MplusAutomation Examples

Michael N. Hallquist

Package version 0.2-5

Contents

1 Overview

This guide provides examples of how to use the functions in the MplusAutomation package. The package is designed to automate three major aspects of latent variable modeling in Mplus: 1) creating related groups of models, 2) running batches of models, and 3) extracting and tabulating model parameters and fit statistics.

The guide tries to make few assumptions about the user's familiarity with the R environment, as many Mplus user's may be unfamiliar with R.

Note: all examples herein rely on examples from the Mplus User's Guide. The input and output files for these examples are assumed to reside in the directory: `C:\Program Files\Mplus\Mplus Examples\User's Guide Examples`. If you have installed Mplus to a different location, please use the correct directory in the code below.

2 Installing and loading the package

The package was built using R 2.10.1, although it should be compatible with older versions of R. To obtain R 2.10.1 for Windows, follow this link: http://cran.r-project.org/bin/windows/base/R-2.10.1-win32.exe

After installing the program and launching R, type the following at the command line:

```
> install.packages("MplusAutomation", .Library)
```

Depending on which version of Windows you're using, you may need to run R as an administrator (right-click, Run as administrator).

To load the MplusAutomation package so that its functions are available for your use, type:

```
library(MplusAutomation)
```

3 Running batches of Mplus models

A major purpose of the MplusAutomation package is to allow for easy runs of batches/groups of Mplus models. Oftentimes, one wants to compare a group of related models, such as testing for different forms of measurement invariance. Depending on the complexity of the models, Mplus can take several minutes to many hours to run each model. The `runModels` routine is designed to run a group of related models located within a directory (or nested subdirectories).

3.1 Basic use of runModels

As an example, say that we want to run all of the models used in the Mplus 5.1 Addendum: http://statmodel.com/download/examples1.pdf. The input files for these are located in: `C:\Program Files\Mplus\Mplus Examples\Addendum Examples`.

To run this batch, enter this at the command line:

```
> runModels("C:/Program Files/Mplus/Mplus Examples/Addendum Examples")
```

Note that you need to use forward slashes ("/"), not backslashes("\") in the path name. Alternatively, you could use double backslashes (e.g., `"C:\\Program Files"` etc.).

3.2 Advanced use of runModels

3.2.1 Recursing through subdirectories

Sometimes it is useful to organize models into one or more subdirectories where each subdirectory contains models for a particular outcome or analytic approach. For example, if one were contrasting latent class analysis (LCA) with confirmatory factor analysis (CFA), one could place all LCA models in a single directory and place CFA models in a different directory. This might yield a file structure like this:

```
ComparingLCAvCFA/LCA/1-class LCA.inp
ComparingLCAvCFA/LCA/2-class LCA.inp
ComparingLCAvCFA/LCA/3-class LCA.inp

ComparingLCAvCFA/CFA/1-factor CFA.inp
ComparingLCAvCFA/CFA/2-factor CFA.inp
ComparingLCAvCFA/CFA/3-factor CFA.inp
```

In this case, all Mplus files for the larger project are housed within a parent directory, `ComparingLCAvCFA`. To run all models within `ComparingLCAvCFA`, including models within any subdirectories (including `LCA` and `CFA`), use the `recursive` parameter of `runModels`.

```
> runModels("C:/Data_Analysis/ComparingLCAvCFA", recursive=TRUE)
```

3.2.2 Logging the outcome of runModels

The `logFile` parameter of `runModels` allows the user to specify a text file containing the results of `runModels`. Included in the log file are the parameters passed to the function, the date when the batch started, which models were run (and which were skipped), and any actions taken if the R process was interrupted (e.g., terminating the Mplus process). By default, `runModels` will create a log file in the same directory as the models to be run, `directory`, called `Mplus Run Models.log`. To change the name or location of this file, specify the `logFile` parameter, such as in this example:

```
> runModels("C:/Data_Analysis/ComparingLCAvCFA", recursive=TRUE, logFile="C:/CFALCA-Comparison-Log.txt")
```

Here, the file `C:/CFALCA-Comparison-Log.txt` will be created in the directory `C:/`.

To specify that no log file should be created, pass `logFile=NULL` to `runModels`.

```
> runModels("C:/Data_Analysis/ComparingLCAvCFA", recursive=TRUE, logFile=NULL)
```

3.2.3 Skipping models with existing output files

Sometimes it is useful to skip models that have already been run to avoid the computing time associated with running all input files within a directory. The `replaceOutfile` parameter allows one to specify which models should be re-run, where models that have an output with the same filename as the input file are considered to have been run. By default, `replaceOutfile="always"`, meaning that all input files will be run, regardless of whether they have a matching output file.

To skip any model that already has an existing output file, pass `replaceOutfile="never"` to `runModels`, such as in this example:

```
> runModels("C:/Data_Analysis/ComparingLCAvCFA", recursive=TRUE, replaceOutfile="never")
```

Oftentimes, after a model or group of models has been run, it is necessary to modify some aspects of the parameterization to improve model fit or address estimation problems. In such cases, output files are inspected and the corresponding input files are modified. In such cases, one may only want to run models that have been updated, but not to re-run models that completed successfully. This can be accomplished by passing `replaceOutfile="modifiedDate"` to `runModels`. The `"modifiedDate"` determines whether there is an existing output file for a given input file. If there is, it checks to see whether the date the input file was modified is *newer* than the output file. If the input file is newer, then the model is run. Otherwise, it is skipped. Here is an example:

```
> runModels("C:/Data_Analysis/ComparingLCAvCFA", recursive=TRUE, replaceOutfile="modifiedDate")
```

3.2.4 Displaying R output in the console

When models are run by the Mplus Windows program (MplusWin.exe), a separate DOS window appears that documents the TECH8 progress of the model, which represents the progress toward maximum-likelihood convergence for the model (including random starts and final stage optimizations for some models). To display the same TECH8 output for models run by `runModels`, pass `showOutput=TRUE` to the `runModels` function.

```
> runModels("C:/Data_Analysis/ComparingLCAvCFA", recursive=TRUE, showOutput=TRUE)
```

If the R session was started through the R GUI (Rgui.exe), the output will be displayed within the R window. If the R session was started using Rterm (the R terminal), a separate DOS window will display the output, as occurs using the built-in Mplus Windows program.

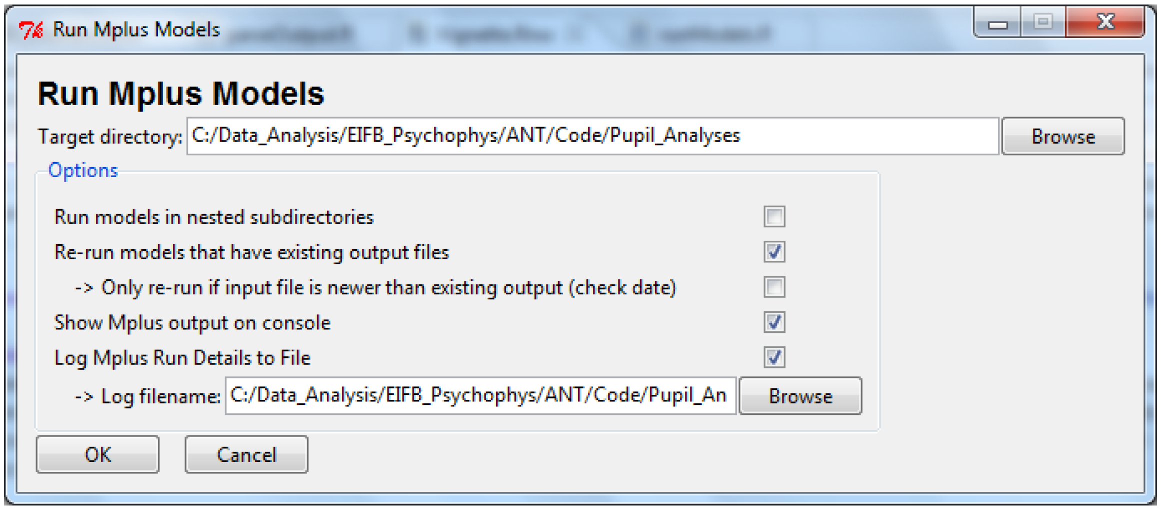
By default, the output is not shown, `showOutput=FALSE`.

4 User-friendly interface to runModels

A wrapper function, `runModels_Interactive`, is included in the `MplusAutomation` package, which provides a simple dialog box for specifying which models to run. To start the interface, type the following:

```
> runModels_Interactive()
```

The picture below documents the appearance of this interface:



Run Mplus Models

Run Mplus Models

Target directory: C:/Data_Analysis/EIFB_Psychophys/ANT/Code/Pupil_Analyses

Options

Run models in nested subdirectories ☐

Re-run models that have existing output files ☑

-> Only re-run if input file is newer than existing output (check date) ☐

Show Mplus output on console ☑

Log Mplus Run Details to File ☑

-> Log filename: C:/Data_Analysis/EIFB_Psychophys/ANT/Code/Pupil_An

Although one can provide parameters to the function to set the initial states of the interface, it is rarely necessary to do so, so the syntax above should suffice.

5 Extracting model summary statistics

Another major purpose of the package is to allow for easy extraction of model summary statistics from a group of models. Such summary statistics include items such as log-likelihood values, root mean squared error of approximation (RMSEA), and Akaike's Information (AIC).

The `extractModelSummaries` function is designed to extract model summaries from a group of models located within a directory (or nested within subdirectories). This function returns a `data.frame` containing one row per model, with columns representing several fit statistics.

A basic call to the function includes the directory containing output files to be parsed:

```
> mySummaries <-extractModelSummaries("C:/Data_Analysis/ComparingLCAvCFA", recursive=TRUE)
```

Now, the variable mySummaries is a `data.frame` containing summary statistics about models contained in the `ComparingLCAvCFA` directory.

As with `runModels`, `extractModelSummaries` includes a `recursive` parameter that specifies whether to parse output files located in subdirectories beneath the target directory (defaults to `FALSE`).

In addition, `extractModelSummaries` also includes a parameter, `filefilter`, that allows the user to parse only files matching certain search criteria. `filefilter` accepts a Perl-compatible regular expression string. If you're unfamiliar with regular expressions in Perl, I suggest these two websites:

http://www.pcre.org/pcre.txt

http://www.regular-expressions.info/

Note that many regular expression in Perl rely on backslashes (\) for defining character classes, escaping certain characters, and so on. In R, backslashes contained in strings must be doubled (i.e., \\).

Here is an example of filtering only files that match ”ex4” followed by any characters Note that the function automatically searches only files with the .out extension, so it isn’t necessary to include .out in the file filter.

```
summaryStats <- extractModelSummaries("C:/Program Files/Mplus/Mplus Examples/User's Guide
    Examples/Outputs", filefilter="ex4.*")
```

Here is a more complex filter that matches filenames that begin with the digits 1, 2, or 3 (for 1-class, 2-class, or 3-class output files) and also contain the text ”Threshold”:

```
summaryStats <- extractModelSummaries("C:/Data_Analysis/Multiclass Models",
    filefilter="[123]{1}-class.*Threshold.*")
```

5.1 Listing of summary statistics extracted by extractModelSummaries

As of this version of the package (0.2-5), the following summary statistics are automatically extracted, when available:

- `Title`: Title for the model, specified by the TITLE: command
- `Filename`: Filename of the output file
- `InputInstructions`: A string containing the full input syntax for the model
- `Estimator`: Estimator used for the model (e.g., ML, MLR, WLSMV, etc.)
- `LL`: Log-likelihood of the model
- `BIC`: Bayesian Information Criterion
- `aBIC`: Sample-Size-Adjusted BIC (Sclove, 1987)
- `AIC`: Akaike’s Information Criterion
- `AICC`: Corrected AIC, based on Sugiura (1978) and recommended by Burnham & Anderson (2002)
- `Parameters`: Number of parameters estimated by the model
- `Observations`: The number of observations for the model (does not suppport multiple-groups analysis at this time)
- `CFI`: Confirmatory Fit Index
- `TLI`: Tucker-Lewis Index
- `RMSEA_Estimate`: Point estimate of root mean squared error of approximation
- `RMSEA_90CI_LB`: Lower bound of the 90% Confidence Interval around the RMSEA estimate
- `RMSEA_90CI_UB`: Upper bound of the 90% Confidence Interval around the RMSEA estimate
- `RMSEA_pLT05`: Probability that the RMSEA estimate falls below .05, indicating good fit
- `ChiSqM_Value`: Model chi-squared value
- `ChiSqM_DF`: Model chi-squared degrees of freedom
- `ChiSqM_PValue`: Model chi-squared p value
- `BLRT_KM1LL`: Log-likelihood of the K-1 model (one less class) for the Bootstrapped Likelihood Ratio Test (TECH14)
- `BLRT_PValue`: P-value of the Bootstrapped Likelihood Ratio Test (TECH14) testing whether the K class model is significantly better than K-1
- `BLRT_Numdraws`: The number of bootstrapped samples used in the Bootstrapped Likelihood Ratio Test
- `SRMR`: Standardized root mean square residual
- `WRMR`: Weighted root mean square residual
- `ChiSqBaseline_Value`: Baseline (unstructured) chi-squared value
- `ChiSqBaseline_DF`: Baseline (unstructured) chi-squared degrees of freedom
- `ChiSqBaseline_PValue`: Baseline (unstructured) chi-squared p value

The `extractModelSummaries` function is designed to work in conjunction with functions that generate tables of summary statistics.

6 Summarizing model fit statistics in tabular form

Once summary statistics for a group of models have been extracted, it is often useful to display them in tabular form to compare fit among models, sorted by a particular criterion (e.g., AIC).

The MplusAutomation package provides three routines for tabulating model summary statistics. At this time, there are three table-generating functions, which are detailed below: showSummaryTable, HTMLSummaryTable, LatexSummaryTable.

As their names suggest, these functions can create tables for on-screen display (showSummaryTable), as an HTML file containing the table (HTMLSummaryTable), or as a LaTex-formatted table (LatexSummaryTable).

6.1 Displaying the summary table on the screen

The showSummaryTable function is designed to display a summary table of model fit statistics on the screen. The function expects a model list created by extractModelSummaries and allows the user to specify which columns should be included in the table.

Here is a simple example of using showSummaryTable by specifying which columns to keep in the table:

```
showSummaryTable(summaryStats, keepCols=c("Title", "LL", "AIC", "BIC", "CFI"), sortBy="AIC")
```

And another example specifying that all columns in the model list should be displayed *except* those specified:

```
showSummaryTable(summaryStats, dropCols=c("InputInstructions", "Observations", "Parameters"),
    sortBy="CFI")
```

6.2 Creating a summary table in HTML

The HTMLSummaryTable function creates an HTML file containing a summary table of model fit statistics. Its syntax is very similar to showSummaryTable, including parameters such as dropCols, keepCols, and sortBy. Two parameters distinguish it from other summary functions: filename and display.

The filename parameter specifies the path and filename of the HTML file to be created. display specifies whether to display the HTML summary table in the web browser after it is created. Here is a simple of using the function:

```
HTMLSummaryTable(summaryStats, filename="C:/MyModelSummary.html", display=TRUE, keepCols=c("Title",
    "LL", "AIC", "BIC", "AICC"), sortBy="AIC")
```

6.3 Creating a summary table in LaTex

One major strength of R is its ability to be interwoven with LaTex, an advanced typesetting language. The most frequently used approach for combining R and LaTex is Sweave (http://www.stat.uni-muenchen.de/~leisch/Sweave/), a built-in R function that runs R code embedded in a LaTex document, thereby permitting the creation of advanced automated reports.

Mplus model fit summary tables can be formatted in LaTex using the LatexSummaryTable function. Unlike showSummaryTable and HTMLSummaryTable, LatexSummaryTable returns a value, specifically the LaTex syntax for the summary table. Here is a simple example of the function

```
myLatexTable <- LatexSummaryTable(summaryStats, keepCols=c("Title", "BIC", "Parameters"),
    sortBy="Parameters",
        caption="Comparing CFA vs. LCA according to number of parameters", label="CFALCATab")
```

Note that LatexSummaryTable supports two distinct parameters relative to other summary table functions: caption and label. These allow the user to set the caption and label properties of the table, which are used in LaTex for displaying a caption with the table and for allowing the table to be easily referenced in other parts of document, respectively. See http://en.wikibooks.org/wiki/LaTeX/Tables#The_table_environment_-_captioning_etc for further details about LaTex tables.

The LaTex syntax for a summary table could be included in an Sweave document in the following way:

```
<<echo=TRUE, results=tex>>=

myLatexTable <- LatexSummaryTable(summaryStats, keepCols=c("Title", "BIC", "Parameters"),
    sortBy="Parameters", caption="Comparing CFA vs. LCA according to number of parameters",
    label="CFALCATab")
```

```
print(myLatexTable)

@
```

See the Sweave manual for more details about combining LaTex with R.