render anything if selected in the output pane.