V2. April 2017.

Imputation for Multilevel Models with Missing Data Using Stat-JR

Richard Parker and Harvey Goldstein

Stat-JR templates written by: Chris Charlton and William Browne

Centre for Multilevel Modelling, Graduate School of Education

University of Bristol, BS8 1JA, UK

Contents

Release No.	Date	Revision Description
v1.0	23/11/2016	First draft.
v2.0	28/04/2017	Updated in light of changes to 2LevelMissingOnePass template (v1.0.1, also released 28/04/2017), and also tutmiss and tutmiss_lev2 datasets; also includes addition of appendix.

Introduction

This document describes two approaches to handling missing data in multilevel generalised linear models. The first described procedure uses ‘multiple imputation’ which is a widely used procedure that will handle a large number of models. A 2-level and N-level version are described with full running details. The second (one pass) approach is a more recent generalisation that will handle a wider range of models and is considerably faster with a more robust theoretical justification. See Table 1 for a summary of the main features of each of Stat-JR’s principal templates which handle missing data.

The procedures are implemented in Stat-JR which is a software package that allows users, through its menu interface, to produce appropriate code and to run statistical models in a variety of packages. In addition it will fit models that are not readily available elsewhere. It operates through a series of ‘templates’ that communicate with the user and run models. In this document we describe three different templates for dealing efficiently with missing data in statistical models. These templates differ in terms of their relative speed of execution and the flexibility and generality of models that can be fitted. In due course they will be integrated into a single template that will be quite general. We will also refer to super-templates that are in effect templates that make considerable use of other templates for some of their procedures.

All procedures use what is known as a ‘joint modelling approach’ as opposed to a conditional modelling or ‘chained equation’ approach. This has certain methodological advantages but it also tends to be computationally slower. The principal advantages lie in the ability to handle multilevel data, including variables defined at level 2 that have missing values, and (in the **2LevelMissingOnePass** template) the ability properly to handle interactions and polynomial terms in the user’s substantive model of interest (MOI). Readers are encouraged to consult Carpenter and Kenward (2013) and Goldstein, Carpenter and Browne (2014) for further details.

Template	**2LevelImpute**	**NLevelImpute**	**2LevelMissingOnePass**
Methodology	Multiple imputation (joint modelling approach)	Multiple imputation (joint modelling approach)	Fully Bayesian 'one pass' procedure†
Multilevel structure	Up to 2 levels	Up to N levels (nested and/or cross-classified)	Up to 2 levels
Model of interest (MOI) response types	Normal, binary, Poisson, multivariate Normal*	Normal, binary, Poisson, multivariate mixed (Normal, binary, ordered, unordered; at any level / classification)*	Normal, binomial, Poisson, negative binomial ¥
Imputation model response types	Normal, binary, ordered, unordered	Normal, binary, ordered, unordered	Normal, binary
Handles polynomial / interaction terms in MOI?	No	No	Yes
Allows for random slopes / coefficients in MOI?	Yes, but for univariate response models only	Yes, but for univariate response models only	Yes
Template dependencies	1LevelMod 1LevelMVMixedResponsecc 1LevelMVNormal 2LevelMod 2LevelMVNormal 2LevelRS CompleteCases Generate Merge Resp2LevelMVMixedResponsemvu Take	1LevelMod 1LevelMVMixedResponsecc 1LevelMVNormal CompleteCases Merge NLevelMod NLevelMVNormal NLevelRS RespNLevelMVMixedResponsemvu Take	N/A

Table 1. Summary of the main features of Stat-JR's three principal templates which handle missing data.
† see: Goldstein, Carpenter & Browne (2014) for further details.
* Poisson or binomial (*cf.* binary) can only be used if no missing data in these variables; see the 'Imputation model response types' row in the table for variable types for which missing data is allowed.
¥ In the case of the 2LevelMissingOnePass template, if there are any missing values in the MOI response variable then that case will be automatically dropped.

It is assumed that the user has a knowledge of the basics of methods for missing data using a joint modelling approach. If not, and for a general background overview visit the missing data web site http://missingdata.lshtm.ac.uk/ that has references to papers and recent developments.

Some users will be familiar with the existing **REALCOM** procedures which fit essentially the same models as **2levelImpute** (below) but Stat-JR provides a very much faster implementation.

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

The Stat-JR interface we shall be using is known as TREE. Each of the three templates will be described in turn. For the first template, **2levelImpute**, we shall go through the setting up in detail with an example dataset. The same dataset will be used for the following two templates where the description will be limited to those aspects where the setting up of the model differs.

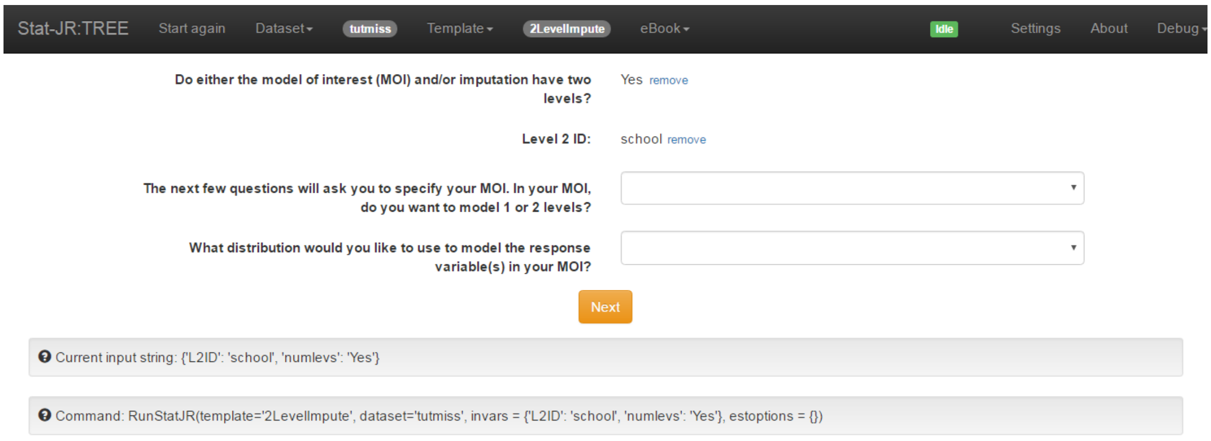
2LevelImpute

This template will fit 2-level models where the model of interest (MOI) may be multivariate and the responses can be normal or categorical. It will produce a specified set of 'completed' datasets, with no missing values. Each of these completed datasets will contain randomly 'imputed' values where the original data were missing and for each of these the MOI is fitted and the separate results combined according to simple rules to produce a final set of parameters estimates together with estimated standard errors.

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

Using the TREE interface

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 (see http://www.bristol.ac.uk/cmm/software/statjr/manuals/).



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?

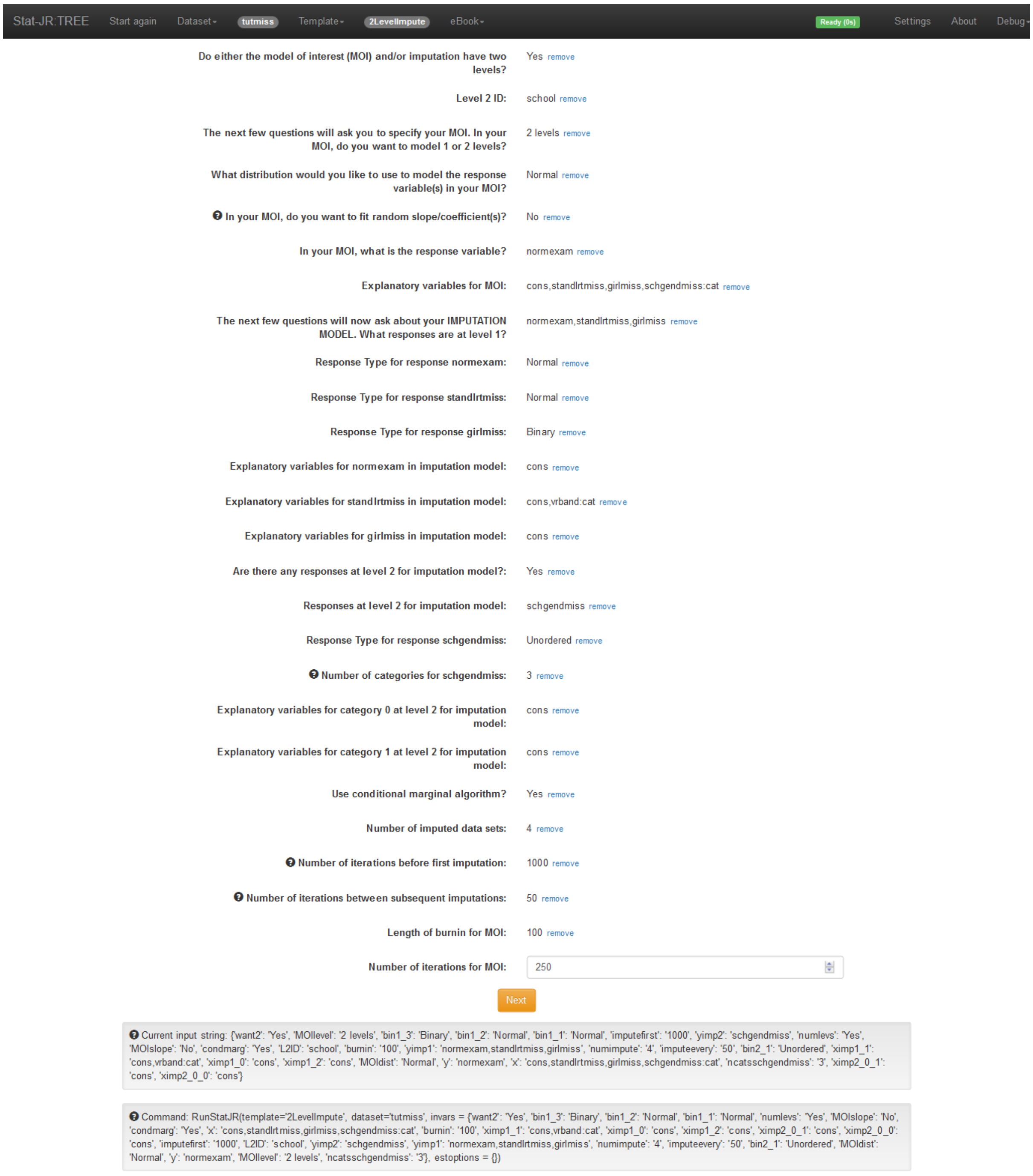
What distribution would you like to use to model the response variable(s) in your MOI?

```
Current input string: {'L2ID': 'school', 'numlevs': 'Yes'}
```

```
Command: RunStatJR(template='2LevelImpute', dataset='tutmiss', invars = {'L2ID': 'school', 'numlevs': 'Yes'}, estoptions = {})
```

You can, of course, upload your own dataset in TREE as well. If it is already saved as a .dta (STATA) file, then you can do so via the black menu bar at the top of the screen. Choose (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). 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 next screenshot, 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).



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

```
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'}
```

```
Command: RunStatJR(template='2LevelImpute', dataset='tutmiss', 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 = {})
```

Overview of inputs

Here we give a brief overview of the inputs that have led to the above screen display. Stat-JR requires these 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 model has 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, for example 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 for 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. The variables are referred to by their names.

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 for each one. 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, but not relevant to the MOI.
- 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 number of iterations before the first imputation (this is setting the interval between adaptation ending and the first imputation, i.e. it effectively sets the burn-in but is nevertheless included in diagnostics returned to aid model-checking), the interval between subsequent iterations (if parallel chains are run, then only the 1st imputation is used; if not parallel chains (i.e. if fewer processors than imputed datasets requested), then this input question pertains to the interval at which iterations are subsequently sampled from the chain) and, for the MOI, the burn in and number of iterations. The choice of number of completed datasets is a topic on which much has been written. It is commonly not less than 5, and may be as high as 20 or more, especially for multilevel data, depending on the amount and pattern of missing values. See Carpenter and Kenward (2013) for further discussion.

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.

A 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 (Rasbash et al, 2014)

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 (although 0=mixed; 1=boys' school; 2=girls' school), 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.
nonmixedmiss	4059	439	0	1	As schgendmiss, but transformed such that 0=mixed; 1=single sex school.

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

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/coefficient(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 in the imputation model *must* have no missing values - if any variable does, then include it as a response. The default missing value code is -9.999*10^{29}

Level 1

- Given our MOI, we suggest the response 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. This implies that **vrband** is being treated as an auxiliary variable that may be associated with the propensity for **standlrtmiss** values to be missing (it is measured around the same time at age 11 for the children).
  - for **girlmiss** use **cons**.
- Since we have in fact 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, that is for any variables defined at level 2 that have missing data. 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 based upon the joint distribution of the variables with missing values, strictly applies only where these have a joint multivariate normal. 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.
- 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 normal draws we condition on the other responses (and any 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 number of iterations before the first imputation, and the number of iterations between subsequent iterations. For the MOI you will then be asked for 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, click 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 output 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 you choose:

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

on a machine with four processors, you 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 However, since this number is less than the requested **5** imputed datasets 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 1.0.3 onwards). 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.

Note that if you wish to rerun the model, perhaps with some changes, then you can repopulate your input values with those you used from a previous run as shown in the grey boy titled "Current input string”. To access this string, select **Template > Set inputs** (via the black bar at the top) and then select the particular template execution you wish to revisit from the list under “History”, and then press the **Use** button; you will then see the input values repopulated with your earlier choices (note that it is advisable to first clear your current input values by pressing **Start again** via the black bar at the top). You can edit the selected input string in the **Set inputs** dialogue box – e.g. to change a particular input value – prior to pressing **Use**.

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 run 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 run 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. Ideally the Markov chain should act like an autoregressive process of order 1. 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.

NLevelImpute

This template is a direct generalisation of **2LevelImpute**. However, we further allow, in the model of interest, a multivariate response with mixed types (Normal, binary, ordered and unordered categorical) at any higher-level classification that has been specified. If a response only at level 1 is specified, then the following choices are available: multivariate Normal, univariate Normal, binomial and Poisson. It is planned to remove this restriction on the response types in a later release.

The data input and output follow the same pattern, but now the user will be asked for the number of levels (classifications) and a set of questions about the variables to be imputed for each level. Note that we use the term classification to include cross classified units as well as higher levels, so that these can also be fitted in this template.

We shall not therefore discuss this template in detail. Note, however, that it has not been as widely tested as **2LevelImpute**, and the STAT-JR team will be very happy to receive comments and questions, which can be posted to the Stat-JR forum (accessible via https://www.cmm.bristol.ac.uk/forum/) or using the bug report form, if appropriate (https://www.cmm.bris.ac.uk/clients/bugreport/).

2LevelMissingOnePass

We now describe a template that will deal with 2-level data having missing values, and will also handle interaction and power terms properly. Rather than producing a set of completed, imputed value, datasets to which the MOI is fitted, at each iteration of the MCMC algorithm it estimates both the imputation model and the MOI, resulting in both a chain of imputed values and a chain for the parameters of the MOI. The latter are what the user requires for inference, and the former may also be used if, for example, some secondary data analysts require a set of imputed datasets.

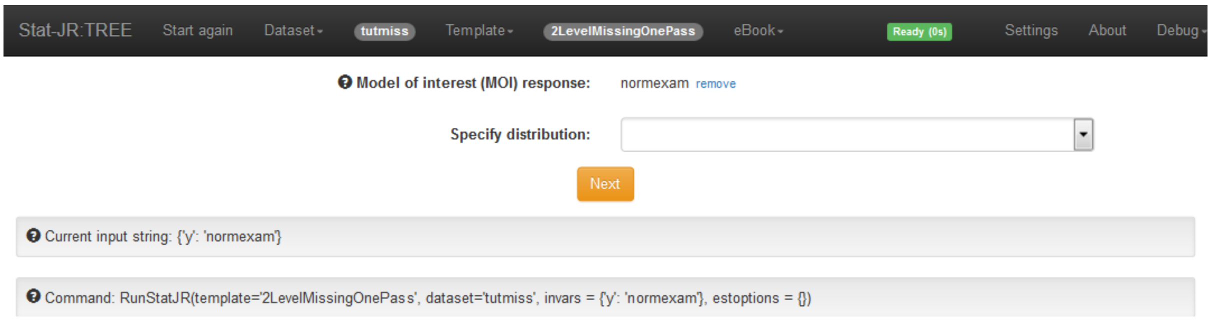
In what follows we will describe the use of this where there is missing data in level 1 variables. The template will also handle level 2 variables with missing values and an outline example is given in the Appendix, along with a brief technical description of the algorithmic steps used by the estimation algorithm.

We shall illustrate the use of **2LevelMissingOnePass** by going through fitting an example which includes an interaction thus:

$$\mu_i = \beta_0 \mathrm{cons}_i + \beta_1 \mathrm{standlrtmiss}_i + \beta_2 \mathrm{girlmiss}_i + \beta_3 \mathrm{girlmiss} \cdot \mathrm{standlrtmiss}_i$$

First of all select the **2LevelMissingOnePass** template and the **tutmiss** dataset.

The following screen will appear:



Model of interest (MOI) response: normexam remove

Specify distribution:

Current input string: {'y': 'normexam'}

Command: RunStatJR(template='2LevelMissingOnePass', dataset='tutmiss', invars = {'y': 'normexam'}, estoptions = {})

We shall not go through all the set up process (although you can see all the responses we have made in the screenshot further below); we will fit the same model as before with the addition of an interaction term between **girlmiss** and **standlrtmiss**. We now explain how the MOI is specified and how polynomials and interactions are included.

Specifying the variables in the MOI

You will be asked a series of questions about which explanatory variables you would like to include in your MOI.

When choosing your explanatory variables for your MOI, note that the imputation model currently only supports normal and binary variables as responses; therefore **only models of interest which have missing data in normal and binary variables** (i.e. not in other categorical variables) are suitable for use with this template. You can fit other variable types (categorical) in the MOI but:

a. these should appear as covariates (not responses) in the imputation component of the model;

b. they must have no missing values;

c. you will need to generate dummy variables for them yourself prior to their use with this template.

Polynomials

You will first be asked if there are any covariates with polynomial terms.

- Note that, for a given variable (e.g. **MyVar**), if you wish to fit **MyVar^2** and **MyVar^3** (i.e. quadratic and cubic terms) you would need to request **2** polynomial terms here, and then separately specify each below (i.e. to maximise user control, lower-order powers are not automatically fitted on your behalf).[1]

Interactions

Having specified any polynomial terms, you're then asked how many interaction terms you would like.

- Note that again, you need to specify all interaction terms separately. E.g. if you had a three-way interaction (between **MyVar1**, **MyVar2**, **MyVar3**), and wished to respect marginality

[1] E.g. if we wished to investigate a nonlinear effect of **standlrtmiss** by including its square then we would answer **1** for **Number of polynomial terms**, then choose **standlrtmiss** as **Variable A**, and enter **2** when asked for the **Power for variable A**.

and fit the lower-order two-way interactions as well (between: **MyVar1** and **MyVar2**; **MyVar1** and **MyVar3**; **MyVar2** and **MyVar3**) you would need to request **4** interaction terms, and specify each separately when prompted by the subsequent input questions.

- In the example of the **tutmiss** dataset, we might be interested in the interaction of **standlrtmiss** and **girlmiss**, for instance. In that case we would enter **1** for the **Number of interaction terms** and then choose **standlrtmiss** and **girlmiss** as the **Variables to include in interaction term 1**.

Explanatory variables

You will next be asked for the explanatory variables for the MOI. If you had earlier specified any polynomial and interaction variables they will be available for inclusion in the model. Also, if applicable (i.e. if earlier indicated that there is a higher level in the MOI) you will be asked for random coefficients at level 2 (you will need to include the intercept term if the intercept is to randomly vary).

Specifying the responses for imputation

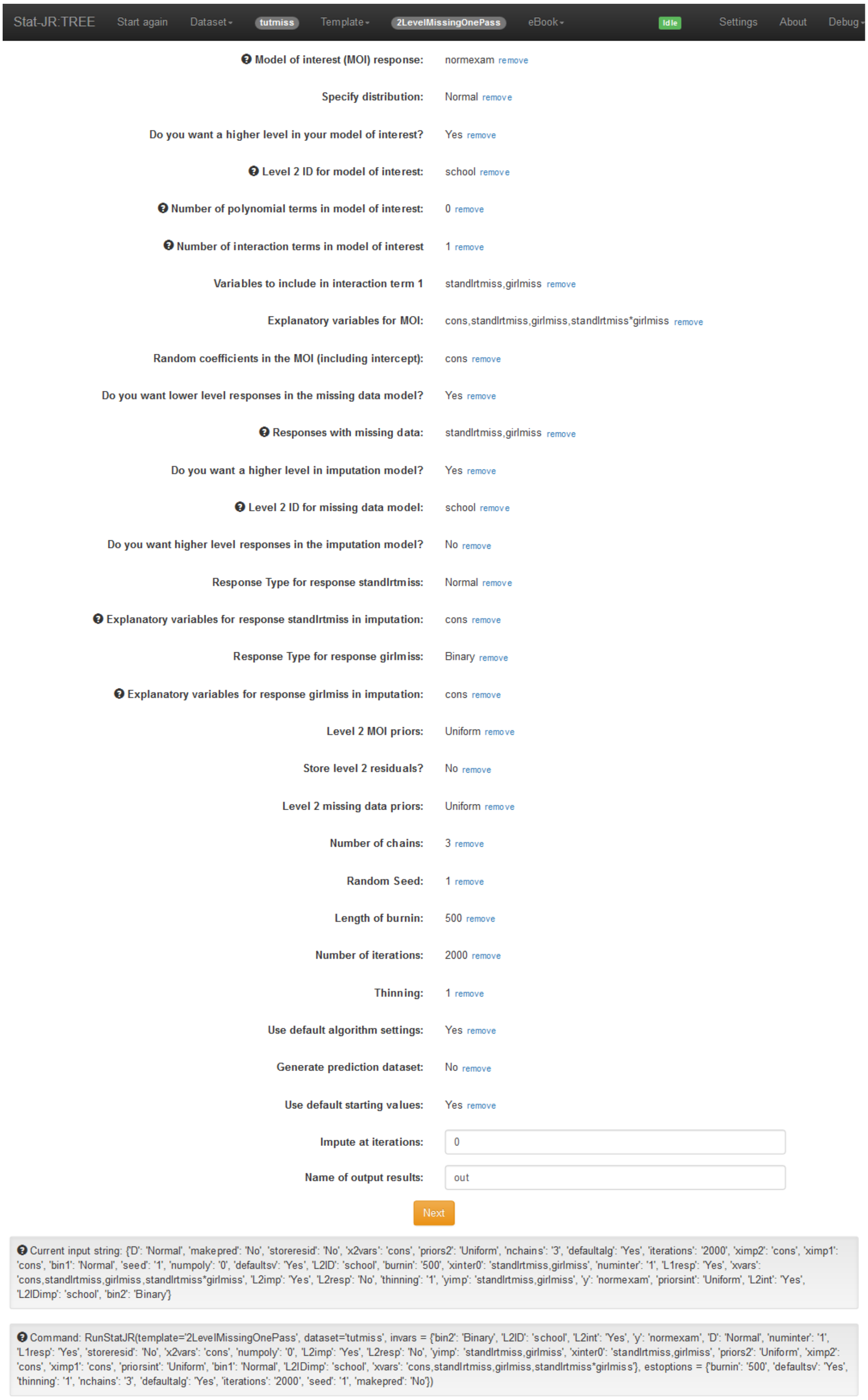
Once you have specified the MOI, you will then be asked to choose your responses for the imputation component, and indicate their distribution and explanatory variables. Note that explanatory variables in the imputation component should have no missing values and should not include variables that are already specified for the model of interest.

Note the following:

1. The imputation component **allows level 1 and level 2 variables as responses**. If you indicate that you do have higher-level responses in the imputation model you will then be asked for a level 2 dataset (see note earlier in this section);
2. Variables having missing values can **only be normal or binary variables**;
3. You should not **include the MOI response variable** in the imputation model since it already appears as part of the model in the MOI component – if there are any missing values in the MOI response variable then that case will be automatically dropped;
4. Note that if you have earlier specified polynomial and/or interaction terms then they *won't* be available for you to choose as part of the imputation model.

For our example, the final screen with the model set up is as follows (with the input string quoted first):

*{'D': 'Normal', 'makepred': 'No', 'storeresid': 'No', 'x2vars': 'cons', 'priors2': 'Uniform', 'nchains': '3', 'defaultalg': 'Yes', 'iterations': '2000', 'outdata': 'out', 'ximp2': 'cons', 'ximp1': 'cons', 'bin1': 'Normal', 'seed': '1', 'numpoly': '0', 'defaultsv': 'Yes', 'imputeiters': '0', 'L2ID': 'school', 'burnin': '500', 'xinter0': 'standlrtmiss,girlmiss', 'numinter': '1', 'L1resp': 'Yes', 'xvars': 'cons,standlrtmiss,girlmiss,standlrtmiss*girlmiss', 'L2imp': 'Yes', 'L2resp': 'No', 'thinning': '1', 'yimp': 'standlrtmiss,girlmiss', 'y': 'normexam', 'priorsint': 'Uniform', 'L2int': 'Yes', 'L2IDimp': 'school', 'bin2': 'Binary'}*

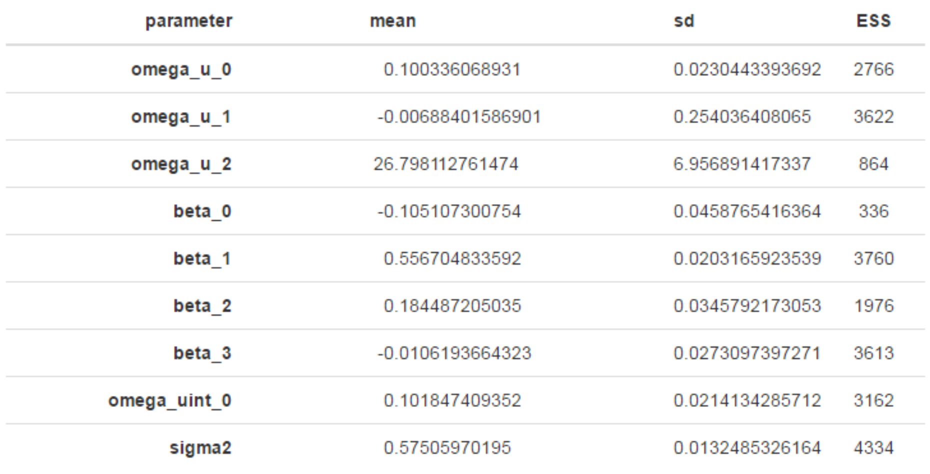


Model of interest (MOI) response:	normexam remove
Specify distribution:	Normal remove
Do you want a higher level in your model of interest?	Yes remove
Level 2 ID for model of interest:	school remove
Number of polynomial terms in model of interest:	0 remove
Number of interaction terms in model of interest	1 remove
Variables to include in interaction term 1	standlrtmiss,girlmiss remove
Explanatory variables for MOI:	cons,standlrtmiss,girlmiss,standlrtmiss*girlmiss remove
Random coefficients in the MOI (including intercept):	cons remove
Do you want lower level responses in the missing data model?	Yes remove
Responses with missing data:	standlrtmiss,girlmiss remove
Do you want a higher level in imputation model?	Yes remove
Level 2 ID for missing data model:	school remove
Do you want higher level responses in the imputation model?	No remove
Response Type for response standlrtmiss:	Normal remove
Explanatory variables for response standlrtmiss in imputation:	cons remove
Response Type for response girlmiss:	Binary remove
Explanatory variables for response girlmiss in imputation:	cons remove
Level 2 MOI priors:	Uniform remove
Store level 2 residuals?	No remove
Level 2 missing data priors:	Uniform remove
Number of chains:	3 remove
Random Seed:	1 remove
Length of burnin:	500 remove
Number of iterations:	2000 remove
Thinning:	1 remove
Use default algorithm settings:	Yes remove
Generate prediction dataset:	No remove
Use default starting values:	Yes remove
Impute at iterations:	0
Name of output results:	out

```
Current input string: {'D': 'Normal', 'makepred': 'No', 'storeresid': 'No', 'x2vars': 'cons', 'priors2': 'Uniform', 'nchains': '3', 'defaultalg': 'Yes', 'iterations': '2000', 'ximp2': 'cons', 'ximp1': 'cons', 'bin1': 'Normal', 'seed': '1', 'numpoly': '0', 'defaultsv': 'Yes', 'L2ID': 'school', 'burnin': '500', 'xinter0': 'standlrtmiss,girlmiss', 'numinter': '1', 'L1resp': 'Yes', 'xvars': 'cons,standlrtmiss,girlmiss,standlrtmiss*girlmiss', 'L2imp': 'Yes', 'L2resp': 'No', 'thinning': '1', 'yimp': 'standlrtmiss,girlmiss', 'y': 'normexam', 'priorsint': 'Uniform', 'L2int': 'Yes', 'L2IDimp': 'school', 'bin2': 'Binary'}
```

```
Command: RunStatJR(template='2LevelMissingOnePass', dataset='tutmiss', invars = {'bin2': 'Binary', 'L2ID': 'school', 'L2int': 'Yes', 'y': 'normexam', 'D': 'Normal', 'numinter': '1', 'L1resp': 'Yes', 'storeresid': 'No', 'x2vars': 'cons', 'numpoly': '0', 'L2imp': 'Yes', 'L2resp': 'No', 'yimp': 'standlrtmiss,girlmiss', 'xinter0': 'standlrtmiss,girlmiss', 'priors2': 'Uniform', 'ximp2': 'cons', 'ximp1': 'cons', 'priorsint': 'Uniform', 'bin1': 'Normal', 'L2IDimp': 'school', 'xvars': 'cons,standlrtmiss,girlmiss,standlrtmiss*girlmiss'}, estoptions = {'burnin': '500', 'defaultsv': 'Yes', 'thinning': '1', 'nchains': '3', 'defaultalg': 'Yes', 'iterations': '2000', 'seed': '1', 'makepred': 'No'})
```

After clicking **Run** the output estimates (via **ModelResults** in the results output pane) are as follows:



parameter	mean	sd	ESS
omega_u_0	0.100336068931	0.0230443393692	2766
omega_u_1	-0.00688401586901	0.254036408065	3622
omega_u_2	26.798112761474	6.956891417337	864
beta_0	-0.105107300754	0.0458765416364	336
beta_1	0.556704833592	0.0203165923539	3760
beta_2	0.184487205035	0.0345792173053	1976
beta_3	-0.0106193664323	0.0273097397271	3613
omega_uint_0	0.101847409352	0.0214134285712	3162
sigma2	0.57505970195	0.0132485326164	4334

We see that these are essentially the same as we had before, allowing for the fact that we have also fitted an interaction (which is not statistically significant at the 5% level). Note omega_uint_0 is the level 2 variance (i.e. σ_u^2; the 'int' refers to the model of interest). Again, we have diagnostics available in the form of MCMC chains. Note that should you wish to see the estimates of any of the coefficients in the imputation component, then these are accessible via **modelparameters.dta**.

In the current version, DIC statistics are not available.

You may find that the program crashes. We are interested to know about such occurrences (see above) but please first of all check that your model is a sensible one!

References

Carpenter, JR. and Kenward, MG. (2013). Multiple imputation and its application. Chichester, Wiley.

Goldstein, H., Carpenter, J. R. and Browne, W. J. (2014), Fitting multilevel multivariate models with missing data in responses and covariates that may include interactions and non-linear terms. Journal of the Royal Statistical Society: Series A (Statistics in Society). 177(2), 553-564
doi: 10.1111/rssa.12022

Appendix

Example

Suppose now that we have one or more level 2 variables in our model of interest that have missing values. As with level one variables with missing values these should either be normal or binary. If you specify that you wish to have higher level responses in the imputation model then you will be asked for a 'level 2' dataset where each record is at level 2 and contains any relevant level 2 explanatory variables and also those level 2 variables having missing data.

To illustrate we use the dataset **tutmiss_lev2**; a summary is as follows

Name	Count	Missing	Min	Max	Mean	SD
school	65	0	1.00	65.00	33.00	18.76
cons	65	0	1.00	1.00	1.00	0.00
schgend	65	0	0.00	2.00	0.62	0.74
avslrt	65	0	-0.76	0.64	-0.03	0.34
schav	65	0	0.00	2.00	0.71	0.80
schgendmiss	65	7	0.00	2.00	0.74	0.88
avslrtmiss	65	7	-0.76	0.64	-0.03	0.35
nonmixedmiss	65	7	0.00	1.00	0.45	0.50

We see that there are three variables with missing data: the average school LRT score (**avslrtmiss**), and two variables relating to whether schools are single sex or not (**schgendmiss** and **nonmixedmiss**). The former categorises schools as either mixed, boys' schools or girls' schools, whilst the latter is simply a binary transformation of that (indicating whether or not the school is mixed or single sex) and is included in the dataset should the user wish to explore fitting models to categorical level 2 variables (since we can only utilise binary variables with missing values). Note that we also have these same variables (or variables from which they are otherwise derived) in their non-missing versions; missing values have been introduced completely at random.

We fit a simple model with **normexam** as response and **girlmiss**, **standlrt** and **avslrtmiss** as predictors, i.e. with just one variable at level 1 and one variable at level 2 having missing values. Note that we could have fitted **standlrtmiss** (**standlrt** with missing values) to give two variables with missing data at level 1. However, while this model can be estimated, we find that the results are somewhat unstable since the level 2 variable is in fact calculated as the average of the level 1 values yet we separately estimate imputed values for the level 2 where missing, rather than averaging from the imputed level 1 values. In future versions of the software such a possibility will be allowed. The input string (see earlier), and MOI parameter estimates are as follows:

{'imputeiters': '0', 'L2int': 'Yes', 'D': 'Normal', 'storeresid': 'No', 'L2IDimp': 'school', 'x2vars': 'cons', 'y': 'normexam', 'nchains': '3', 'defaultalg': 'Yes', 'iterations': '2000', 'ximp2_1': 'cons', 'outdata': 'out', 'burnin': '500', 'ximp1': 'cons', 'seed': '1', 'numpoly': '0', 'defaultsv': 'Yes', 'L2Data': 'tutmiss_lev2', 'L2ID': 'school', 'yimp2': 'avslrtmiss', 'numinter': '0', 'L1resp': 'Yes', 'makepred': 'No', 'L2imp': 'Yes',

'L2resp': 'Yes', 'thinning': '1', 'yimp': 'girlmiss', 'priors2': 'Uniform', 'priorsint': 'Uniform', 'bin1': 'Binary', 'bin2_1': 'Normal', 'xvars': 'cons,standlrt,girlmiss,avslrtmiss'}

parameter	mean	sd	ESS	
beta_0	-0.08442	0.042128	441	**cons**
beta_1	0.557172	0.012613	5589	**standlrt**
beta_2	0.169034	0.033923	2046	**girlmiss**
beta_3	0.273155	0.119529	295	**avlrtmiss**
omega_uint_0	0.086238	0.018993	2783	
sigma2	0.562936	0.012586	5634	

You can verify these values correspond closely to those that are obtained from the non-missing dataset.

An outline of algorithm steps

We set out here the steps involved with imputing missing values and estimating the parameters of the imputation models. At each MCMC iteration, given the imputed values the steps for the 2-level model of interest are standard.

To illustrate the steps we shall use a simple model as follows:

$$y_{ij} = \beta_0 + \beta_1 x_{1ij} + \beta_2 x_{2j} + u_{oj} + e_{0ij} \qquad \text{A1}$$

Where both explanatory variables are normally distributed and have missing values, so that the imputation components are:

$$x_{1ij} = \alpha_0 + u_{1j} + e_{1ij} \qquad e_{ij} \sim N(0, \sigma_e^2) \qquad \text{A2}$$

$$x_{2j} = \alpha_1 + u_{2j} \qquad \text{A3}$$

$$\begin{pmatrix} u_{oj} \\ u_{1j} \\ u_{2j} \end{pmatrix} \sim N\begin{pmatrix} \sigma_{u0}^2 & & \\ 0 & \sigma_{u1}^2 & \\ 0 & \sigma_{u12} & \sigma_{u2}^2 \end{pmatrix} \qquad \text{A4}$$

We note that the level 2 random effect in the MOI is independent of those in the imputation components – this is essentially the standard assumption of endogeneity.

When the template displays the estimated model parameters, the u_{oj} are designated as ‘uint0_n’ where n indexes the 65 level 2 units numbered in this case 0,….,64. The u_{1j} are designated ‘u0_n’ where n again indexes the 65 level 2 units numbered in this case 0,….,64. The estimate for α_0 is designated ‘beta2imp_0’. The parameter σ_{u0}^2 is designated as ‘omega_uint_0’ and the (2 x 2) covariance matrix elements $\sigma_{u1}^2, \sigma_{u12}, \sigma_{u2}^2$ are designated as ‘omega_u_0’, ‘omega_u_1’ , and ‘omega_u_2’ respectively. The estimate of σ_e^2 is designated as ‘sigma2’.

The sampling steps involve updating the level 2 covariance matrix for the imputation components, updating the level 1 and level 2 variances from the MOI, and updating the fixed coefficients in A1-A3 utilising the appropriate variance and covariance elements, and sampling level 2 effects. All of these can use Gibbs steps, and there are metropolis steps for each of the missing values in turn, using A1-A2 and A1-A3.