Let R Make Your Life Easier:

An Introduction to the MplusAutomation Package

Graham G. Rifenbark

February 1, 2017

Talk Objectives

1. Run a single model from R and extract parameter estimates.
2. Generate unique Mplus input files for a simulation.
3. Estimate simulation files.
4. Extract simulation results.
5. Grab bag of tricks

Load MplusAutomation Library

```
library(MplusAutomation)
```

View Available Functions

```
lsp(MplusAutomation)[1:15]
```

```
##  [1] "cd"                      "compareModels"
##  [3] "createModels"            "createSyntax"
##  [5] "extract"                 "extract.mplus.model"
##  [7] "extract.mplusObject"     "extractModelParameters"
##  [9] "extractModelSummaries"   "extractModIndices"
## [11] "getSavedata_Bparams"     "getSavedata_Data"
## [13] "getSavedata_Fileinfo"    "HTMLSummaryTable"
## [15] "LatexSummaryTable"
```

View Available Functions Cont'd

```
lsp(MplusAutomation)[16:30]
```

```
##  [1] "lookupTech1Parameter"      "mplus.traceplot"
##  [3] "mplusModeler"              "mplusObject"
##  [5] "mplusRcov"                 "paramExtract"
##  [7] "parseMplus"                "prepareMplusData"
##  [9] "readModels"                "runModels"
## [11] "runModels_Interactive"     "showSummaryTable"
## [13] "SummaryTable"              "testBParamCompound
## [15] "testBParamConstraint"
```

Single Model

Step No. 1

- Read in data

```
simDat <- read.csv("templateDat.csv",header = TRUE)
```

Variable Names

- Outcome Variable and Level Identifiers

```
colnames(simDat)[1:3]
```

```
## [1] "Y"      "SSID"   "SCHID"
```

Variable Names Cont'd

- Level One Predictors

```
colnames(simDat)[4:8]
```

```
## [1] "X1" "X2" "X3" "X4" "X5"
```

Number of Units Per Level

```
# Level 2
length(unique(simDat$SCHID))
```

```
## [1] 200
```

```
# Level 1
length(unique(simDat$SSID))
```

```
## [1] 10000
```

Goal

- Estimate an organizational MLM including random effects for:
  - the intercept: γ_{00}
  - all within level slopes: $\gamma_{10} : \gamma_{50}$

Mplus Syntax – TITLE/VARIABLE

```
twoLevel <- mplusObject(
TITLE = "Template;",
VARIABLE = "
NAMES = Y SSID SCHID X1 X2 X3 X4 X5;
USEVARIABLES = Y X1 X2 X3 X4 X5;
CLUSTER = SCHID;
WITHIN = X1 X2 X3 X4 X5;",
## ...
```

Mplus Syntax – ANALYSIS

```
ANALYSIS = "TYPE = TWOLEVEL RANDOM;",
```

Mplus Syntax – MODEL

```
MODEL = "
%WITHIN%
S1 | Y ON X1;
S2 | Y ON X2;
S3 | Y ON X3;
S4 | Y ON X4;
S5 | Y ON X5;
%BETWEEN%
Y WITH S1 S2 S3 S4 S5;
S1 WITH S2 S3 S4 S5;
S2 WITH S3 S4 S5;
S3 WITH S4 S5;
S4 WITH S5;
",
```

Mplus Syntax – OUTPUT

```
OUTPUT = "TECH1;")
```

Prepare Data File

```
prepareMplusData(simDat,
                 "template.dat",
                 inpfile = TRUE, # eq "template.inp"
                 overwrite = TRUE,
                 interactive = FALSE)
```

Estimate Model From R via runModels()

```
runModels(recursive = FALSE,
          filefilter = "knitr*",
          replaceOutfile = c("always",
                             "modifiedDate",
                             "never"))
```

```
runModels(filefilter = "knitr*")
```

```
##
## Running model: knitr_template.inp
## System command: C:\windows\system32\cmd.exe /c cd "C:\Us
```

Extract Output

```
singRun <- readModels()
```

```
## Reading model:  C:/Users/grr13002/Dropbox/Conn/Consults/
## Reading model:  C:/Users/grr13002/Dropbox/Conn/Consults/
```

```
names(singRun)
```

```
## [1] "knitr_template.out" "template.out"
```

Determine Slots Available For First Element

```
names(singRun[[1]])
```

```
##  [1] "input"           "warnings"          "errors"
##  [4] "summaries"       "parameters"        "class_counts"
##  [7] "residuals"       "tech1"             "tech3"
## [10] "tech4"           "tech7"             "tech9"
## [13] "tech12"          "fac_score_stats"   "gh5"
```

$errors & $warnings

- Good, no estimation warnings or errors!

```
singRun[[1]]$errors
```

```
## list()
## attr(,"class")
## [1] "list"          "mplus.errors"
```

```
singRun[[1]]$warnings
```

```
## list()
## attr(,"class")
## [1] "list"            "mplus.warnings"
```

$summaries

```
names(singRun[[1]]$summaries)
```

```
##  [1] "Mplus.version"      "Title"              "Analysis
##  [4] "DataType"           "Estimator"          "Observat
##  [7] "Parameters"         "LL"                 "LLCorrec
## [10] "AIC"                "BIC"                "aBIC"
## [13] "AICC"               "Filename"
```

$parameters

```
names(singRun[[1]]$parameters)
```

```
## [1] "unstandardized"
```

Extract Parameter Estimates

```
paramEst <- singRun[[1]]$parameters$unstandardized
colnames(paramEst)
```

```
## [1] "paramHeader"    "param"          "est"              "se"
## [5] "est_se"         "pval"           "BetweenWithin"
```

Store Fixed and Random Components

```
gammaEst <- paramEst[which(paramEst$paramHeader
                           == "Means"),]
tauEst <- paramEst[which(paramEst$paramHeader
                           == "Variances"),]
sigmaEst <- paramEst[which(paramEst$paramHeader
                           == "Residual.Variances"),]
```

Combine into single data.frame

```
singleRunEST <- rbind(gammaEst,tauEst,sigmaEst)
singleRunEST[seq(1,12,2),]
```

```
##    paramHeader param   est    se est_se  pval BetweenWit
## 17       Means     Y 0.038 0.020  1.939 0.052        Betw
## 19       Means    S2 0.013 0.014  0.870 0.384        Betw
## 21       Means    S4 0.180 0.014 12.965 0.000        Betw
## 23   Variances     Y 0.077 0.007 10.573 0.000        Betw
## 25   Variances    S2 0.041 0.004 10.105 0.000        Betw
## 27   Variances    S4 0.037 0.003 10.802 0.000        Betw
```

Check Summary Statistics

```
singRun[[1]]$summaries$Parameters
```

```
## [1] 28
```

```
singRun[[1]]$summaries$LL
```

```
## [1] -1090.369
```

```
singRun[[1]]$summaries$AIC
```

```
## [1] 2236.738
```

Simulation

Sim Conditions

	Fixed Conditions
Level Two Units:	50
Level One Units:	200
Total Sample Size:	10000
Fixed Components:	Γ
Level Two Random Components:	$\tau_{00,11:33} = 0.801, 0.04$
Level One Random Component:	$\sigma^2 = 0.046$
Estimator:	Full Maximum Likelihood
	Varying Conditions
X4 Slope Variance (τ_{44}):	$0.04, 0.02, 0.01, 0.001, 0.000$
X4 Slope Variance (τ_{55}):	$0.04, 0.02, 0.01, 0.001, 0.000$

Note: Random simulation facets are fully crossed, resulting in 15 conditions

Sim Recap

- Conditions
  - 15, condition specific directories
- Data file structure:
  - dat_mplus_cond_**nCond_nRep**.dat
- Model remains the same across all conditions

Steps To Run Simulation

1. Alter single run input file
2. Use R to generate all input files via *createModels()*
3. Estimate all models via *runModels()*
4. Extract all model information via *readModels()*

Step 1: **init**

- The **init** section goes on top of your standard mplus input file.

```
[[init]]
iterators = conds rep;
conds = 1:15;
rep = 1:10;
filename = "cond_[[conds]]_[[rep]].inp";
outputDirectory =
"C:/Users/grr13002/Dropbox/Conn/Consults...
/McCoach/Simulation/c[[conds]]";
[[/init]]
```

Create Input Files

```
createModels("knitr_template.txt")
```

Run Models

- From the top level directory, we can set the recursive logical to TRUE
- The argument, *"modifiedDate"*, tells MplusAutomation to estimate models for which the modified date of the input file is more recent than its respective output file.

```
runModels(recursive = TRUE,
          replaceOutfile = "modifiedDate")
```

Read in Models

```
mplus.Extract <- readModels(recursive = TRUE)
```

- Determine all output files have been read in:
  - 150 = (10 rep * 15 conds)

```
length(mplus.Extract)
```

```
## [1] 150
```

Investigate list object

- Notice the same slot names are available as before

```
names(mplus.Extract[[1]])
```

```
##  [1] "input"           "warnings"          "errors"
##  [4] "summaries"       "parameters"        "class_counts"
##  [7] "residuals"       "tech1"             "tech3"
## [10] "tech4"           "tech7"             "tech9"
## [13] "tech12"          "fac_score_stats"   "gh5"
```

What information to store

1. All fixed components:
2. All random components:
3. Model Summaries:
   - Number of Free Parameters
   - Loglikelihood
   - Akaike's information criterion
   - Bayesian information criterion
4. Data set identifier
   - Rep & Condition Number

Tip

- Extract the needed information for a single replication
  - Once you do, its easy to do so for all 150 models

Model 2: Fixed and Random Components

- Where is the Standard Error Estimate?!

```
mplus.Extract[[2]]$parameters$unstandardized[1:6,]
```

```
##           paramHeader param    est BetweenWithin
## 1 Residual.Variances     Y  0.045         Within
## 2             Y.WITH    S1  0.012        Between
## 3             Y.WITH    S2  0.010        Between
## 4             Y.WITH    S3  0.010        Between
## 5             Y.WITH    S4  0.000        Between
## 6             Y.WITH    S5 -0.001        Between
```

- We should check the errors...

Consult Errors

- How many errors?

```
length(mplus.Extract[[2]]$errors)
```

```
## [1] 3
```

Error No. 1

```
mplus.Extract[[2]]$errors[[1]]
```

```
## [1] "THE MODEL ESTIMATION DID NOT TERMINATE NORMALLY DUE
## [2] "FISHER INFORMATION MATRIX.  CHANGE YOUR MODEL AND/O
```

Error No. 2

```
mplus.Extract[[2]]$errors[[2]]
```

```
## [1] "THE MODEL ESTIMATION DID NOT TERMINATE NORMALLY DUE
## [2] "DEFINITE FISHER INFORMATION MATRIX.  THIS MAY BE DU
## [3] "BUT MAY ALSO BE AN INDICATION OF MODEL NONIDENTIFIC
## [4] "NUMBER IS       0.541D-10."
```

Error No. 3

```
mplus.Extract[[2]]$errors[[3]]
```

```
## [1] "THE STANDARD ERRORS OF THE MODEL PARAMETER ESTIMATE
## [2] "COMPUTED.  THIS IS OFTEN DUE TO THE STARTING VALUES
## [3] "AN INDICATION OF MODEL NONIDENTIFICATION.  CHANGE Y
## [4] "STARTING VALUES.  PROBLEM INVOLVING THE FOLLOWING P
## [5] "Parameter 3, %BETWEEN%: [ S2 ]"
```

Where is the Standard Error Estimate?!

- This model did not converge, therefore, no standard errors are printed.
- This will cause problems when extracting estimates across all 150 models.
- Figure out the number of columns to make things easier later:

```
ncol(mplus.Extract[[2]]$parameters$unstandardized)
```

```
## [1] 4
```

Extract Parameter Name, Estimate, & Standard error

```
r2c1.pe <-unlist(mplus.Extract[[2]]
$parameters
$unstandardized[c(1,17:28),3])

r2c1.se <- ifelse(
ncol(mplus.Extract[[2]]$parameters$unstandardized) == 4,
NA,
unlist(mplus.Extract[[2]]
$parameters
$unstandardized[c(1,17:28),4])
)
```

Check Parameter Est. & SE Values

```
comb.PeSe <- rbind(r2c1.pe,r2c1.se)
comb.PeSe[,1:4]
```

```
##             [,1]  [,2]   [,3]  [,4]
## r2c1.pe 0.045 0.037 -0.041 0.001
## r2c1.se     NA    NA     NA    NA
```

Extract Parameter Names: The Problem

```
tail(mplus.Extract[[2]]$parameters$unstandardized)
```

```
##    paramHeader param   est BetweenWithin
## 23   Variances     Y 0.064       Between
## 24   Variances    S1 0.035       Between
## 25   Variances    S2 0.044       Between
## 26   Variances    S3 0.039       Between
## 27   Variances    S4 0.000       Between
## 28   Variances    S5 0.000       Between
```

The Fix

```
paramID.1 <- unlist(mplus.Extract[[2]]
$parameters
$unstandardized[c(1,17:28),1])
    paramID.2 <- unlist(mplus.Extract[[2]]
    $parameters
    $unstandardized[c(1,17:28),2])
paramName <- paste0(paramID.1,".",paramID.2)
```

Check Parameter Names from *paste0()*

```
sample(paramName,3)
```

```
## [1] "Variances.Y"  "Variances.S2" "Means.S3"
```

Incorporate Parameter Name and Estimates

```
colnames(comb.PeSe) <- paramName
comb.PeSe[,1:4]
```

```
##          Residual.Variances.Y Means.Y Means.S1 Means.S2
## r2c1.pe                 0.045   0.037   -0.041    0.001
## r2c1.se                    NA      NA       NA       NA
```

Extract Model Summary

- Recall that this model did not converge, therefore, there will be fewer values available from **$summaries**
  - Determine number of values

```
length(mplus.Extract[[2]]$summaries)
```

```
## [1] 7
```

```
length(mplus.Extract[[3]]$summaries) # Converged
```

```
## [1] 14
```

No. Params & LL

```
r2c1.nParm <- ifelse(
length(mplus.Extract[[2]]$summaries) < 14,
NA,
unlist(mplus.Extract[[2]]
$summaries
$Parameters)
)
    r2c1.ll <- ifelse(
    length(mplus.Extract[[2]]$summaries) < 14,
    NA,
    unlist(mplus.Extract[[2]]
    $summaries
    $LL)
    )
```

AIC & BIC

```
r2c1.aic <- ifelse(
length(mplus.Extract[[2]]$summaries) < 14,
NA,
unlist(mplus.Extract[[2]]
$summaries
$AIC)
)
    r2c1.bic <- ifelse(
    length(mplus.Extract[[2]]$summaries) < 14,
    NA,
    unlist(mplus.Extract[[2]]
    $summaries
    $BIC)
    )
```

Extract Rep and Condition IDs

```
repID <- unlist(mplus.Extract[[2]]$summaries$Filename)
```

- Notice that this is a character value, but we need the numeric value. . .

```
repID
```

```
## [1] "cond_1_10.out"
```

- Notice how the condition number is between _ & _?

strsplit()

```
splt <- strsplit(repID,c("_",".out"))
length(splt[[1]])
```

```
## [1] 3
```

```
splt
```

```
## [[1]]
## [1] "cond"   "1"      "10.out"
```

```
print(idCond <- as.numeric(splt[[1]][2]))
```

```
## [1] 1
```

Pull Numeric Values

```
splt[[1]][3]
```

```
## [1] "10.out"
```

```
print(idRep <- as.numeric(
    strsplit(splt[[1]][3],".out")[1]))
```

```
## [1] 10
```

Compile All Results

- Must create empty matrices to store all information:

1. Parameter estimates - 13 cols
2. Standard Errors - 13 cols
3. Rep & Cond IDs - 2 cols
4. Model Information - 4 cols

- Each matrix needs 150 rows, corresponding to each simulation run

Matrix Storage

```
paramE <- matrix(NA,ncol = 13, nrow = 150)
seE <- matrix(NA,ncol = 13, nrow = 150)
simR.C <- matrix(NA,ncol = 2, nrow = 150)
sumStat <- matrix(NA,ncol = 4,nrow = 150)
```

Utilize an incremental loop

- Because we want to automate this process, we must alter our code minimally:

```
for (i in 1:length(mplus.Extract)){
paramE[i, ] <- unlist(mplus.Extract[[i]]
$parameters
$unstandardized[c(1,17:28),3])
                ...

}
```

Let's Run Through It

- You should have all the necessary files.

Grab Bag

Error Variances

- Given a set of manifest variables, R can generate the appropriate Mplus syntax to model:
  - Homogeneous
  - Heterogeneous
  - Compound Symmetry
  - Toeplitz
  - Autoregressive
  - Unstructured

```
timeV
```

```
## [1] "t1" "t2" "t3" "t4" "t5"
```

MplusAutomation Implementation

- Homogeneous

```
mplusRcov(timeV,"homogenous",collapse = TRUE)
```

```
## t1 t2 t3 t4 t5 (e);
## t1 WITH t2@0 t3@0 t4@0 t5@0;
## t2 WITH t3@0 t4@0 t5@0;
## t3 WITH t4@0 t5@0;
## t4 WITH t5@0;
```

MplusAutomation Implementation

- Heterogeneous

```
mplusRcov(timeV,"heterogenous",collapse = TRUE)
```

```
## t1 t2 t3 t4 t5;
## t1 WITH t2@0 t3@0 t4@0 t5@0;
## t2 WITH t3@0 t4@0 t5@0;
## t3 WITH t4@0 t5@0;
## t4 WITH t5@0;
```

MplusAutomation Implementation

- Compound Symmetry

```
mplusRcov(timeV,"cs",collapse = TRUE)
```

```
## t1 t2 t3 t4 t5 (e);
## t1 t2 t3 t4 PWITH t2 t3 t4 t5 (rho);
## t1 t2 t3 PWITH t3 t4 t5 (rho);
## t1 t2 PWITH t4 t5 (rho);
## t1 PWITH t5 (rho);
```

MplusAutomation Implementation

- Toeplitz

```
mplusRcov(timeV,"toeplitz",collapse = TRUE)
```

```
## t1 t2 t3 t4 t5 (e);
## t1 t2 t3 t4 PWITH t2 t3 t4 t5 (rho);
## t1 t2 t3 PWITH t3 t4 t5 (rho2);
## t1 t2 PWITH t4 t5 (rho3);
## t1 PWITH t5 (rho4);
```

MplusAutomation Implementation

- Autoregressive

```
mplusRcov(timeV,"ar",collapse = TRUE)
```

```
## t1 t2 t3 t4 t5 (e);
## t1 t2 t3 t4 PWITH t2 t3 t4 t5 (rho);
## t1 t2 t3 PWITH t3 t4 t5 (rho2);
## t1 t2 PWITH t4 t5 (rho3);
## t1 PWITH t5 (rho4);
## MODEL CONSTRAINT:
##   rho2 = ((rho/e)^2) * e;
##   rho3 = ((rho/e)^3) * e;
##   rho4 = ((rho/e)^4) * e;
```

MplusAutomation Implementation

- Unstructured: $\frac{p*(p+1)}{2}$
  - p = No. of parameters

```
mplusRcov(timeV,"un",collapse = TRUE)
```

```
## t1 t2 t3 t4 t5;
## t1 WITH t2 t3 t4 t5;
## t2 WITH t3 t4 t5;
## t3 WITH t4 t5;
## t4 WITH t5;
```

Interactive Run Models

```
runModels_Interactive()
```

Reference

@hallquist2011mplusautomation